Beyond Playing to Win: Elicit General Gameplaying Agents with Distinct Behaviours to Assist Game Development and Testing

Cristina Guerrero Romero

A thesis submitted in partial fulfillment of the
requirements of the Degree of Doctor of Philosophy

School of Electronic Engineering and Computer Science
Queen Mary University of London, UK

November 2021

To my family, especially to *el Abu*.

Statement of Originality

I, Cristina Guerrero Romero, confirm that the research included within this thesis is my own work or that where it has been carried out in collaboration with, or supported by others, that this is duly acknowledged below and my contribution indicated. Previously published material is also acknowledged below.

I attest that I have exercised reasonable care to ensure that the work is original, and does not to the best of my knowledge break any UK law, infringe any third party's copyright or other Intellectual Property Right, or contain any confidential material.

I accept that the College has the right to use plagiarism detection software to check the electronic version of the thesis.

I confirm that this thesis has not been previously submitted for the award of a degree by this or any other university.

The copyright of this thesis rests with the author and no quotation from it or information derived from it may be published without the prior written consent of the author.

Signature: *Cristina Guerrero Romero*

Date: 23rd November 2021

Details of collaboration and publications are given in Section 1.4

Abstract

General Video Game Playing (GVGP) creates agents capable of playing several different games while maintaining competitive performance. Even when the generality of these agents has evident potential, there is a lack of research looking for applications for them. This work explores filling that void by advocating the integration of GVGP agents into the game development process. Additionally, it proposes studying the GVGP agents from a Player Experience perspective to facilitate their use in games as an alternative AI approach.

GVGP agents are essentially designed to win and achieve a high score. However, the players' actions are driven by different motivations, resulting in diverse behaviours. These motivations may ultimately involve winning, but it is not necessarily their primary goal. Thus, why are agents designed with merely this purpose in mind? This work considers that the path that eventually allows finding applications for the agents starts with eliciting differentiated behaviours by providing them with objectives beyond winning. It introduces the concept of *heuristic diversification* that, in the scope of search algorithms, refers to isolating the evaluation function of the controllers providing the goals externally without affecting their foundation.

This work proposes that a *team* of GVGP agents with differentiated behaviours can assist in the game development and testing processes. The solution applies *heuristic diversification* and describes the behaviour of an agent with simplicity and easiness to evolve. Diverse behaviours can be generated and used to assemble the *team* independently of the game's characteristics. Based on their stats, the resulting agents are allocated in a behavioural space, which is used to identify *behaviour-type* agents. The agents are portable between levels and facilitate diverse automated gameplay. They can detect design flaws and bugs when introducing modifications to the game or trigger external development tools without having to play the game manually.

Acknowledgements

This PhD has been a journey. I do not think I would be writing this section if I had not been surrounded by such amazing people along the way that, one way or the other, have helped me through it. Five years is a long time, and many people come and go, so it is impossible to include everyone here. I would like to start by thanking my supervisors, Diego Perez-Liebana and Simon Lucas, for their guidance. I also thank the examiners, Gillian Smith and James Walker, for the enjoyable discussion during the Viva and for providing excellent feedback.

I thank my two very dear friends, Shringi and Henrik, for sharing this journey with me, the long conversations, and their honest support. You are both wonderful people, and this PhD would have been much harder without you. I thank Ari for always being there, listening, and supporting me even in the most challenging moments. Thank you for being such a good friend. I thank Jose, Ani, and Nicole for friendships beyond the workplace. I thank Willy and Isa for caring and still being part of my life after all these years. I also thank my old football team and the friends I made that, although we do not see each other that much anymore, have been a big part of my life during the PhD, with a special mention to Hollie, Fran, Holly, and Joana.

This past year and a half have been tough, but somehow I have met some good people around the world who have helped immensely to keep it together. I thank Abs, Cel, and Payton for the games and movie nights, and I thank my *Khaos Krew* for introducing me to DnD and giving me a well-needed weekly break from my writing. I am looking forward to meeting you all in person someday.

Last but not least, I thank my family for their unconditional support. I thank my parents for always believing in me and making everything in their power so I could have the opportunity to follow the path I wanted, even when that path took me to live more than a thousand kilometres away from them. And especially, I thank my sister for supporting me to the point of reading all my papers and proofreading this very long thesis. You are the best.

Thank you all for being there.

This work was supported by the EPSRC CDT in Intelligent Games and Game Intelligence (IGGI) EP/L015846/1.

The icons included in some of the figures throughout the thesis have been made by *Smashicons*: `https://www.flaticon.com/authors/smashicons`.

Contents

List of Figures

List of Tables

List of Algorithms

List of Abbreviations

AI	Artificial Intelligence
API	Application Programming Interface
CPU	Central Processing Unit
EA	Evolutionary Algorithm
EMH	Exploration Maximisation Heuristic
EoG	End of Game
GGP	General Game Playing
GVGAI	General Video Game Artificial Intelligence
GVGP	General Video Game Playing
GPU	Graphics Processing Unit
HOLOP	Hierarchical Open-Loop Optimistic Planning
IMI	Intrinsic Motivation Inventory
KDH	Knowledge Discovery Heuristic
KEH	Knowledge Estimation Heuristic
MAP-Elites	Multi-dimensional Archive of Phenotypic Elites
MCTS	Monte Carlo Tree Search
ML	Machine Learning
NPC	Non-Player Character
OLE	Open-Loop Expectimax
OLETS	Open-Loop Expectimax Tree Search
OLMCTS	Open-Loop Monte Carlo Tree Search
OSLA	One Step Look Ahead
PCG	Procedural Content Generation
PENS	Player Experience of Need Satisfaction
PX	Player Experience
QA	Quality Assurance
RAM	Random-Access Memory
RHEA	Rolling Horizon Evolutionary Algorithm
RL	Reinforcement Learning
RS	Random Search
SDT	Self-Determination Theory
UCB1	Upper Confidence Bound
VGDL	Video Game Description Language
WMH	Winning Maximisation Heuristic
2D	2-Dimensional
3D	3-Dimensional

Part I

Introduction and Background

Chapter 1

Introduction

Video game playing approaches, and especially General Video Game Playing (GVGP), focus on creating agents with the ultimate goal of winning the game and, ideally, being the best at it. However, when human players play a game, they do not always focus on winning first. And, even when they do ultimately play to win, they present different ways to interact and react to the game, driven by their interests. During the development of a game, the existence of this diversity of player behaviours, known as player-types or *personas*, holds a relevant role in their design and the expected experience of the players. Furthermore, testing and Quality Assurance (QA) techniques are also directed to detect bugs and design flaws by asking testers to focus on specific tasks. These goals can be as simple as going through the level of a game. However, they can also be, for example, interacting with the walls and objects present in the game to detect errors with collisions, unintentional shortcuts, or other bugs that can break the gameplay. Game development is an incremental process, and new changes are included often. Every change can have unexpected effects on the rest of the game, so QA tests should ideally be carried out after a new modification is included. However, it is not always possible because manual testing is tedious and requires organisation and resources. Artificial Intelligence (AI) agents are being used for testing to address those limitations, but these approaches are typically game-dependant. As a result, the agents in use also need to be updated after certain modifications on the game to fit its changes and maintain their purpose.

In AI, a heuristic contains the criteria to follow and decide the actions to take, so it ultimately describes and controls the decisions of the agent. This thesis looks into diversifying the heuristics in existent GVGP solutions to provide the controllers with goals beyond winning to elicit a very diverse range of behaviours.

The ultimate goal of my work is to assist in the game development and testing processes by facilitating developers with a method to automatically trigger tests and tools during the development of the game. My research does not solve automated testing but introduces a new step toward tackling such a complex topic. I define a theoretical vision involving the use of general agents with a diversity of heuristics and develop a technical prototype and proof of concept based on it. I present an approach to generate a *team* of agents with differentiated behaviours, goals, and achievements that can be at the disposal of the developers to play the game automatically and, with their generality, adapt to the changes carried out during its development. Humans are irreplaceable and would still be required. My idea does not look at removing humans entirely from the equation but at providing the means to make some of the tasks related to testing and QA easier. These agents are not expected to simulate players or QA testers. Their behaviour originates from the agents focusing on different goals driven by their heuristics, accomplishing various tasks. An example would be colliding with the walls and objects distributed on the level, which, as mentioned above, is a usual task during game testing to detect collision bugs. Although the agent would not replicate game testers, it would still trigger errors and log information when playing the game, being capable of identifying such bugs quicker or immediately after a change is made to the game. I believe that having a range of agents where different behaviours and proficiencies can be identified, as well as a method to generate new ones when needed, addresses the limitations of current approaches.

I also explore the idea of taking advantage of the proposed diversification of heuristics in general game-playing agents to study their potential integration in games as Non-Player Characters (NPCs). This kind of synthesis may become possible when the general agents are analysed from the perspective of the player and the effect they have on their experience in the game, instead of looking at their performance. Related to this goal, I present an exploratory study that we carried out to evaluate the impact of the behaviour of general agents on Player Experience. This study looks into the results of various Player Experience constructs in two versions of the same game. The only difference between these two versions comes from the goal given to the general agent integrated as NPC.

The experiments are carried out in the GVGAI Framework and are focused on GVGP search algorithms. I believe the idea can be extended to cover other AI areas and solutions, but these extensions are out of the scope of this thesis. I encourage the research community to follow the lines of research opened by my work and develop them.

1.1 Scope

This thesis focuses on the study of AI in games, where the research is broad and involves several areas: Non-Player Character (NPC) behaviour learning, search and planning, player modeling, games as AI benchmarks, Procedural Content Generation (PCG), computational narrative, believable agents, AI-assisted game design, general game AI, and AI in commercial games [Yannakakis and Togelius, 2014]. Not all of these fields are of interest for this thesis, and, similarly, some of the AI solutions I describe can apply to areas different to the ones covered. Please note that I may omit or overlook some information as I focus on the relevant approaches in the scope of my work, which is limited to the following areas:

General Video Game Playing (GVGP) My research is within the scope of GVGP, so it focuses on the implementation of agents that play games. The heuristics implemented and provided to these agents are general, so they do not make decisions based on hard-coded or explicit information about the game. Their decisions are entirely transparent to the rules of the game. The heuristics consider in their calculations elements available in the framework for every game supported by it.

Search algorithms My research focuses on approaches and solutions for controllers with a forward model at their disposal, which covers tree-search and evolutionary algorithms. Therefore, I do not look into learning approaches or develop solutions for these. However, the background refers to work carried out with learning methods, as some are relevant to the motivation of my work or have served as inspiration for it. I do not look into human-like approaches, so imitation learning techniques are also out of the scope of my research.

Tile-based 2D video games My work focuses on video games, so I do not look at combinatorial, card, or board games. Only games supported by the GVGAI Framework (Section 3.1) are in the scope of my research. The games in this Framework are developed in VGDL (Section 3.1.1) and have some limitations. These are 2D tile-based games where the interaction between their sprites triggers the rules. Each game session has a limited time set by default and is formed by a non-scrollable level fixed to the game screen. The AI takes control of the player, who is represented as an avatar and can move right, left, up, down, and carry out an action. There are no physics in the games I employ. These games do not present complex rules or include puzzle or strategy games.

Single player The foundation and core part of my work are carried out in single-player games. The exploratory case study is carried out in a 2-player setup, but it is an agent-vs-player game. I do not look at multi-agent systems or collaboration between agents.

1.2 Research Questions

Current research in General Video Game Playing is mainly focused on creating new and improving existing approaches instead of taking advantage of what already exists and finding different ways to apply the game-playing solutions. Games have helped improve the algorithms to reach a high level of quality. Now, GVGP agents could be applied to video games and assist in their development by being used for automated gameplay to run tests and trigger testing tools or by taking the place of Non-Player Characters (NPCs). Most of the development of GVGP agents focuses on winning and achieving a high score. This focus could be broadened by providing the agents with goals beyond winning the game, allowing them to elicit new behaviours. This diversification of behaviours could be the key to finding new applications for these agents. Therefore, the Research Questions (RQs) I look at answering in my work are the following:

Research question 1 (RQ1) *Which general heuristics can be defined and implemented beyond the goal of winning the game, and how does each of these affect the performance and behaviour of existing GVGP agents when it is the only variation in the algorithm?*

This research question is motivated by the sentiment that the start point of ultimately allowing a successful application of GVGP agents in games and their development process is looking at providing these agents with goals beyond winning to elicit a range of behaviours. To answer this question, I first need to identify general goals that can apply to several games and define and implement their corresponding heuristics. The ultimate purpose of my work is to enlarge the research and applications of general game-playing agents, so I also need to apply them to different GVGP agents to study and compare their performance (based on the characteristics of the goal provided) and behaviour. Once this question is answered, I should have the foundation for the rest of my work.

Research question 2 (RQ2) *How to define, create, and use a team of GVGP agents with distinct behaviours to assist in the development and evaluation of games?*

This research question covers the central part of my research and is motivated by looking at applications for the GVGP agents in the game development and testing processes. To answer this question, I first need to define a long-term vision that describes the expectations that should allow these agents to be used for such a purpose. Then, I need to implement a prototype based on this concept presenting an approach capable of generating a team with an identifiable diversity of behaviours. This procedure should be flexible enough to be applied to different games and allow the agents to adapt to changes in the levels. Finally, I need to define a proof of concept that uses the agents generated to support identifying issues that may arise from such modifications.

Research question 3 (RQ3) *Can GVGP agents with distinct behaviours potentially be integrated into commercial video games as an alternative AI approach when these agents are studied from a Player Experience (PX) point of view?*

This research question is motivated by a plausible alternative application of GVGP agents to games. To answer this question, I need to define an exploratory case study to examine the impact the behaviour of general agents has on various Player Experience constructs. I need to compare the results when players are presented with two versions of the same agent-vs-player game where the only difference is the general goal of the agent included in the game. It looks at the possibility of building a bridge between GVGP and Player Experience and opening a new line of research that could ultimately allow general agents to be integrated within games as NPCs.

1.3 Contributions

The main contributions of my work can be summarised as follows:

- It broadens the research in General Video Game Playing (GVGP) by providing new applications for general agents in the scope of video games.
- It elicits differentiated behaviours in the general agents by providing them with goals beyond winning. Their play-throughs and reactions to the game based on their motivations differ to cover multiple states and situations, equipping the game developers with a rich selection of agents to use.

The detailed contributions of my thesis broken down by their corresponding RQ are the following:

RQ1

- It introduces the concept of *heuristic diversification*. It defines an approach to change the heuristics in search controllers by isolating their evaluation function without modifying the core of the algorithm. This is the first step in the path of broadening the research in GVGP and serves as the foundation for the rest of the work.
- It identifies general heuristics beyond playing to win and presents their implementation and application to different GVGP agents. These heuristics can be applied to games with various characteristics. Different motivations can drive the way a game is played, so I include this idea in the study of general agents to expand their applications beyond merely learning to play games to win.
- It studies multiple GVGP agents when, by *heuristic diversification*, they are provided with different general heuristics and goals. It compares the resulting performance (based on the characteristics of the goal provided) and behaviour of the GVGP agents when the heuristic is the only modification in their algorithm.

RQ2

- It defines a theoretical methodology and implements a technical proof of concept based on that long-term vision. It proposes using a *team* of GVGP agents with different behaviours to assist in the development and testing of video games. It reviews existing literature, presents the elements and steps that the methodology should cover, and discusses its strengths and limitations.
- It includes a list of heuristics that describe behaviours that can be identified in games and can be applied to the general agents.
- It presents an approach to creating the *team* of agents proposed in the methodology. It applies the MAP-Elites algorithm to generate different behaviours for GVGP

agents and implements a general solution in the GVGAI Framework. In contrast to existing solutions, the agents are given diverse behaviour by their heuristics and identified after the execution of the algorithm. The description for each agent is generated and located in a behavioural space based on the gameplay results when provided with it.

- It proves the validity and generality of the approach by applying it to four games with differentiated characteristics, generating a *team* of agents with differentiated behaviours for each of them.
- It implements an provides an interactive tool to go through the resulting MAP-Elites, including details about each of the agents generated and a pre-recorded preview of their behaviour during gameplay [Guerrero Romero, 2021e].
- It introduces the concept of *behaviour-types*. In contrast to *personas* or *player-types*, these agents do not simulate human players or aim to imitate their gameplay. These agents are proficient in particular aptitudes related to the game as they are expected to obtain particular resulting stats during gameplay, comparable to the achievement of different tasks when playing the game. Each of these agents allows testing different interactions with the game based on their corresponding *behaviour-type*. They do not simulate play tester behaviour either, but the tasks usually carried out by QA testers can be used to inspire the identification of these agents.
- It presents a procedure to identify agents of different *behaviour-types* from the MAP-Elites generated. It would still be applicable if the MAP-Elites approach is used in other frameworks or games. This selection is carried out based on the location of the agents in the behavioural space and their resulting stats.
- It demonstrates that the *behaviour-types* agents identified are general enough to be transferable to levels of the game different to the one they were generated for.
- It proposes and carries out an exploratory experiment to use the identified *behaviour-types* agents from the *team* for testing new 'broken' levels and includes a discussion about this application. The generality of the approach developed makes it applicable to other frameworks and games.

RQ3

- It proposes a new line of research that studies GVGP agents from a new perspective, looking at their impact on Player Experience instead of the quality of the algorithms. The quality of the AI algorithms is usually measured by how well the agent performs.
- It includes an exploratory work aimed to build a bridge between the study of General Video Game Playing (GVGP) agents and Player Experience (PX) to ultimately

allow the integration of those agents in commercial video games. In this preliminary study, general agents are integrated into a game as Non-Player Characters (NPCs).

- Applies *heuristic diversification* to a 2-player controller from the GVGAI Framework.
- It introduces a new game developed for the GVGAI Framework: *Skulls and Tombstones*, used in the case study.

1.4 Publications

The following list of published papers constitutes the work detailed in the thesis. The portion of the thesis they contribute to is listed in bold. Please note that they also add to the introduction, background, and conclusions.

1. Cristina Guerrero-Romero, Annie Louis and Diego Perez-Liebana. *Beyond Playing to Win: Diversifying Heuristics for GVGAI.* Proceedings of the 2017 IEEE Conference on Computational Intelligence and Games (CIG), 118-125, 2017 [Guerrero-Romero et al., 2017].
   Contributions: Implemented, carried out the experiments, processed the results, and wrote the paper. The second and third authors participated in the discussions and helped with shaping and polishing the paper.
   Included in Chapter 4.

2. Cristina Guerrero-Romero, Simon M Lucas and Diego Perez-Liebana. *Using a Team of General AI Algorithms to Assist Game Design and Testing.* Proceedings of the 2018 IEEE Conference on Computational Intelligence and Games (CIG), 1-8, 2018 [Guerrero-Romero et al., 2018].
   Contributions: Wrote the paper. The third author participated in the discussions and helped with shaping and polishing the paper. The second author helped with polishing the paper.
   Included in Chapter 5

3. Cristina Guerrero-Romero*, Shringi Kumari*, Diego Perez-Liebana and Sebastian Deterding. *Studying General Agents in Video Games from the Perspective of Player Experience.* Proceedings of the AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment (AIIDE), 217-223, 2020 [Guerrero-Romero et al., 2020].
   Contributions: Implemented the code, shaped and carried out the study, processed the results, and wrote the paper. It was an equal collaboration with co-author Shringi Kumari, as each of us brought our own expertise in the two fields the paper brings together: Player Experience and General Video Game Playing and heuristics. The third and fourth authors helped with polishing the paper.
   Included in Chapter 8

4. Cristina Guerrero-Romero and Diego Perez-Liebana. *MAP-Elites to Generate a Team of Agents that Elicits Diverse Automated Gameplay.* Proceedings of the 2021 IEEE Conference on Games (CoG), 2021 [Guerrero-Romero and Perez-Liebana, 2021].
   Contributions: Implemented, carried out the experiments, processed the results, and wrote the paper. The second author participated in the discussions and helped with polishing the paper.
   Included in Chapter 6

The following paper has been submitted to IEEE Transactions on Games, and it is being reviewed at the moment of the thesis submission. Similarly to the published papers, it also adds to the introduction, background, and conclusions.

1. Cristina Guerrero-Romero, Simon M Lucas and Diego Perez-Liebana. *Beyond Playing to Win: Creating a Team of Agents with Distinct Behaviours for Automated Gameplay.* 2022.
   Contributions: Implemented, carried out the experiments, processed the results, and wrote the paper. The third author participated in the discussions and helped with polishing the paper. The second author helped with polishing the paper.
   Included in Chapters 6 and 7

Other publications related and resulting from my main work but that do not contribute to the core of the thesis are the following:

1. Damien Anderson, Cristina Guerrero-Romero, Diego Perez-Liebana, Philip Rodgers and John Levine. *Ensemble Decision Systems for General Video Game Playing.* Proceedings of the 2019 IEEE Conference on Games (CoG), 1-8, 2019 [Anderson et al., 2019].
   Contributions: Provided code of some of the heuristics that have been adapted and used in the experiment and participated in the discussions. Helped with creating the resulting graphs, writing, shaping, and polishing the paper.

2. Diego Perez-Liebana, Cristina Guerrero-Romero, Alexander Dockhorn, Dominik Jeurissen and Linjie Xu. *Generating Diverse and Competitive Play-Styles for Strategy Games.* Proceedings of the 2021 IEEE Conference on Games (CoG), 2021 [Perez-Liebana et al., 2021].
   Contributions: Provided code of the MAP-Elites solution that has been adapted and used in the experiment. Helped with polishing the paper.

1.5 Structure

The main contributions of the thesis are described in Part II, structured in three blocks: The first one, included in Chapter 4, presents the foundation, *heuristic diversification.* The second block is covered in Chapters 5, 6 and 7, and corresponds to the main work of the thesis, which describes a long-term vision and implements a technical proof of concept based on it. The last block is comprised of Chapter 8, which introduces a new line of research with initial exploratory work. Each of these differentiated sections focuses on a research question. Overall, the thesis is structured as follows:

Chapter 2 reviews literature related to the work presented in the thesis and its motivation. It goes through the evolution of the research in AI and games, lists the primary types of game-playing agents in the field, and introduces General Video Game Playing (GVGP) and the list of frameworks that facilitates its research. It also discusses the existence of differentiated behaviour during gameplay, existent methods to identify them, and the approaches followed to apply these behaviours to game-playing AI. Lastly, it includes methods that propose the use of game-playing agents for automated testing and a high-level overview of the AI techniques commonly applied to commercial games.

Chapter 3 lists and describes in detail the framework, algorithms, and games used.

Chapter 4 presents the foundation of the work. It introduces the term of *heuristic diversification* and defines a series of general game-playing goals to apply to different controllers to compare them in terms of performance and behaviour. The approach introduced and the results obtained serve as inspiration for the rest of the work.

Chapter 5 presents the vision of using a *team* of general agents with different behaviours to assist in the development and testing processes of games.

Chapter 6 presents and implements an approach to generate the *team* of agents with distinct behaviours envisioned by applying the MAP-Elites algorithm. The method is applied to four games, validating its usability across games of different types and characteristics. It proposes using the resulting agents to trigger automated gameplays of the game with various tasks.

Chapter 7 reviews the *team* generated in previous work and identifies agents eliciting different behaviours and tasks. It tests the portability of the proficiency exhibited by each of the agents in four new levels with differentiated characteristics and, given the success of the results, proposes using the *team* of agents to test new levels. It presents a preliminary work employing this concept.

Chapter 8 presents an exploratory experiment that diverges from the main line of work to propose studying the quality of general agents with distinct behaviours from a player experience perspective; to ultimately integrate GVGP agents as NPCs in video games.

Chapter 9 concludes the thesis. It gives an overview of the work presented and discusses the unexplored line of research and future open work.

Appendix A contains details about the games from the GVGAI Framework used in the experiments without modifications.

Appendix B contains a list of the resources related to each of the experiments executed. It includes a list with links to the code and results, relevant demos, and details about the corresponding appendices.

Appendices C, D and E gather the results obtained in the experiments presented in Chapters 4, 6, and 7.

Appendix F includes the material used for the Case Study presented in Chapter 8.

Chapter 2

Background

Artificial Intelligence (AI) is the study of designing and developing artefacts capable of presenting an *intelligent behaviour*, which involves perception, reasoning, learning, communicating, and acting in complex environments [Nilsson, 1998]. Intelligence is defined by McCarthy [1998] as "*the computational part of the ability to achieve goals in the world*". Many disciplines include the study the AI, but this thesis focuses in AI in games. For a detailed overview of AI and games and a full collection of the existing methods and applications, I recommend referring to specialised books [Russell and Norvig, 2021, Millington, 2020, Yannakakis and Togelius, 2018].

This chapter presents and describes the topics related to my work. It introduces the use of games as benchmarks for AI solutions, the evolution of game-playing AI techniques, presents the primary type of algorithms in the field, and identifies the ones in the scope of my research. It covers General Video Game Playing (GVGP) and the list of frameworks that facilitates research in general game AI. It also discusses the existence of differentiated behaviour during gameplay, existent methods to identify them, and the approaches followed to apply these behaviours to game-playing AI. Lastly, it covers current automated play-testing approaches and a high-level overview of some techniques used in commercial games and their development process that are relevant to the motivation of my work.

2.1 Definitions

I define some concepts used in this chapter and throughout the thesis. Some chapters also include a list of definitions that are relevant specifically to each of them.

Sprite 2-Dimensional (2D) visual representation of an element of a game. The sprite that refers to the player is called *avatar*.

Game state Representation of a game in a precise moment in time that includes information about its characteristics at that moment, e.g. player(s) location and status, score, and distribution of the elements, among others.

Agent An entity capable of carrying out particular actions in an environment to achieve a goal. When applying this concept to games, the goal is generally winning and, therefore, the game-playing agent will take a series of decisions to reach the victory. It bases these decisions on its knowledge of the surroundings and the state of the game. As a result, an agent is developed and optimised for the game under consideration to ultimately achieve a performance comparable to human players and overtake them.

Controller Alternatively term used for *agent*.

Search space Set of possible solutions. It is constituted by different game states and the plan of actions required to reach them. It is also called *state space*.

Heuristic As defined by Pearl [1984]: *"Heuristic are criteria, methods, or principles for deciding which among several alternative courses of action promises to be the most effective in order to achieve some goal. They represent compromises between two requirements: the need to make such criteria simple and, at the same time, the desire to see them discriminate correctly between good and bad choices"*.

In the context of games, the ultimate goal is generally to win, but it is not necessarily true in every case. How victory or any other objective is achieved depends entirely on the characteristics and rules of the game considered. The knowledge about the game and the tricks to beat it may be included as heuristics of the game-playing agents. This information allows the algorithms to explore the search space in a certain way, serving as a guide to achieving their goals. How these heuristics are defined and implemented affects the agents' decisions while playing the game, shaping their behaviour.

Heuristic value/reward Number that represents the quality of a game state. The higher the number is, the better, as it is considered closer to fit the criteria.

Evaluation function Method operated by the agent to obtain/asses the heuristic value of a game state.

Forward model Function available in some systems to simulate future states of the game without playing it, generally used to calculate the heuristics of the potential plan of actions and make a decision about the next move. In deterministic games, the simulated states are expected to be an accurate representation. In stochastic games, a random seed is also involved in the simulations, so the future states are an approximation.

Search algorithm When I refer to search algorithms in this thesis, I allude to game-playing controllers that have a forward model at their disposal. It encompasses both tree-search and evolutionary algorithms. I also use the term *planning algorithm*.

2.2 Game-Playing Artificial Intelligence

Games have helped develop and improve Artificial Intelligence (AI) for a long time. They are formed by a set of well-defined rules that allow the existence of a finite number of states, which provide information about the environment at every game tick. The state space can be very extensive and depends on the complexity of the game. Games provide interactive problems to solve in a controlled environment. As a result, they are commonly used as a benchmark to check the quality of the solutions, being possible to transfer the validated approaches to other disciplines and the real world.

Even before computers were built as they are known today, visionaries suggested the possibility of using machines to play and master the game of *Chess*, being this game one of the firsts to get the interest of the researchers. It was considered as an appropriate start point to ultimately reach the creation of machines capable of solving more significant problems on account of a series of reasons listed by Shannon [1950]: Firstly, *Chess* is a well-defined problem in terms of the allowed actions (moves) and the goal to achieve (checkmate). Secondly, the solutions are not trivial or too complicated, but a skilful play requires planning. Lastly, the game structure fits well in computation, being viable to create a program capable of playing successfully. Turing [1953] described a series of rules that fundamentally defined a computer program to play *Chess*. However, given the means of the time, it was not possible to execute it in a machine. For years, and with the born of modern computers, researchers put a lot of effort into looking for methods to build a system capable of playing *Chess* with good performance. However, the problem was not solved until 1997, when *Deep Blue*, a machine built by IBM, defeated the World Chess Champion Garry Kasparov in a six-game match [Campbell et al., 2002]. This event is the first landmark in Artificial Intelligence.

Since the peak of achievements in AI of the late 90s (*Checkers* and *Othello* were also solved [Schaeffer, 1997, Buro, 1997]), new challenges and ambitious goals have appeared over the years. *Go* took the attention of the community due to its complexity; it requires years of learning and practising for a human to develop the skills to become a good player. From a computer program point of view, the size of the search space of the game is immense, becoming very difficult to evaluate moves and come up with strategies. For a long time, it was considered the new biggest challenge in AI, so a new milestone was reached in 2016 when *AlphaGo* [Silver et al., 2016] defeated Lee Sedol, recognised as the best *Go* player of the decade.

All these achievements refer to combinatorial games. Combinatorial games are two-player, finite, discrete, and deterministic with perfect information. These characteristics make these games a good start point for every AI research, but it is not limited to this kind. AI research also covers the study of imperfect-information games, where not all the information is available to the player, like *Poker* [Billings et al., 1998, Brown and

Sandholm, 2019] or *Hanabi* [Bauza, 2014, Bard et al., 2020]; and stochastic games where there is an element of chance involved (e.g. rolling the dice), like *Backgammon* [Berliner, 1977, Tesauro, 1994]. It has even been extended to the study of video games, where the types and characteristics are wide-ranging. The visual aspect associated with video games facilitates the comparison between algorithms, so they have become a recognised benchmark for AI controllers leading to the creation of several competitions. These competitions are very diverse, covering a wide range of games and problems to solve. Some examples are the creation of drivers for car races [Loiacono et al., 2010], players for classic games like *Ms. Pac-man* [Midway, 1982, Lucas, 2007], first-person shooters like *Doom* [id Software, 1993, Wydmuch et al., 2018], and real-time strategy games like *Starcraft* [Blizzard Entertainment, 1998, Ontanón et al., 2013]. Some competitions have gone a step further and, in addition to the development of agents capable of playing a certain game (*Super Mario Bros.* [Nintendo, 1985] in this case), they include tracks to create human-like players or to generate levels [Togelius et al., 2013]. The diversity of challenges results in a variety of AI approaches that have developed over time.

In summary, the interest of researchers in AI has evolved over the years. The increase of computational power has facilitated the origin of various approaches and their improvement and development over time. The algorithms used to create game-playing agents can be classified into three types: tree-search, evolutionary, and learning algorithms. The latter are out of the scope of my work (Section 1.1).

2.2.1 Tree-Search algorithms

Tree-search algorithms solve challenges in real-time and have a forward model available to them that allows simulating future states by providing state-action pairs. These algorithms build a tree representing feasible plans of action, called a search tree (Fig. 2.1). Each node of this tree symbolises a simulated state and is reached by taking an action from a previous one. The root is the current state of the game, which serves as a starting point, and its distance to a node of the tree determines the depth of that node. The agent can estimate how the game changes as the different actions play out. It decides which action to take next based on the rewards obtained in the future states, calculated by an evaluation function.

There are different ways to visit the nodes of the tree. Uninformed search looks into the state space without information about the goal, and the basic algorithms are Breadth-first search (BFS) and Depth-first search (DFS). BFS explores the tree transversally, expanding the nodes at a certain depth before exploring the next level. DFS, on the contrary, explores vertically, increasing and expanding from each node in depth until a maximum one is reached.

In terms of using information about the goal (heuristics) to guide search, the various approaches differ in the building and navigation of the tree, the existence of a depth or

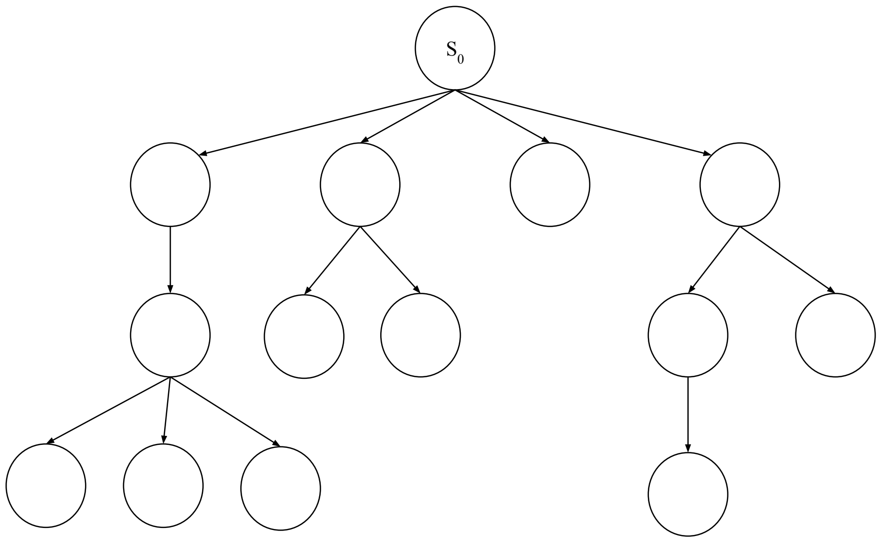


Figure 2.1: Search tree: The root node S_0 represents the state of the game where the algorithm starts. The nodes represent simulated states and, the edges, the actions that lead to them.

time limit, and the policy that leads to decide the final action.

In my work, I use three agents that belong to this group: One Step Look Ahead (OSLA), Monte Carlo Tree Search (MCTS) and Open-Loop Expectimax Tree Search (OLETS), presented in detail in Section 3.2.

2.2.2 Evolutionary algorithms

Evolutionary Algorithms (EA) look at optimising a problem and are based on Darwin's natural selection. These algorithms do not build a search tree because they consider complete solutions and not the path followed. However, they have access to a forward model to simulate states and calculate the rewards (fitness) corresponding to each of the solutions, also called *candidates*. The group of solutions taken into consideration by the algorithm at each point in time is called *population*. Evolutionary algorithms adopt a similar *template* [Yannakakis and Togelius, 2018, Chapter 2] that starts with the *initialisation* of the population. The fitness of each solution (obtained by their *evaluation*) and other criteria are used to choose various candidates (*parent selection*) that ultimately lead to the next generation by *reproduction*. The offspring may be generated through crossover, simply duplicating the candidates, or by a combination of both. Mutation may also be applied to the parents and new solutions to introduce *variation*. The new generation is formed by selecting the parents and offspring based on a particular criterion and strategy. These steps can be repeated until a *termination* condition is met, resulting in the final population. An overview of EA is shown in Fig. 2.2

Crossover combines the parents by selecting different components of each and joining them to create a new solution. Mutation, on the other hand, alters one or multiple elements of a solution. Various techniques can be used for crossover and mutation [Eiben et al., 2003].

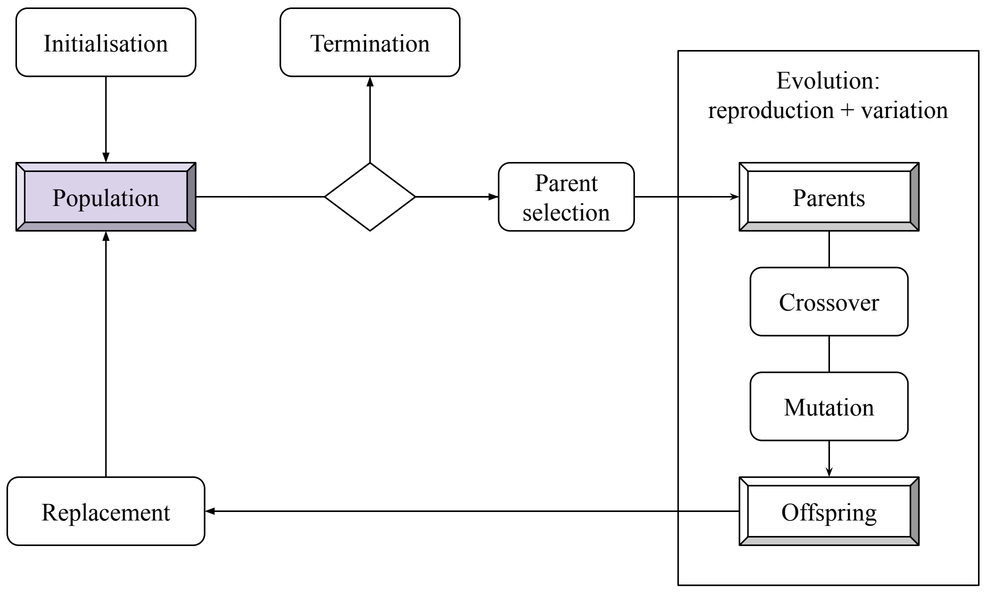


Figure 2.2: Evolutionary Algorithm overview; based on flowchart in [Eiben et al., 2003].

In my work, I use two agents of this type: Rolling Horizon Evolutionary Algorithm (RHEA) and Random Search (RS), detailed in Section 3.2.

2.3 General Video Game Playing (GVGP)

The research on Artificial Intelligence and its application in games are extensive and promising. However, all the agents developed for the challenges and competitions that have been mentioned until this point have a definite limitation: they are created having a specific game in mind. Having all the information about the game available helps programmers and AI designers to take advantage of that knowledge to tweak the algorithms and guide the AI in a specific direction. For instance, when creating drivers for the *Simulated Car Racing Championship*, programmers know tricks to overcome other cars in certain circumstances, a piece of knowledge they use and include in their heuristics. Similarly, when creating an agent for the *Mario AI Championship*, they are aware that colliding with certain elements would kill them, but others would give them a boost or points. Lastly, *AlphaGo* can defeat the top human *Go* player of the decade. However, if the algorithm is provided with a similar interface to compete in a simpler game like *Tic-tac-toe*, it would not be able to play and succeed without being previously specifically trained to do so. All these agents are developed exclusively for those games and, although their design and implementation are still not simple, the fact mentioned above opens the discussion on whether or not the algorithms are really *'intelligent'*.

General Game Playing (GGP) tries to solve this problem and refers to the creation of algorithms that are general enough to have a good performance in numerous games instead of focusing on just one. These algorithms should be able to play a game without having previous knowledge about it or being aware of its rules. Pitrat [1968] was the first to propose the concept of GGP studying the use of a program with the capacity of playing various combinatorial games. This program is informed about the moves and win condition and, without knowing the rules of the game, must find the plan that leads to its victory while avoiding dangerous moves. Years later, Pell [1996] created the first practical GGP program. They also introduced the concept of *Metagame* [Pell, 1992]: the idea of developing programs capable of taking the rules of any game within a well-defined class as input and play against opponents. The start point was generalising a set of games already studied in game-playing and transferring some game-specific methods to a general problem. Later on, they would look at generalising to any game. They defined a class capturing the aspects of most of the *symmetric chess-like* games, built a *game generator*, and created a *move grammar*, which is a language that allows players to communicate their moves when playing the games. The game generator was intended to prevent the developers from focusing on specific rules, making them look at a general representation of their knowledge instead. They used this generator to produce an example, analyse it, and determine that complex rules can offer an interesting strategic analysis. However, the process used to study the games and build game-playing solutions did not work when looking for a general-purpose evaluation function. The conclusion reached was that the generator created served as proof of the possibility of providing the means to build and test general game-playing approaches. However, the techniques used to build AI relied on the programmers' capability to understand the rules of the game and build evaluation functions accordingly. Therefore, the problem they were trying to tackle may be beyond the possibilities of the time.

Since this first attempt in the early 90s, AI has evolved, benefiting the research in GGP as well. In 2005, the Stanford Logic Group of Stanford University developed the first General Game Playing framework: *Gamemaster*, a system capable of accepting a formal description of a game and playing it effectively without human intervention. They organised the *AAAI GGP Competition* [Genesereth et al., 2005], where participants built general agents for finite, synchronous games. These were written using a Game Description Language (GDL), similar to *Prolog* in syntax, where it was possible to specify player roles, initial and goal states, legal moves, and state changes. In *Gamemaster*, game-playing agents do not know the rules beforehand as they receive the declarative descriptions of the game at runtime. The system includes a game manager responsible for running the games, communicating with the players, and making sure the controllers follow the rules of the game (Fig. 2.3). The competition ran for several years, receiving between 10 to 15 general game players entries annually, becoming more sophisticated and powerful every year [Genesereth and Björnsson, 2013]. Some innovations were put in place during this time. The most significant ones were the introduction of Monte Carlo Tree Search (MCTS) methods, described in Section 3.2.2, and

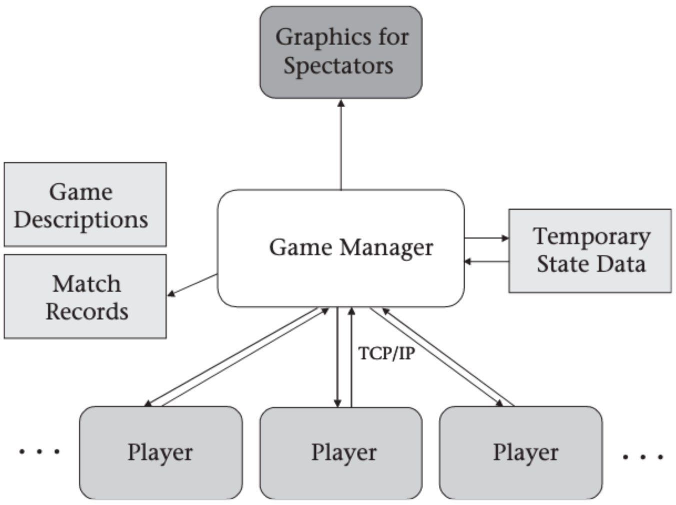


Figure 2.3: Details of the Game Manager; from [Genesereth et al., 2005]. It is part of the *Gamemaster* system used in the GGP Competition and is responsible for running the games, communicating with the players, and ensuring the controllers follow the rules.

the use of game-independent heuristics. They created heuristics introducing mobility (number of legal moves), inverse mobility (limiting opponents' freedom), and goal proximity. Although agents including these heuristics performed better than *random*, they did not have a good performance overall. Therefore, there was still research to be done in this area. The authors concluded that, even when some interesting technology emerged from the competition, it was insufficient to the GGP scope, resulting in them not being applicable to real-world problems. Thus, there was room for modernisation and progress.

During the last few years, the idea of General Game Playing has been extended to real-time games, originating the concept of **General Video Game Playing (GVGP)** [Levine et al., 2013]. Researchers of this field aim to develop algorithms capable of playing video games without having prior knowledge about them, their rules, or the environment and where agents have mere access to the state and the actions available.

Video games come with a diverse spectrum of characteristics and genres, ranging from 2-dimensional single-player with constrained levels to 3-dimensional multi-player open-world games. This challenge has driven the creation and existence of several platforms and competitions available to assist the research community.

2.3.1 Frameworks for research in GVGP

General Video Game Playing (GVGP) is ongoing research applied to several areas of games (single or multi-player, collaborative or competitive, deterministic or stochastic, etc.), which increases the complexity of the generalisation sought. The interest in this challenging area of research has grown over the past years. Thus, there is an active community of researchers aiming to tackle the problem, leading to the existence of multiple open-source **Frameworks** to encourage its study. The most important ones are the Arcade Learning Environment (ALE) [Bellemare et al., 2013], General Video Game AI (GVGAI) Framework [Perez-Liebana et al., 2019b], OpenAI Gym [Brockman et al., 2016], Project Malmo [Johnson et al., 2016], and Unity ML-Agents [Juliani et al., 2018], among others.

Arcade Learning Environment (ALE) It is a platform to evaluate general, domain-independent AI agents making use of 55 *Atari 2600* games, like *Space Invaders* and *Ms Pac-Man* [Bellemare et al., 2013]. It provides an interface to work as a benchmark for planning and learning problems, but most of the research carried out using this framework focuses on RL. Results obtained suggested that general Atari game playing was challenging but manageable, having the potential to help the development of general agents. Mnih et al. [2015] used Deep Q-Networks (DQN) to play 49 of the games available by having access only to the pixels of the screen and the score. The AI managed to achieve a level of performance comparable to a professional human in many of the games. Although ALE also supports planning algorithms, the research done in that area is lacking; presumably due to the complexity of finding heuristics general enough to work for every game [Machado et al., 2018].

General Video Game AI (GVGAI) Framework It is an open-source platform that facilitates the research in GVGP in single or two-player 2D arcade games. It supports the study of planning and learning approaches and has been used for several competitions, resulting in the availability of a range of agents [Perez-Liebana et al., 2019b]. These agents are general within the framework and can play any of its games. I use this framework to carry out my work, and it is described in detail in Section 3.1.

OpenAI Gym It is a toolkit that provides a common interface for a collection of environments based on pre-existent RL benchmarks to test learning approaches [Brockman et al., 2016]. This collection is growing over time and includes, among others, ALE and the learning track of the GVGAI Competition. In contrast with other frameworks, *OpenAI* provides an abstraction for the environment instead of the agent and does not include a hidden test set. Instead of arranging competitions, it encourages peer review and collaboration by sharing the code and a description of the approach followed. The framework focuses on both the performance of an algorithm and the amount of time it takes to learn.

Project Malmo It is a platform designed to support general AI research in RL, planning, multi-agent systems, computer vision, and robotics [Johnson et al., 2016]. It is built on top of *Minecraft*, so agents are exposed to a complex 3D environment. In 2017, this environment was used to run the *Malmo Collaborative AI Challenge*, focused on collaborative AI. The goal was to create agents capable of learning to achieve high scores when working with artificial or human players. In 2018, they announced the *Multi-Agent Reinforcement Learning in MalmO (MARLO) Challenge* [Perez-Liebana et al., 2019a] to encourage research in multi-agent reinforcement learning using multiple 3D games.

Unity ML-Agents It is a framework to use and integrate learning approaches onto games [Juliani et al., 2018]. It includes a series of pre-existent reinforcement and imitation learning algorithms that allow using these techniques in environments built on Unity or creating personalised ones with *TensorFlow*. A series of tutorials and existent environments serve as an example and starting point. This platform has been used to run the *Obstacle Tower Challenge* [Juliani et al., 2019] to test the capabilities of generalisation of AI agents when navigating through a procedurally generated environment.

Table 2.1 summarises the critical information about each of these frameworks. Most of the toolkits have tutorials and sample agents at the disposal of the researchers, but in some of them, it is not possible to customise the games or update the existing ones. It was important to use a framework that, apart from providing sample agents, would allow me to access the games and rules to adapt them when needed. Furthermore, my work focuses on planning algorithms, so it was essential to use a framework with support for agents with a forward model. The GVGAI Framework covers all my needs. Although the games it supports are simpler and more limited than the ones in other frameworks, my research is the first step in a larger vision, and that simplicity is enough to create a proof of concept. I believe the approaches designed and implemented could be extended to more complex systems in the future with further research.

	ALE	**GVGAI Framework**	**OpenAI Gym**	**Project Malmo**	**Unity ML-Agents**
Research	Mainly RL	Planning Learning	RL	RL Planning Computer vision Robotics	RL Imitation learning Neuroevolution
Games	*Atari 2600*	2D arcade	Common interface to existing research environments	*Minecraft*	2D, 3D, VR/AR
Players	Single player	Single or 2-player	-	Multi-player	Single or multi-player
Agents interface	Screen pixels	Game state, Forward model	Abstraction of the research environment	Minecraft environment + API	Learning environment + Python API, *Tensorflow*
Customisable games	-	VGDL	-	-	Unity
Competition	-	GVGAI Competition	-	MARLO Challenge	Obstacle Tower Challenge

Table 2.1: Overview of the main frameworks that are used for research in GVGP: Arcade Learning Environment (ALE), General Video Game AI (GVGAI) Framework, OpenAI Gym, Project Malmo, and Unity ML-Agents. It summarises the essential information about each: the type of research they are used for, the games and number of players they support, the interface presented to the agents, the language used for the games if they support the creation of customisable ones, and the corresponding competition if any.

While the scope of my research is limited to GVGP, it is worth mentioning two recent frameworks oriented to the research of GGP in card and board games. These systems are designed to study state-of-the-art methods that could apply to video games in the future: OpenSpiel [Lanctot et al., 2019] and Polygames [Cazenave et al., 2020].

OpenSpiel It is a framework to encourage the research on general reinforcement learning and search approaches in card and board games [Lanctot et al., 2019]. It supports a wide range of games with different characteristics: single/multiplayer; zero-sum; cooperative and general-sum; one-shot and sequential; both turn-taking and simultaneousmove; games with perfect and imperfect information; and multi-agent environments like grid worlds and social dilemmas. The platform provides an extensive list of games and algorithms and supports the creation of new ones.

Polygames It is a framework focused on zero-learning techniques. It contains a compilation of games and checkpoints that allow quick self-training of agents with good results [Cazenave et al., 2020].

The existence of these (and other) platforms to study general agents implies that there is a huge variety of algorithms with different characteristics, weaknesses, and strengths at everyone's disposal. The most common techniques to tackle and develop these agents are Tree-Search, Evolutionary Algorithms and Reinforcement Learning, introduced in Section 2.2. The level of generality of the agents depends on the heuristics included in the algorithm, which, in most cases, are focused on winning the game and maximising the score achieved. The idea behind the research in GVGP is the creation of high-quality algorithms that perform well in different games, ensuring the strength comes from the approach and not from a game tailored hand-crafted heuristic.

2.3.2 Heuristics for GVGP agents

When facing the design of players for General Video Game Playing (GVGP), it is not possible to apply game knowledge into the algorithms, as the rules and characteristics of the games are unknown. Therefore, the chances of writing heuristics that can successfully guide search in every game are significantly reduced. When designing heuristics in the context of generality, it is essential to keep in mind that the agents would make decisions without knowing if these put them closer to their ultimate goal. It is necessary to compromise on their priorities and the information they rely on to make these decisions. Heuristics are the basis of every general algorithm and, therefore, should be implemented carefully.

Most of the heuristics used in GVGP solutions focus on **winning and maximising the score** when a win condition is not reached. The limitation of this approach arises when it is necessary to carry out specific actions to get to the winning states, the change in score is not directly connected to winning, or the rewarding states are not reachable. When the agent cannot get enough information to make the right decisions to win, their behaviour would be erratic and their efforts unsuccessful. By looking at the agents submitted to the GVGAI Competitions over the years [Perez-Liebana et al., 2019b], I can attest how even with the existence of a range of controllers and approaches, the use of alternative heuristics to winning and maximising the score is practically nonexistent. However, some authors have looked into expanding the description of the heuristics to search the game space more effectively and overcome the limitations.

Exploration Perez Liebana et al. [2015] implemented a nature-inspired technique based on pheromones that would prevent the agent from staying in the same position too often, hence encouraging the physical exploration of the level. Each of the locations of the grid is assigned with a number between 0 and 1, corresponding to the number of pheromones. Every game tick, the pheromones in the proximity of the current and recent positions of the avatar increases, creating a high concentration of them in its proximity (Fig. 2.4). The amount of pheromones assigned to a cell decays with time. Although the priority of the heuristic is winning with a high score, it rewards reaching those locations where the pheromones value is small. It also penalises actions that originate *blocked movements* (i.e. the position of the avatar does not change) or *opposite actions* (i.e. performing two consecutive contrary movements like *Up* and *Down*). Nielsen et al. [2015] created a much more straightforward exploratory heuristic that stores the information about the visited locations of the level and simply encourages visiting new ones.

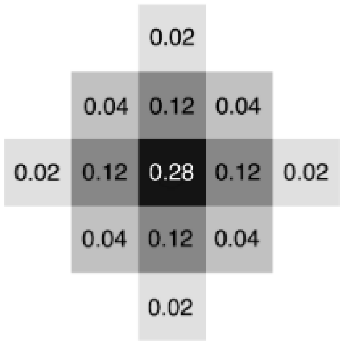


Figure 2.4: Pheromone diffusion; from [Perez Liebana et al., 2015]. Each cell is assigned a number of pheromones. It is a value between 0 and 1 that decays over time.

The exploratory heuristic based on pheromones is used in a study that compares the performance of Multi-Objective (MO) approaches in GVGP [Perez-Liebana et al., 2016]. Multi-Objective approaches look into optimising different objectives simultaneously to overcome the difficulties given by using only one when not enough information may be provided by it. In that experiment, the authors use the level exploration and score as the two heuristics applied to their Multi-Objective implementation of Monte Carlo Tree Search (MCTS). They created three different versions of MO-MCTS and compared them to a single-objective MCTS. Results show an improvement in the performance when using Multi-Objective versus the Single-Objective MCTS. However, interestingly, these results also show that just switching between objectives uniformly at random and one at a time obtains decent results in many games. MCTS is one of the algorithms I use in my research, and it is detailed in Section 3.2.2.

Knowledge gain An innovative approach in GVGP is using heuristics to acquire knowledge of the level while it is being played. This procedure was first proposed in [Perez et al., 2014]. The motivation came from the limitations found when using the MCTS algorithm in a general video game environment. Firstly, it is not possible to reuse past information from the events that cause changes in the score. Secondly, some sprites of the game are never reached by future simulations. Lastly, the outcome of the interactions with the sprites are unknown, and, as a result, the agent has no information about the ones that should be targeted or avoided. The Knowledge-based Fast Evolutionary (KB Fast-Evo) MCTS is introduced to tackle these issues by taking advantage of past experiences and using the interactions with the sprites to estimate the value resulting from colliding with them. Two new concepts are defined: *curiosity*, which refers to discovering the effects of the agent colliding with other sprites of the game; and *experience*, which refers to rewarding those events that cause a positive change in the score. MCTS is combined with an evolutionary technique to guide the simulations during its execution. The evaluation function prioritises the actions that lead to a score gain, followed by those that provide more information to the knowledge database or lead the avatar closer to sprites that provided a score gain in the past. A series of revisions of the KB Fast-Evo MCTS have been presented. An improvement proposes to replace the Euclidean distance employed as a feature in [Perez et al., 2014] by a method that determines the potential of the surrounding sprites and influences the movement of the avatar, slightly improving the victory rate of the algorithms in a similar set [Chu et al., 2015a]. An extension combines the KB Fast-Evo MCTS with Dijkstra's pathfinding algorithm to take advantage of the knowledge gain while guiding the agent to their targets in an efficient manner [Chu et al., 2015b]. The strengths and limitations of the KB Fast-Evo MCTS are analysed in detail in [van Eeden, 2015], which also includes and proposes further modifications to improve the performance of the algorithm. This knowledge heuristic has not been investigated further since the work described above or applied to other controllers, but I believe it has potential.

Search space navigation Additionally, Park and Kim [2015] propose the use of Influence Maps (IM) to assist the navigation of an MCTS algorithm through the search space more efficiently. Even when the agent cannot find rewards, the IM will guide it to potential ones. It uses the concept of *curiosity* to motivate the agent and resolve the goodness of the objects if there is no information about them. This goodness is determined with the forward model by learning the outcome of the interaction with each of the elements of the game. The information gained is used to create the IM to be at the disposal of the algorithm (Fig. 2.5). This solution tackles the problem of sparse rewards present in some games and, on average, can boost the resulting score.

The motivation instigating the study of alternative heuristics to winning and maximising the score comes from overcoming the limitations in the scarcity of rewards given by the baseline approaches. Still, they are ultimately created with the target of victory in mind. The following section looks at the existence of a variety of behaviours from the players based on their motivations. These motivations may ultimately involve winning, but it is not necessarily their primary goal or their goal at all, so why is the focus of GVGP agents primarily on achieving victory?

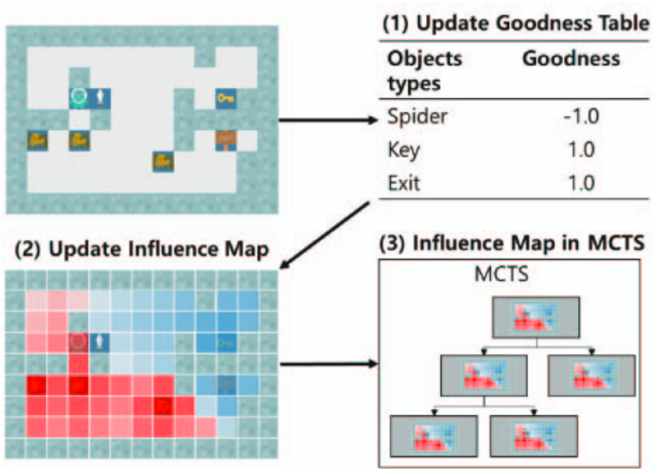


Figure 2.5: Overview of the use of an Influence Map; from [Park and Kim, 2015].

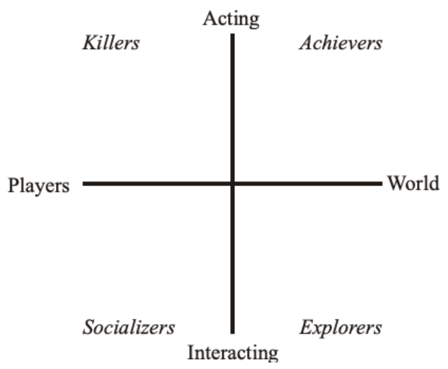
The generality of the agents has evident potential and, although there is a large and active community of researchers working on improving the general algorithms and creating new ones, there is a lack of research looking for applications for them. My work explores filling that void by looking at the possible integration of GVGP agents into the game development process or as an alternative AI approach within it. I believe that the path that eventually would take to finding these potential applications starts with eliciting differentiated behaviours for the GVGP agents by providing them with objectives beyond winning.

2.4 Gameplay Behaviour

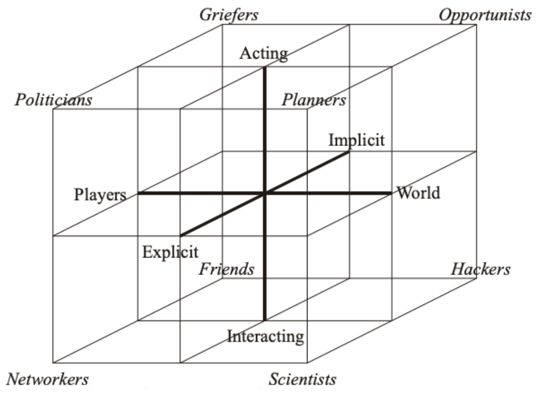
The concept of *persona* is used in Software Development to identify end-users and assist in the design of the product [Cooper, 1999]. It is also applicable to game development by considering that players present different ways to play a game based on their motivations, which leads to distinct behaviours and the existence of recognisable *player-types*.

2.4.1 Player-types

Bartle [1996] identifies the existence of four type of players in Multi-User Dungeon (MUD) games, based on their interests: *achievers*, *explorers*, *socialisers*, and *killers*. Players are differentiated regarding the elements they direct their attention to (the environment or other players) and how they act and interact towards them (Fig. 2.6a). Achievers focus on game-related goals that they give to themselves, while explorers try to find as much as possible about the virtual world. Socialisers communicate and interact with other players, and killers prioritise their imposition upon other players using elements of the game to cause an effect on them. Players generally have a primary style, but the classification is not a strict division. There may be a cross over between the areas, and players may change their style based on their mood or motivation at the time of playing. This information can be applied to the design of the game and balance the existence of the different play styles by putting strategies in place to encourage or discourage each of them. Bartle [2005] revises the initial theory due to some limitations: Firstly, it does not explain how or why players may change over time. Secondly, it is possible to identify sub-types in every player-type that the theory does not address. These issues are resolved by including a third dimension that distinguishes between planned actions to achieve an ultimate goal (explicit) and those that are done unconsciously (implicit). This additional dimension results in a total of 8 *player-types*, where each of the original types is divided into two different ones (Fig. 2.6b).



(a) Original (2D)



(b) Reviewed (3D)

Figure 2.6: Bartle's *player-types*; from [Bartle, 2005]

In any case, the focus of the player type theory is to explain why people play Massive Multiplayer Online (MMO) games, and it is not supposed (or expected) to be applied to other types of games [Bartle, 2012]. Therefore, the target audience is MMO designers to design and create their virtual worlds.

Bartle's player-types are well-defined, so if I used MMOs for my research, I could take each of the player-types as a reference to design and implement heuristics that represent similar behaviours for the agents. However, MMOs are complex, which is a barrier to use as a starting point for a new line of research. Furthermore, my research focuses on generality, so I look at designing an approach that can apply to different games. The way to validate this approach is to design and implement heuristics that can be used in different games. Using and defining a method for MMOs would limit this vision. Therefore, I decided to use this theory merely as an inspiration to recognise that I may assume that winning and achieving a high score are not the only motivations of the players. Because I do not use MMOs, I cannot simply take these player-types and assume they apply to every game, define them, and use them for testing. As a result, I first need to identify a list of different behaviours that can be found (or not) in one or more games. Then, I use this list to identify and implement general heuristics that I can use in games supported by the same framework. Finally, in each of the games I use to validate my approach, I choose and apply only the heuristics that make sense for each particular game to generate the final behaviours of the agents. My goal is to define and provide the tools to designers and game developers to determine the behaviours and heuristics that would apply to their games.

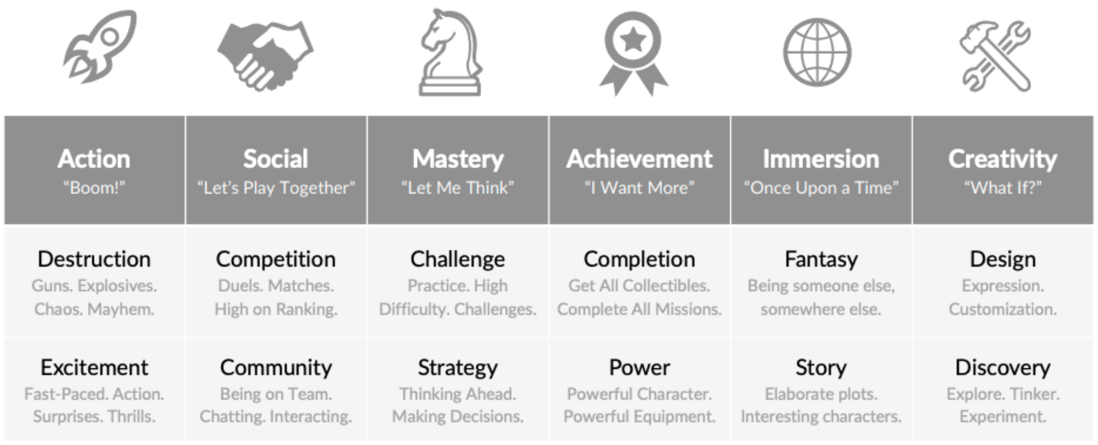
More recently, Yee [2016] has introduced a player motivation profile resulting from survey data from more than $250,000$ players. The model results in a total of 12 player motivations based on how players focus their interactions with the environment, game, or other players: *destruction*, *excitement*, *competition*, *community*, *challenge*, *strategy*, *completion*, *power*, *fantasy*, *story*, *design*, and *discovery*. These motivations can be paired to distinguish between six high-level groups, presented in Fig. 2.7a: *action*, *social*, *mastery*, *achievement*, *immersion*, and *creativity*. Similarly, at an even higher level of granularity, these can be grouped in 3 motivation clusters that facilitates the interpretation of the profiles [Yee, 2019a]: *action-social*, *mastery-achievement*, and *immersion-creativity*. Each of the motivation factors is a spectrum. The games from the data set in the top 20% and bottom 20% of each ranking are analysed in an attempt to address the blind spot or *negative space* common across most motivation taxonomies [Yee, 2019b]. These games exemplify the preferences of players with extreme motivations, so they are used to identify those preferences and the anchors on the extremes of the spectrum. I include the fantasy spectrum in Fig. 2.7b as an example. The profile generalises to several types of games, and the games themselves may show peaks in some of the classifications or do not present them at all.

Independently of the type of game under consideration, it is safe to assume that the players' motivation influences their goals, affecting their gameplay and reactions. I do not apply these models but take them as a revelation to look beyond the objectives of winning and achieving a high score to identify goals that can be used as heuristics for agents in different games. I define this list of possible goals in Section 5.3.

The following section looks into how authors have been able to classify and group distinct players' behaviours and motivations based on the metrics resulting from their gameplay.

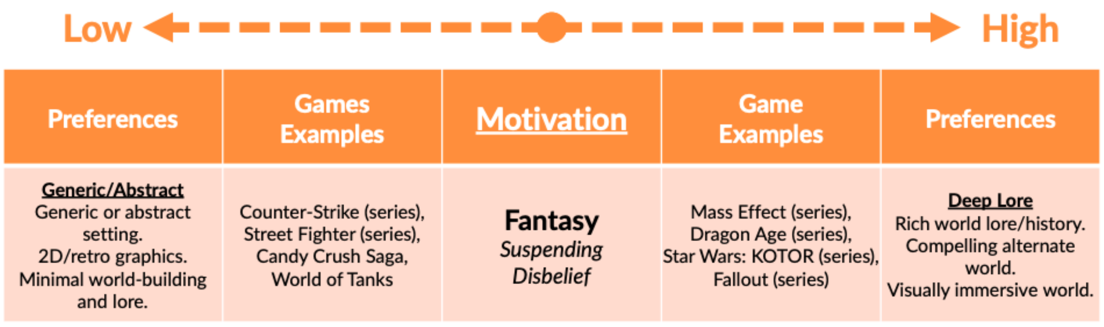
2.4.2 Play-personas and gameplay metrics

Players display distinct behaviours that are also dependant on the game's characteristics. These behaviours can be identified by recording metrics of their gameplay. Tychsen and Canossa [2008] make a distinction between character-bound metrics, associated with the Player Character (PC), and event metrics, related to every element unbound to the PCs, like NPCs or changes in the environment. PC metrics are classified in navigation, interaction, narrative, and interface metrics. Different types of analysis of these metrics are possible based on the requirements, so it is necessary to allow various levels of data aggregation: *play mode*, referring to the behaviour of the player regarding a few metrics, *play-style*, which combines multiple play modes, and *play-persona*, that emerges when the player uses play-styles consistently and represents different models of how players



Action "Boom!"	Social "Let's Play Together"	Mastery "Let Me Think"	Achievement "I Want More"	Immersion "Once Upon a Time"	Creativity "What If?"
Destruction Guns. Explosives. Chaos. Mayhem.	**Competition** Duels. Matches. High on Ranking.	**Challenge** Practice. High Difficulty. Challenges.	**Completion** Get All Collectibles. Complete All Missions.	**Fantasy** Being someone else, somewhere else.	**Design** Expression. Customization.
Excitement Fast-Paced. Action. Surprises. Thrills.	**Community** Being on Team. Chatting. Interacting.	**Strategy** Thinking Ahead. Making Decisions.	**Power** Powerful Character. Powerful Equipment.	**Story** Elaborate plots. Interesting characters.	**Discovery** Explore. Tinker. Experiment.

(a) Motivation factors grouped by pairs to distinguish between 6 high-level groups: *action*, *social*, *mastery*, *achievement*, *immersion*, and *creativity*.



Low ← → High

Preferences	Games Examples	Motivation	Game Examples	Preferences
Generic/Abstract Generic or abstract setting. 2D/retro graphics. Minimal world-building and lore.	Counter-Strike (series), Street Fighter (series), Candy Crush Saga, World of Tanks	**Fantasy** *Suspending Disbelief*	Mass Effect (series), Dragon Age (series), Star Wars: KOTOR (series), Fallout (series)	**Deep Lore** Rich world lore/history. Compelling alternate world. Visually immersive world.

(b) Detailed example: *fantasy* spectrum. Each motivation factor can be considered a spectrum, and games are identified in their extremes (low fantasy, high fantasy) to exemplify the preferences of players with extreme motivations. As an example of high fantasy, we encounter games like *Mass Effect* [BioWare, 2007], with an immersive world, rich history, and lore. On the other side of the spectrum, we find games like *Counter-Strike* [Valve, 2000], where the settings are generic, and the world-building is minimal.

Figure 2.7: Gamer Motivational Model; from [Yee, 2019b]

interact with the game. As an example, they define metrics particular to *Hitman: Blood Money* [IO Interactive, 2006] and use these levels of aggregations to describe patterns of gameplay corresponding with *personas hypotheses* that designers identify in the early stages of the game design. The authors conclude that metrics can provide precise information about the behaviour of the players, so this method can be used as a tool to develop and test a variety of gameplay styles. Alternatively, Ferguson et al. [2020] propose to cluster different play-styles based on visual information instead of gameplay data, as there may be cases where metrics are not available or accessible.

Play-personas are not limited to motivation profiles; they are both theoretical models, called *metaphor*, and data-driven representations of the players, called *lens* (Fig. 2.8) [Canossa and Drachen, 2009]. In other words, they represent identifiable behaviours available to the players given the characteristics and composition of the game. The designers can identify the play-personas, given the possibilities of the game rules, and verify their hypotheses using gameplay metrics and categorising the emerging behaviours by similarity.

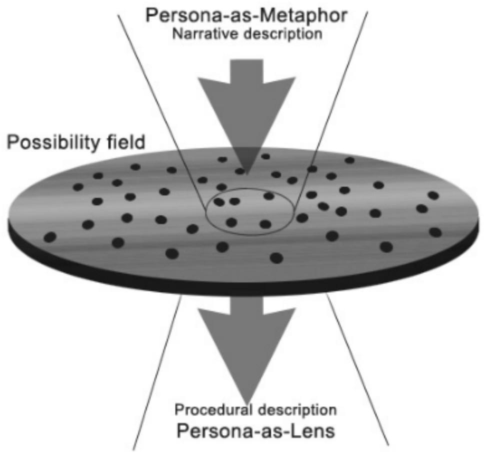


Figure 2.8: Play-persona as metaphor and lens; from [Canossa and Drachen, 2009]: *"The black dots on the possibility field plane represent game mechanics; thanks to the a-priori description of the persona-as-metaphor a certain subset is individuated. This persona hypothesis can be checked against metric data gathered from players and inform the creation of persona-as-lens."*

Drachen et al. [2009] use high-level metrics and unsupervised ML to identify four different types of players in *Tomb Raider: Underworld (TRU)* [Crystal Dynamics, 2008]. The data obtained from the gameplay of several players comprises the total number of deaths, the causes (by opponent, environment, or failing), completion time, and help-on-demand information. The type of player clusters identified are *veterans*, *solvers*, *pacifists*, and *runners*. Veterans are the group with the most well-performing players. Solvers focus on solving the puzzles of the game and do not usually ask for puzzle hints of answers, die quite often (mainly related to failing), and it takes them a long time to solve the game. Pacifists die mainly from active components, and both their completion time and asking for help is below average. Finally, runners complete the game quickly and die often, mainly caused by opponents and the environment. The existence of these differentiated behaviours proves that the players take advantage of the range of mechanics of the game and its possibilities rather than using a monolithic strategy. Moreover, being able to identify they exist can provide useful information for the designers and the development of the game. Sifa et al. [2013] go a step further and, instead of limiting their study to a unique picture of the behaviour overall in the game, they examine the evolution of player behaviour across the main levels of TRU.

Melhart et al. [2019] bring together player motivation and play-personas by studying the use of the gameplay data to predict the motivations of the players in *Tom Clancy's The Division* [Massive Entertainment, 2016]. They use ML to model different psychological constructs related to motivation, reaching a high accuracy percentage in unseen players. They conclude that player motivation can be predicted using gameplay features.

This section shows how it is possible to identify the motivations and behaviours of the players by looking at their gameplay results, which extends the theoretical work introduced in the previous section. Similarly, it is an inspiration to my research. The research presented focuses on human players, but the idea of using metrics to identify or understand how the game is played should also apply to agents that take the place of a player. Although I do not use ML or classification models, I look into the gameplay results of the agents to identify different behaviours in each game.

The following section looks at work that has also been inspired by the existence of distinct behaviours during gameplay to define game-playing agents that represent them, called *procedural personas*.

2.4.3 Procedural personas

The existence of archetypes in games have inspired the creation of artificial *personas* to model the decision making of players [Holmgård et al., 2014b]. A proof-of-concept theory-based method trains a series of Q-learning agents (a particular approach in Reinforcement Learning described in [Watkins and Dayan, 1992]), so their reward function simulates the utility function of different players. They present a dungeon themed puzzle game called *Minidungeon*, identify different playing styles, and train the agents by providing rewards related to each of them: a *baseline player* focuses on reaching the exit, a *runner* on minimising the moves, a *survivalist* on minimising the risk and not being killed, a *monster killer* on killing the monsters, and a *treasure collector* on collecting the treasures of the level. Independently of their style, all the agents are rewarded for finishing the game by arriving at the exit. The resulting behaviour of the agents depends on the level used for their training. Fig. 2.9 shows an example of the resulting heatmaps of the agents in one of the levels. The overall statistics show different play styles aligned with their description: the baseline and the runner show similar behaviours to the survivalist: reaching the exit, but the latter does not die in any of the executions. In contrast, the runner finishes the game by visiting the least number of tiles. Similarly, the collector gathers more treasures than any other agent, and the monster killer kills a high percentage of the monsters while at the same time collects the most significant number of potions to be able to do so.

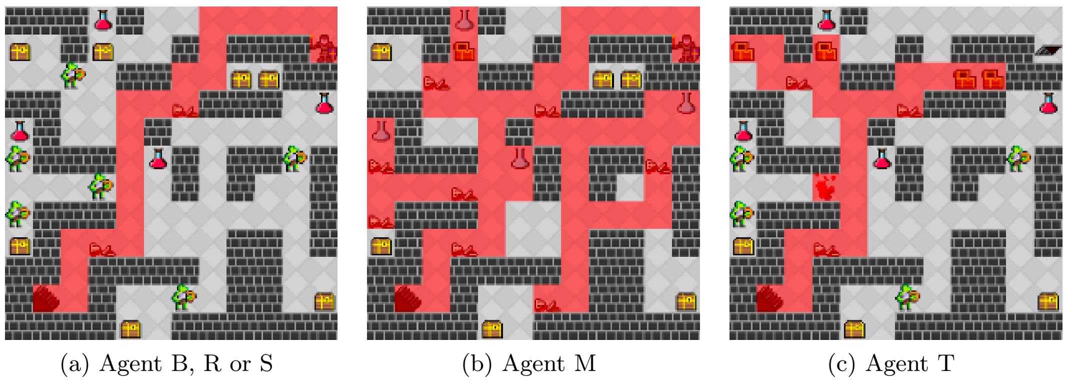

(a) Agent B, R or S (b) Agent M (c) Agent T

Figure 2.9: Example of heatmaps of the agents in one of the levels of *Minidungeon*; from [Holmgård et al., 2014b]. The nomenclature assigned to the agents is as follow: B for the *baseline* player, R for the *runner*, S for the *survivalist*, M for the *monster killer* and T for the *treasure collector*.

Additionally, Holmgård et al. [2014b] collect play traces from human players to compare them with the decisions taken by the agents in similar situations of the game. A *random* controller is also included in this study. This analysis looks for the fitting representation of each player by replaying the games and registering the persona that relates to the actions taken by the human. The results indicate that all the agents except *random* are suitable approximations of some player, the treasure collector dominating the data. The authors conclude that it should be possible to achieve a high-level abstraction of human decision making.

Next, Holmgård et al. [2014a] build over the work described above to overcome some limitations inherent to the use of Q-learning algorithms. They also introduce the concept of *procedural persona* to describe evolved game-playing agents that represent archetypal play-styles. The alternative method uses evolvable perceptrons to represent each persona as, although structured differently, they achieve a similar decision-making style to the first agents. Five personas with similar objectives to the ones described for the original experiment are defined, but the baseline player is named *exit* instead. To assess the performance of the *procedural personas*, they are compared with the Q-learning agents and a series of baseline agents in experiments similar to [Holmgård et al., 2014b]. The baseline agents defined are one random and a set of five hand-crafted agents, each of them created to achieve the primary goals related to each persona by following the shortest path to them. The results show that the *procedural personas*, which can generalise to unseen levels, are a better solution at matching human players and optimising the rewards to achieve the objectives.

Lastly, Holmgård et al. [2018] describe a final method to define and generate *procedural personas* for a reviewed version of the dungeon game called *Minidungeons 2*, which is detailed in [Holmgård et al., 2015]. The simulation of the behaviour in these personas is created by evolving a mathematical formula to replace the Upper Confidence Bound (UCB1) equation of Monte Carlo Tree-Search (MCTS) [Browne et al., 2012]. They identify four primary objectives, leading to four differentiated behaviours: *runner*, *monster killer*, *treasure collector*, and *completionist.* UCB1 is adapted to each of the primary objectives related to those behaviours: reaching the exit, killing the enemies, collecting items, and consuming any game object that it is possible to either collect or kill. Similarly to the previous experiments, the agents are also given a secondary goal: always finishing the game by arriving at the exit. They study the performance and behaviour of each of these *procedural personas* in several levels of the game. They observe clear distinction in the way they interact with the environment, leading to play-styles and performance based on their own goals. They notice how the agents are sensitive to the patterns of the levels. As a result of these observations, they propose using their *procedural personas* to create artificial play-traces and evaluate the levels of a game, arguing that the results can serve as feedback to the human designers.

Minidungeons 2 has a total of 11 levels, with a variety of characteristics and interactive objects. Analysing the personas' play-through across these levels shows that each persona leads to different significant correlations between metrics related to their properties and utility. As an example, *runner* obtains a significant correlation between total computation time and shortest path between the entrance and exit. The two levels with the most and least differences in metrics and between personas (Fig. 2.10) are further analysed to study how the elements' layout and disposition affect the diversity of behaviour and gameplay across the different personas. The authors also analyse which maps would be preferred by each persona based on the resulting metrics and conclude that these evaluations can be executed quickly, being a feasible method to take part in an iterative design process. Mugrai et al. [2019] apply the strategy of procedural personas for automated testing to Match-3 games.

Similarly, Ariyurek et al. [2021] extend the portrayal of the personas in RL to the characterisation of a dynamic model that addresses the static aspect of the personas defined in [Holmgård et al., 2014b], given by their bound to the utility function. These new goal-based personas, called *developing personas*, are constituted by a linked sequence of goals instead of a unique utility function, allowing a progression of behaviours during gameplay. It grants the designers a higher level of control and variations, as they can choose the criteria that fulfil each goal and continues the sequence, like in the example shown in Fig. 2.11.

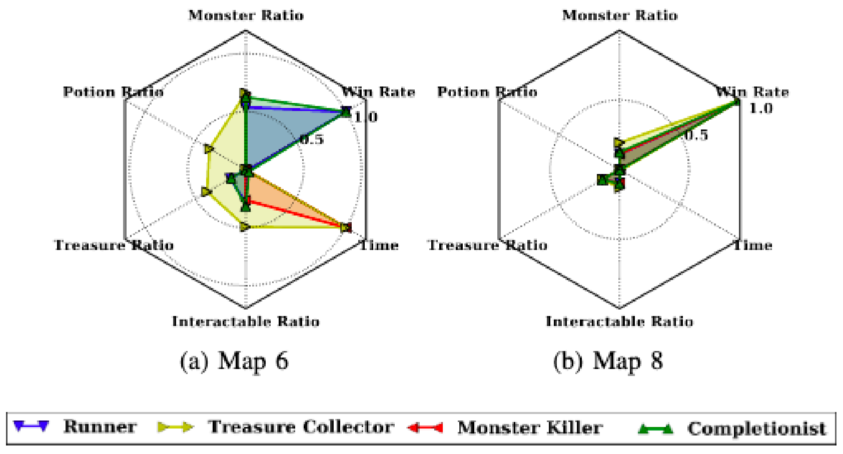


Figure 2.10: Metrics comparison between the *procedural personas* for the levels with the most and least differences between them; from [Holmgård et al., 2018].

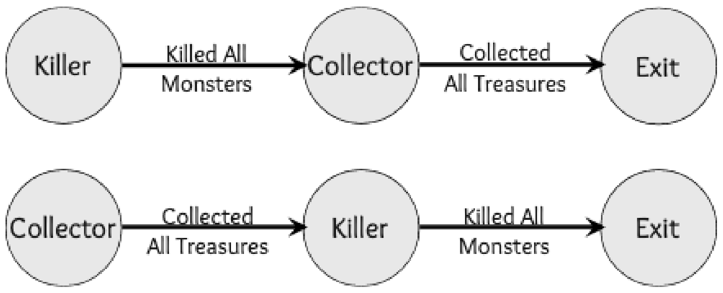


Figure 2.11: *Developing personas* example; from [Ariyurek et al., 2021]. The behaviour of the *persona* changes when a goal is reached, allowing a progression of behaviours during gameplay.

My work is inspired by [Holmgård et al., 2018], but it aims to have a more general and portable approach, capable of being applied to several different games without having to design specific types, or utility functions, to fit the game under consideration. I believe that extending the idea to use general agents, developed with general goals that apply to several different games, can provide significant advantages. A *team* of pre-defined types with general purposes and approaches can provide the designer with the opportunity to choose agents to fit different properties of the game. The following section looks into current approaches related to the use of automated testing in the games industry and academia.

2.5 Video Game Development and Testing

Game development is an interactive process, so a game evolves and goes through continuous modifications. Even minor updates can have an impact on the game and cause unexpected problems. Thus, when including changes to the game or integrating new levels, it should go through a testing process to ensure the status is aligned with the expectations. Different areas are covered to ensure the robustness of the game, both technically and related to the players' experience. *Player Experience* (PX) is a research field that focuses on studying the perception and responses of people when playing a game [Bernhaupt, 2015], and it is related to Games User Research (GUR) [Drachen et al., 2018].

I overlook GUR and focus on the technical domain that ensures that no bugs break the playability, influence the gameplay, or affect any other element or characteristic of the game as an artefact. *Play-testing* is used for this purpose, and refers to playing a game, or a portion of it, to identify potential issues in the game. It is usually carry out by going through a series of tasks to make sure all of them fulfill the expectations. This process of actively looking for issues in the game can also be called *bug finding*. This Quality Assurance (QA) of the games is usually carried out by members of the development team or testers. When these tests require playing the game manually, QA is time-consuming and prone to neglect issues. An alternative is adopting automated solutions capable of identifying problems and assisting with repetitive tasks required during the QA process. The purpose of automated testing is to facilitate the QA by using processes that are executed systematically and are capable of identifying issues in the game.

My work proposes a new automated approach that makes use of GVGP agents. Play-testing involves taking the player's role and is commonly manually carried out by humans. I argue that it should also be possible to use game-playing agents to assist in part of this process because these agents also play the game and, given the right set-up, should be able to encounter bugs during their play-through. Some authors are also looking into this idea. In this section, I present existing automated testing solutions in the industry and academia. It is a very complex problem, so there is a wide variety of approaches aimed at tackling it.

2.5.1 Automated testing in the industry

Some public examples of automated testing in commercial games are the following.

Some automated tests, like the ones applied in the *Call of Duty* [Activision, 2003] games, focus on validating the builds and checking if they pass or fail after every modification made to the game [van Valburg, 2018]. They also complement the manual QA with additional nightly performance tests to ensure the game executes on a specific budget. These performance tests measure the frames and the load of the CPU and GPU in different game locations. The tools used to obtain low-level performance tests and track execution traces to optimise the programs (including games) are called *profilers*. They are commonly used in the games industry to make sure the game executes within the resources budget.

Automated tests were also integrated into the development of *Sea of Thieves* [Rare, 2018]. The motivation came from the complexity of the open-world and the short length of the release cycles to include new features addressing the feedback of the players. Masella [2019] explains the different types of tests used, oriented to testing the robustness of gameplay features after the changes, being the main ones: unit tests, integration tests, and actor tests, which are gameplay tests but at a lower level. Incorporating these tests addresses the necessity of manually testing repetitive tasks, reducing the number of manual tests and time needed to verify the build. However, these tests do not align with iterative development as they require updates to adapt to the changes.

Other automated tests include automated play-through functionality and adaptability. In the point-and-click adventure game *SoBlonde: Back to the Island* [Wizarbox, 2010], they integrated testing as part of the game using dynamic tests capable of adapting to changes during the development [Marlin, 2011]. They created an engine and series of scripts embedded in the game as modules, one per scene, that can listen to the events generated and execute actions. In addition, they implemented a *solver* that searches different states to find a way to play the game to the end; in practice, this solver can be considered a brute-force game-playing agent. This solution resolves the need for manual testing and the risk of corrupting the code when modified, but it requires time and resources to integrate it as part of the game.

Moreover, Daedalic Entertainment, the publisher of adventure games like *Deponia* [Daedalic Entertainment, 2012] and *Anna's Quest* [Krams Design, 2015], uses ICARUS, a generic *"framework for autonomous video game playing, testing, and bug reporting"* [Pfau et al., 2017]. The system focuses on adventure games, completes a play-through in an amount comparable to a human play-tester, and can detect bugs and track performance measurements in real-time. It can identify the particular circumstance of the game where, for example, there is a peak in RAM usage or a drop in frame rates, while at the same time, supports the report of general bugs in the game like crashes and blockers.

Generally, there is much secrecy in the video game industry and the strategies followed by the studios during the development of the games, so it is challenging to assess precisely how frequent automated testing is or its extent. However, it is likely that many approaches, particularly those that involve executing automated play-through of the game, are game-dependant. Therefore, it may be expected that they are heavily hand-crafted and tailored towards the game, requiring regular updates. My work proposes using GVGP agents to assist in the testing process as a solution that allows automated gameplay and can adapt to different circumstances providing a more general approach.

2.5.2 Game-playing agents and automated play-testing

Automated testing and AI-assisted game design in academia is a recurring topic, and various approaches exist to tackle it in different areas [Zarembo, 2019, Albaghajati and Ahmed, 2020]. This section covers further research and methods proposed related to the use of game-playing agents for automated testing and assisting the design of the game to different extents that I consider relevant to my work. There are examples in multiple genres of automation using heavily hand-crafted techniques that create agents to play the game and fulfil tasks automatically [Iftikhar et al., 2015, Schatten et al., 2017]. However, I do not focus on these as I am more interested in studying the application in this area of more advanced game-playing techniques, like the ones introduced in Section 2.2.

The existence of diverse behaviours in the game is present in some of the methods proposed for automated testing. For example, the techniques and motivations of the players to build decks in digital card games can be taken as a reference to design a heuristic and balance a playing deck in *Hearthstones* [Blizzard Entertainment, 2014, García-Sánchez et al., 2018]. This procedure allows to build competitive decks and automatically test the design or addition of new cards without requiring to do it manually.

In the scope of dealing with player behaviour but related to game-playing agents instead of deck building, I have previously introduced Holmgård et al. [2018]'s work. They propose using *procedural personas* to evaluate the levels of a game and integrate them as part of their iterative design process.

Other approaches look into human-like automated testing by using Deep Learning to train the agents from data from human players' play-throughs [Gudmundsson et al., 2018]. Alternatively, some authors look at creating agents specific to play-testing instead of resembling players. Zhao et al. [2020] propose building play-testing agents following different gameplay objectives relevant to the designers instead of simulating players' behaviours. They present two case studies that build these agents based on the factors the designer requires to test in the games. Ariyurek et al. [2019] use and compare two strategies (synthetic and human-like) to generate goals related to play-testing and compare the performance of each of the agents in terms of finding bugs in the game.

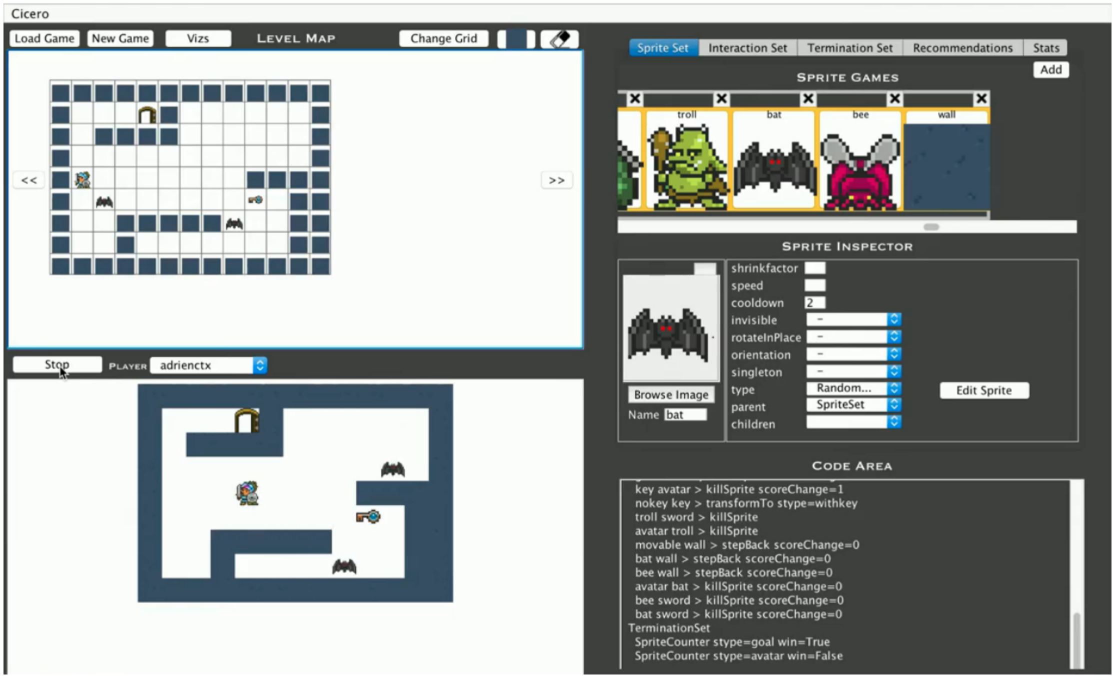
Although I am inspired by the existence of diverse behaviours and gameplays, I do not look at human-like approaches. My research does not consider that play-testing requires impersonating humans to the point that agents need to behave as they do. Instead, I look at providing enough diversity to elicit various interactions and actions identifiable with different types of play or tasks. I carry out all this work using general heuristics.

Next, I analyse some methods that link the use of general agents and testing.

Browne and Maire [2010] used GGP to test the validity of automatically generated board games. Their work presents *Ludi*, a framework capable of creating combinatorial games by mutating the rules of existing ones. In this system, general agents play the games generated to evaluate and decide if they are good enough. The agents do not have specific knowledge about the games, but their generality allows them to follow the rules when they are well-formed and play them so that the system can assess their quality. The ultimate goal of Ludi was to generate new compelling and publishable games, and it succeeded. *Yavalath* is a strategic game with an innovative rule (winning by making four-in-a-row but losing by making three-in-a-row) that has been the first computer-generated game to be commercially published. Although GGP is out of the scope of my research, this work motivates the idea of flexibility and adaptability of automated testing using general agents to extend it to GVGP. These types of agents are transparent to the rules, so they should be adaptable to changes on the game or the levels, applying not only to Procedural Content Generation (PCG) but also to the game development process.

In the scope of GVGP and planning algorithms, Nielsen et al. [2015] presents the Relative Algorithm Performance Profile (RAPP) approach to estimate the quality of a game based on the performance of various general agents. They compare the performance between known algorithms in a range of hand-designed, mutated, and random generated VGDL games. Their premise claimed that a game that has a high skill differentiation is likely to be a good one. Despite the results backing their hypothesis, I argue that complexity does not necessarily infer quality and, although this approach can be useful in the context of PCG, on its own, it cannot provide much information about a game.

In addition, the Computationally Intelligent Collaborative EnviROnment (*CICERO*) is a general-purpose AI-assisted tool for 2D tile-based game design built on top of the GVGAI Framework [Machado et al., 2017a]. The system assists in the creation and development of Video Game Description Language (VGDL) games. It provides a mechanism to add sprites and edit the rules of the game. It also includes a *mechanics recommender* that suggests certain content based on the existing one. The most relevant feature of the system is automated testing, which provides game rule statistics in real-time and a visualisation of the level. It allows playing the game manually or automatically with one of the general agents available in the framework. In their examples, they use OLETS. Running the game with a GVGP agent provides heatmaps of the play-through and the NPCs and information about the different rules that constitute the game. It shows a list with these rules and includes stats for the resulting interactions related to each, highlighting those not used during gameplay. Machado et al. [2017b] expand Cicero to incorporate *SeekWhence*, a gameplay recording feature that provides forward and backwards navigation of the session to analyse the sequence of events. Fig. 2.12 shows a screenshot of the tool. The limitation of Cicero is that it is very oriented to VGDL and the GVGAI framework, impeding its application to a broader range of games. Moreover, the system does not take advantage of most of the inherent information that can be extracted from the play-through, despite being possible to use any of the general algorithms available in the framework for evaluation. Data obtained from the agents' gameplay can provide richer information to the users. Lastly, the ultimate goal of these agents is winning, so the diversity and coverage are limited, and it is expected that their behaviour is considerably alike.



Player: adrienctx

Name	bat
shrinkfactor	
speed	
cooldown	2
invisible	-
rotateInPlace	-
orientation	-
singleton	-
type	Random...
parent	SpriteSet
children	

```
  key avatar > killSprite scoreChange=1
  nokey key > transformTo stype=withkey
  troll sword > killSprite
  avatar troll > killSprite
  movable wall > stepBack scoreChange=0
  bat wall > stepBack scoreChange=0
  bee wall > stepBack scoreChange=0
  avatar bat > killSprite scoreChange=0
  bee sword > killSprite scoreChange=0
  bat sword > killSprite scoreChange=0
TerminationSet
  SpriteCounter stype=goal win=True
  SpriteCounter stype=avatar win=False
```

Figure 2.12: Screenshot of *Cicero*. It shows the OLETS agent (*adrienctx*) playing a game to test its design; from [Machado, 2018].

Furthermore, aside from Cicero and including non-general premises, the use of agents to assist in the design and generation of levels is a common practice, usually referred to as *co-creativity* [Yannakakis et al., 2014]. Some methods focus on the playability of the level [Shaker et al., 2013]. At the same time, others go a step further and try to make sure that the navigation between different points of the level is possible and include an estimation of its difficulty [Hoyt et al., 2019].

There are a lot of different existing methods looking into automated play-testing. Most of these methods have something in common: they define and create game-playing agents. As a result, I have introduced and discussed a diverse range of AI solutions specific for playing a game, i.e. taking the position of the human player. In contrast, the following section gives a quick overview of the existing AI techniques used within the games, being integrated into the artefact and taking a role in its environment. I particularly highlight the applications of AI as Non-Player Characters (NPCs).

2.5.3 AI techniques within games with a focus on NPCs

NPCs are those characters in a game that the player does not control, and that can be enemies, allies, or offer a neutral behaviour towards the player. The behaviour they present is linked to their objective in the game and the interactions expected from them. They can attack or run away from the player, assist by accompanying them, provide hints, dialogue, or simply move and be part of the background of the game. In any case, NPCs are controlled by the game and can be considered Artificial Intelligence (AI). However, their implementation and, therefore, the characteristics of the algorithm used in each case vary. The solutions can be simple scripts or behaviour selection algorithms like Finite-State Machines (FSM), Behaviour Trees (BT), Utility Systems, Goal-Oriented Action Planners (GOAP) or Hierarchical Task Networks (HTN) [Rabin, 2013, Chapter 4]. In any case, they are usually hand-designed to give specific steps and goals based on the details of the game's design and the purpose of the NPC.

Search algorithms are extensively used in commercial games as path planning techniques, but their application for the development of the decision-making of NPCs is rare. However, the usage of search approaches seems to be growing [Rabin, 2015, Chapter 22]. An example is the successful application of MCTS in strategic games like *Total War: Rome II* [Creative Assembly, 2013, Rabin, 2015, Chapter 25]. However, their heuristics are still heavily tailored toward the game [Thompson, 2018]. The use of GVGP agents with heuristics that do not have specific information about the game has not yet found its way into commercial applications. My work also includes a first attempt to integrate GVGP agents with different goals as NPCs.

2.6 Summary and Conclusion

This chapter provides an overview of the areas related to my research and covers the topics that have served as an inspiration for my work.

First, it introduces a brief historical overview of how games have assisted in developing and improving AI solutions because they have been a crucial benchmark for their development. My work looks into taking advantage of the multitude of existing game-playing agents that have come up over the years to integrate them as part of the game development process and assist in their evaluation. Specifically, I focus on General Video

Game Playing, so this concept is also presented. Various frameworks are used for GVGP research, but not all of them suit the needs of my research, so I list them and summarise their critical information. My research focuses on tree-search and evolutionary algorithms, which are planning algorithms and have access to a forward model to simulate future states. Furthermore, my experimental work is the first step in a large vision, so it is essential to prove that the approach is generalisable to different games to validate its potential. The GVGAI Framework covers all my needs because it supports planning algorithms, has many 2D arcade single-player games available, and provides a diversity of sample controllers that can be used and modified. Likewise, the games are developed in VGDL, a very simple language that allows easily customising the existing games or creating new ones. The following chapter includes details about this framework and its characteristics, as well as the list of games I use in each of the experiments.

Most of the current GVGP solutions focus on winning. However, I believe that the path that leads to finding potential applications of these agents in the game development process starts with eliciting different behaviours. Therefore, I introduce and discuss a few general heuristics that already exist in the literature with a goal beyond winning: exploration, knowledge gain, and search-space navigation. All these have served as an inspiration for my work and are referenced later in the thesis. The idea of providing different behaviours to the agents is inspired by the existence of a diversity of gameplay behaviours in human players. This chapter also goes through literature related to this area. It introduces Bartle's *player-types*, a well-defined theory that identifies the existence of differentiated type of players in MMOs based on their interests. If I had carried out my work exclusively in MMOs, I could have applied this theory to my research and base my team of agents on these player-types. However, given the complexity of these games and my goal of defining an approach generalisable to multiple games, I take this work simply as an inspiration and proof that winning is not only the only motivation of the players. The Gamer Motivational Model presented also supports this statement, and it influences the definition of some of the goals listed in my vision in Section 5.3. There is existing work also inspired by the existence of different types of players. My research is heavily inspired by Holmgård's *Procedural Personas*, which I present and detail in this chapter. However, my goal is to define a more general and portable approach capable of being applied to multiple games without having to design specific utility functions or heuristics for each. Furthermore, I look at the behaviour of the agents as the results of their gameplay, instead of necessarily resembling human players or being based on their traces, and I refer to them as *behaviour-types*. This way of identifying the behaviours is inspired by current research that defines *play-personas* and looks into identifying the motivations of the players by looking at their gameplay metrics. This chapter also includes details about this research, which is focused on human players. However, I consider that using metrics to identify or understand how the game is played should also apply to agents that take the place of a player.

Lastly, this chapter provides an overview of the current techniques used for automated testing, both in the industry and academia. It clarifies the scope of my research within testing and focuses on current approaches that use game-playing agents. Most of the methods are heavily game-dependent, so they require regular updates given by the iterative development process of games. My work proposes using GVGP agents to assist in the testing process as a solution that allows automated gameplay and can adapt to different circumstances and changes. My long-term vision contemplates integrating GVGP agents not only to assist in the development process but within the game itself as another AI solution at the disposal of the developers. Specifically, I visualise GVGP agents being used as NPCs. Therefore, this chapter also covers existing AI techniques used for this purpose as they are relevant for the case study presented in Chapter 8, which is an exploratory experiment driven by this potential integration.

In conclusion, this chapter familiarises the reader with the areas of influence of my research and clarifies the motivation behind my work when it is in perspective with the existing one. The following chapter has a more specific focus on my research and provides detailed information about the tools that I use: the GVGAI Framework (Section 3.1), the controllers that I adapt in my experiments (Section 3.2), the evolutionary algorithm that I employ to generate the *team* of agents (Section 3.3), and the VGDL games I used in each of the experiments (Section 3.4).

Chapter 3

Tools

This chapter presents and describes the framework, algorithms, and games that have been employed in my work. Their use in the experiments is detailed in the corresponding chapter.

3.1 GVGAI Framework

I carry out my research in the General Video Game Artificial Intelligence (GVGAI) Framework, which is an open-source tool that facilitates the research in General Video Game Playing (GVGP), introduced in Section 2.3. The framework allows executing game-playing agents in different games as they are independent of the specification of the rules. In this section, I include details about the language the games are written in; the communication between the framework, the game, and the agent; as well as the use of the framework in several competitions over the years resulting in means available for research.

3.1.1 The Video Game Description Language (VGDL)

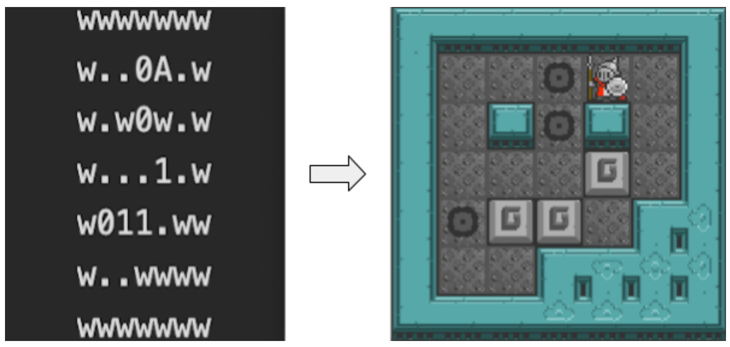
Games supported by the GVGAI framework are written in the Video Game Description Language (VGDL), which allows the implementation of 2D tile-based games. The origin of VGDL comes from the challenge of creating a clear, human-readable, and unambiguous language to develop video games, having a representation easy to parse and extend. The purpose of the language is to use it for GVGP research and for the automatic generation of content and game design [Ebner et al., 2013]. The project was inspired by previous work on description and high-level scripting languages for games, addressing their limitations and targeting the application of the language specifically to video games. Schaul [2013] created *PyVGDL*, a simple and high-level description language that allows the implementation of 2D tile-based video games. *PyVGDL* was originally supported by a system written in *Python*, and its subsequently port to Java resulted in VGDL and the GVGAI Framework [Perez-Liebana et al., 2015]. One of the ground-breaking approaches of the GVGAI Framework compared to previous solutions is the complete abstraction of the game and its elements, which are hidden to the agent. Each element of the game is represented by an entity (sprite), and collisions between them trigger the rules. The player (represented by an avatar) is not aware of the definition of the games, rules, or game over conditions. The engine exposes an API that allows querying the game status (game tick, score, and winning condition), the state of the avatar (position, velocity, orientation, etc.), the actions available, and a complete view of the sprites of the game via *Observations*. Given the generality of the framework, the sprites are not individually identified, but it is possible to recognise their category. They can be avatar (player) sprites; sprites generated from the avatar; sprites representing Non-Playable Characters (NPCs); sprites generated from the NPCs; resources; immovable objects; or spawn points. Collisions between sprites are called interactions, and those that happen between the avatar or the sprites generated by it, cause events.

A game in VGDL is defined by two components, provided by two files written in plain text: the definition of the game and the definition of the level. Four sets of rules define the game: *SpriteSet*, outlining the entities that form the game; *LevelMapping*, linking the sprites with their character representation on the level; *InteractionSet*, describing the results of the collisions between sprites; *TerminationSet*, specifying the end of the game and its winning or losing conditions. By default, games have a maximum number of game ticks (2000), after which the game is over. It is possible to set a different time limit in the game definition and override that number. Details about the VGDL language and the GVGAI framework are given in [Perez-Liebana et al., 2019c, Chapter 2].

To show the simplicity and readability of VGDL, I include the implementation of *Sokoban*, a puzzle game where the goal is to push all the boxes into holes. Listing 3.1 represents the implementation of the game in VGDL, and Fig. 3.1 defines a level and shows its corresponding representation at execution time. *SpriteSet* defines a total of 5 entities that constitute the game: *floor*, *hole*, *avatar*, *box*, and *wall*. The type *MovingAvatar* indicates that the actions available to the player are moving up, down, right, and left. The floor, hole, and walls are static objects (*Immovable*) with a definite position, while the box is an object the player can interact with (*Passive*). *LevelMapping* defines the symbols that will represent each of the sprites in the level file. The *floor* is a background sprite, so it will appear in every tile of the game when not entirely covered by the entity image. The *InteractionSet* defines the rules of the game and the result of the interaction between each pair of elements: 1) The wall blocks the way of the avatar, so every action moving towards it ends up with it in the same initial position (*stepBack*). 2) An interaction between the avatar and a box moves the latter (*bounceForward*), 3) unless it is in the direction of the wall, in which case the box stays in its initial position (*undoAll*). 4) A box interacting with a hole *destroys* the box sprite and increases the score by 1. Lastly, the *TerminationSet* indicates a unique end of game-winning condition, which corresponds to the number of sprites of the type box reaching 0. Therefore, the player wins when no boxes are left in the game because they have been destroyed by being pushed into holes.

```
BasicGame key_handler=Pulse square_size=40
    SpriteSet
        floor  > Immovable img=newset/floor2
        hole   > Immovable img=oryx/cspell4
        avatar > MovingAvatar img=oryx/knight1
        box    > Passive img=newset/block1
        wall   > Immovable img=oryx/wall3 autotiling=True
    LevelMapping
        0 > floor hole
        1 > floor box
        w > floor wall
        A > floor avatar
        . > floor
    InteractionSet
        avatar wall  > stepBack
        box avatar   > bounceForward
        box wall box > undoAll
        box hole     > killSprite scoreChange=1
    TerminationSet
        SpriteCounter stype=box limit=0 win=True
```

Listing 3.1: VGDL implementation of *Sokoban* provided by the GVGAI Framework: to win, the goal is to push all the boxes into the holes.



```
wwwwwww
w..0A.w
w.w0w.w
w...1.w
w011.ww
w..wwww
wwwwwww
```

Figure 3.1: Definition of a level of *Sokoban* and its representation during execution.

3.1.2 The GVGAI Framework and Competition

The GVGAI Framework serves as a benchmark for both learning and planning algorithms. The generality and abstraction of the framework and games encourage the development of general controllers that can play any of the games supported by it. The framework has been used to run several competitions since 2014, receiving the submission of hundreds of entries [Perez-Liebana et al., 2019b].

The planning engine receives the definition of the VGDL games and levels and presents the controller (agent) with a Java object interface containing the state of the game. Listing 3.2 shows the API interface provided to single-player controllers, which must inherit from *AbstractPlayer*. Every game tick, the game makes a call to the *act* method, so the agent must implement it to determine the next action carried out in the game. The method *result* is non-mandatory, and it is called at the end of the game.

```java
public <ClassName>(StateObservation , ElapsedCpuTimer);

Types.ACTIONS act(StateObservation, ElapsedCpuTimer);

void result(StateObservation, ElapsedCpuTimer);
```

Listing 3.2: API interface for single-player games, from [Perez-Liebana et al., 2019c]

The framework also provides the agent with a forward model that can be used to simulate future states and to assist in deciding the action that should be taken next. As stated in the previous section, the agent is not informed about the nature of the game or its rules but can query certain information about its state to make a decision: history of interactions and events, IDs and category of the existent sprites, score, health points, game tick, winning or losing status, etc. The actions available depend on the rules and the state of the game, but they are within: moving up, down, right, or left; shoot or use an object (set on the game rules); do nothing. In the competition, the controller is given a budget of time of 40ms to return an action. If an action is returned after this time, the *do nothing* action is carried out no matter the choice, and if no action is returned in 50ms, the controller is disqualified. Fig.3.2 presents a high-level flow chart and the sequence diagram of the execution of the game. Although it is not indicated, the framework also allows setting the interface of the game to be controlled by a human player instead.

The first GVGAI Competition focused on single-player planning algorithms [Perez-Liebana et al., 2015]. The single-player planning track provides two training sets to the contestants to evaluate the strength of their algorithms and improve them. The controllers submitted to the competition play in a final set of games and levels, unknown beforehand, that determines the final ranking. The agent resulting with the best performance overall on the final set of games is considered the winner. The evaluation system

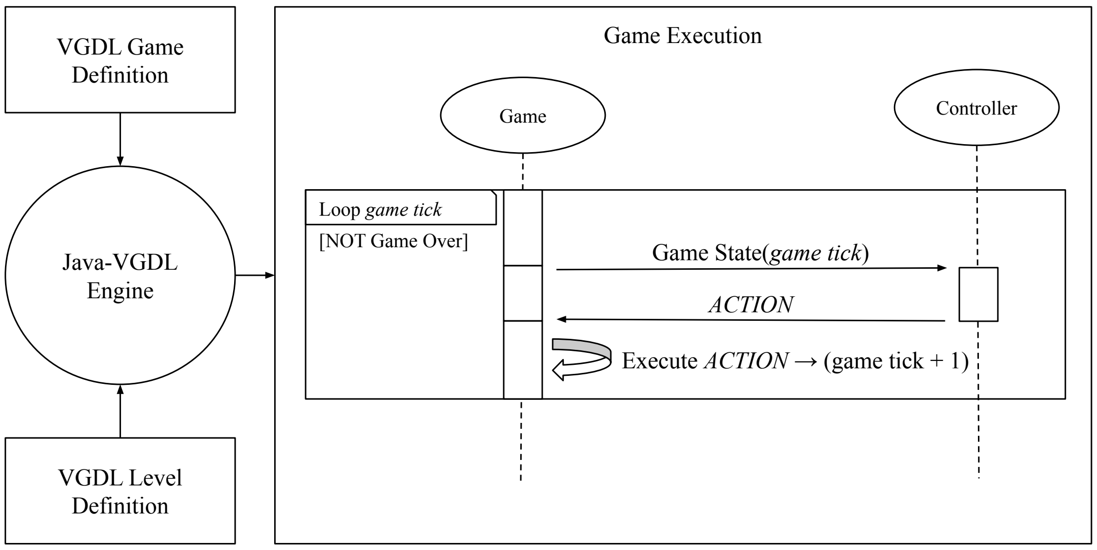


Figure 3.2: The GVGAI Framework: single-player planning track. The engine receives the VGDL definition of the game and level and executes it. For each game tick, the controller is provided with the game state to run its simulations and return the selected action. This interaction is repeated until the game is over. Inspired by [Perez-Liebana et al., 2019c, Chapter 2]

encourages the participants to build general algorithms to maximise a good result for any game instead of focusing on a specific one. The framework also supports two-player games [Gaina et al., 2016], where agents have also access to the state of the opponent and the actions taken by them. Listing 3.3 shows the API interface provided to two-player controllers, which makes use of *StateObservationMulti* and includes an *int* that determines the id of the player.

```
public <ClassName>(StateObservationMulti , ElapsedCpuTimer, int);

Types.ACTIONS act(StateObservationMulti, ElapsedCpuTimer);

void result(StateObservationMulti, ElapsedCpuTimer);
```

Listing 3.3: API interface for two-player games, from [Perez-Liebana et al., 2019c]

The 2-player planning track competition presents the controllers against each other in complex problems, both competitive and cooperative games, forcing them to adopt different play styles and techniques. Over the years, the GVGAI framework has been extended, and new tracks have been included in the competition to cover other novel areas of general AI. Outside the scope of my work, the framework also supports learning algorithms [Perez-Liebana et al., 2019c, Chapter 5], where no forward model is provided, and Procedural Content Generation (PCG) approaches, like the generation of game levels [Khalifa et al., 2016] and rules [Khalifa et al., 2017].

Perez-Liebana et al. [2019b] give an insight into the details of the framework, its application to several areas of research in general agents since its release, the algorithms implemented for it, and a complete list of the winners of the different tracks for each edition of the GVGAI Competition. The success of the competition has resulted in the existence of several controllers available in the framework. My work leverages the diversity of the approaches to use some of the available algorithms, detailed in the next section.

3.2 Agents

My work leverages the existent agents available in the GVGAI Framework. This section lists and describes the controllers provided by the framework that I use in my experiments. I use both tree-search (Section 2.2.1) and evolutionary algorithms (Section 2.2.2). Although I carry out some modifications to adapt the algorithms to my needs, their core does not change, so the following descriptions are relevant and applicable. Details about the modifications and differences with the sample agents provided by the framework are given in the corresponding chapters.

3.2.1 One Step Look Ahead (OSLA)

The OSLA is a simple tree-search algorithm (Section 2.2.1) that simulates only one state in the future. The controller checks the reward gained in the states reached by taking every available action and executes the one with the best rating (Fig. 3.3).

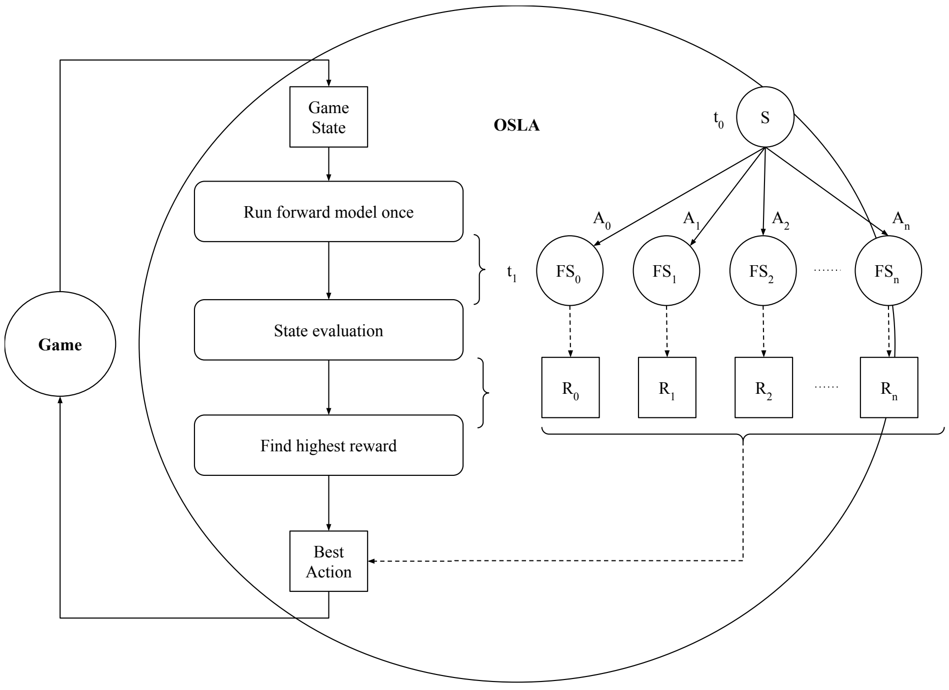


Figure 3.3: OSLA algorithm: receives the current game state (S) from the game, executes the forward model for each available action (A), resulting in a future game state (FS). Each state is evaluated by the heuristic to obtain the value of its reward (R). The action related to the highest reward is selected and returned to the game.

Algorithm 1 OSLA algorithm.
Nomenclature: $s \leftarrow$ game state; $s' \leftarrow$ simulated game state; $a \leftarrow$ action; $A(s) \leftarrow$ available actions; $R(s) \leftarrow$ reward at state s; $\Delta \leftarrow$ reward value.

```
1: procedure OSLA($s_0$)                                   ▷ current game state
2:     for $a \in A(s_0)$ do                ▷ for each available action at state $s_0$
3:         $s' \leftarrow ForwardModel(s, a)$                 ▷ run forward model once
4:         $\Delta \leftarrow R(s')$                                  ▷ state evaluation
5:     return Best action   ▷ return action resulting in state $s'$ with highest reward
```

The evaluation of the future state in the sample agent is done as follows: the heuristic returns a high positive reward if the resulting state of the game is winning; a high negative reward if it is losing; and the resulting score points if an end state is not reached. This controller has been available in the framework since the first GVGAI Competition [Perez-Liebana et al., 2015].

3.2.2 Monte Carlo Tree Search (MCTS)

MCTS is a tree-search algorithm (Section 2.2.1) that combines search and random sampling. It builds the tree incrementally and asymmetrically (Fig.3.4) and balances between the exploration and exploitation of its nodes. Exploration refers to visiting those nodes that have not been extensively analysed, and exploitation refers to exploring those nodes that look promising. Browne et al. [2012] created a survey detailing the basic MCTS algorithm (vanilla implementation) and the numerous derivations and enhancements included until then. This section focuses on the vanilla implementation and the description of its Open-Loop version (detailed below). I refer the reader to the survey for further information about the variants of the algorithm and its applications.

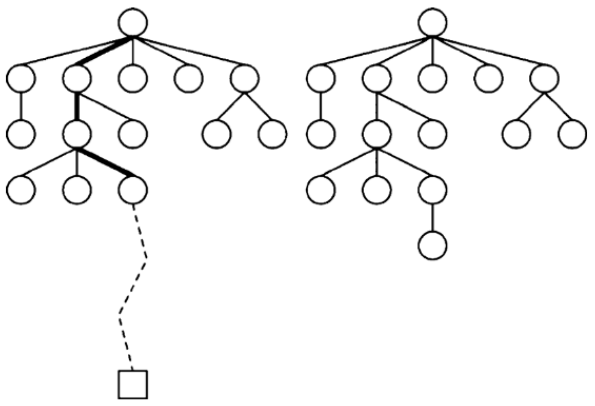


Figure 3.4: Basic MCTS algorithm process; from [Browne et al., 2012].

MCTS builds the game tree until it reaches a stop condition. More computational power leads to better performance. However, there are usually limitations related to the budget allocated for its computation in terms of time or memory. Four steps take part in each iteration of the algorithm until reaching this budget: *selection*, *expansion*, *simulation*, and *backpropagation* (Fig. 3.5).

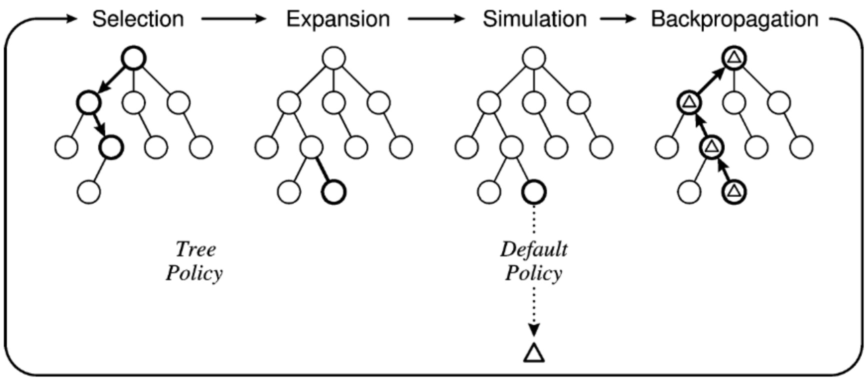


Figure 3.5: One iteration of the MCTS algorithm: *selection*, *expansion*, *simulation*, and *backpropagation* steps; from [Browne et al., 2012].

For the *selection* and *expansion* steps, the MCTS uses a *Tree Policy*, which takes care of navigating the tree and selecting or creating new nodes from the existent ones. The policy used for the child node selection is the *Upper Confidence Bound* (UCB1), defined at line 25 of Algorithm 2. The search tree is navigated starting from the root node and while non-terminal nodes are reached. The navigation continues until it finds the next expandable node. The terminal or expanded node is then *simulated.* According to Browne et al. [2012], simulation in the context of this algorithm is "*playing out the task to completion according to the default policy,*" i.e. the sequence of actions carried out from the node resulting from the selection and expansion steps. By default, the actions to use in the simulation phase are selected uniformly at random until an end-of-game state or certain pre-defined depth is reached [Perez-Liebana et al., 2019c, Chapter 3]. When the simulation ends, the state is evaluated to determine its reward. Finally, the statistics assigned to each of the nodes that were passed through in the previous steps are updated, applying that reward (*backpropagation*). When the computational budget breaks the loop, the algorithm processes the performance obtained by each available action and returns the one with the best final reward.

The **Open-Loop MCTS (OLMCTS)** introduced by Perez Liebana et al. [2015] is a version of the vanilla MCTS that uses an open-loop approach for the nodes of the tree. It is suitable for stochastic scenarios, where it is impossible to estimate how accurate the simulated states stored in the nodes are. In contrast to deterministic environments, an unknown probability takes part in the simulation. The solution given by open-loop approaches is saving the statistics in the nodes instead, as they represent the set of states navigated in the tree. OLMCTS requires to use the forward model every time a new action is chosen during *selection* and *expansion.*

The *sampleMCTS* algorithm provided by the GVGAI framework is a vanilla implementation of the OLMCTS. The heuristic used is as follows: high positive reward for winning end-of-game states; high negative reward for losing end-of-game states; and score of non-end-of-game states.

Algorithm 2 MCTS algorithm; from [Browne et al., 2012].
Nomenclature: $s \leftarrow$ state; $v \leftarrow$ *node*; $s_0 \leftarrow$ initial state; $a \leftarrow$ action; $v_0 \leftarrow$ root node; $v_t \leftarrow$ terminal node; $s(v) \leftarrow$ game state in node v; $\Delta \leftarrow$ reward; $N(v) \leftarrow$ times visited node v; $Q(v) \leftarrow$ total simulation reward of node v; $p \leftarrow$ player; $A(s) \leftarrow$ available actions at state s; $f(s, a) \leftarrow$ state reached after applying a to s; $C_p \leftarrow$ arbitrary constant

```
function MctsSearch($s_0$)
    create root node $v_0$ with state $s_0$
    while within computational budget do
        $v_t \leftarrow$ TreePolicy($v_0$)
        $\Delta \leftarrow$ DefaultPolicy($s(v_t)$)
        Backup($v_t, \Delta$)
    return $a$(BestChild($v_0, 0$))

function TreePolicy($v$)
    while $v$ is nonterminal do
        if $v$ not fully expanded then
            return Expand($v$)
        else
            $v \leftarrow$ BestChild($v, C_p$)
    return $v$

function Expand($v$)
    choose $a \in$ untried actions from $A(s(v))$
    add a new child $v'$ to $v$
        with $s(v') = f(s(v), a)$
        and $a(v') = a$
    return $v'$

function BestChild($v, c$)
    return $\arg\max_{v' \in children\ of\ v} \frac{Q(v')}{N(v')} + c\sqrt{\frac{2 ln N(v)}{N(v')}}$
function DefaultPolicy($s$)
    while $s$ is non-terminal do
        choose $a \in A(s)$ uniformly at random
        $s \leftarrow f(s, a)$
    return reward for state $s$

function Backup($v, \Delta$)
    while $v$ is not null do
        $N(v) \leftarrow N(v) + 1$
        $Q(v) \leftarrow Q(v) + \Delta(v, p)$
        $v \leftarrow$ parent of $v$
```

3.2.3 Open-Loop Expectimax Tree Search (OLETS)

OLETS is a tree-search algorithm (Section 2.2.1) implemented by *Adrien Couëtoux* [Perez-Liebana et al., 2015] and winner of two competition tracks: the first Single-player Competition and one of the Two-player Competitions [Perez-Liebana et al., 2019b]. The algorithm is based on the Hierarchical Open Loop Optimistic Planning (HOLOP) [Weinstein and Littman, 2012] with three key differences: 1) it is designed for finite action spaces; 2) makes use of the Open Loop Expectimax (OLE) method to evaluate the reward value; 3) does not use any roll-out. OLETS does not store the states in memory but generates a tree where the nodes represent the sequence of actions explored. It randomly chooses the actions to simulate until there are no more unexplored actions left in a node. At that moment, the OLE method gives a score to the branches, and the algorithm chooses to simulate the one with the best reward. The reward obtained for each node of the tree is accumulated. When the final state is reached or the controller is requested to end, the action with the corresponding highest value is executed. The pseudocode is included in Algorithm 3.

The heuristic used in the algorithm provided by the GVGAI framework depends on the status of the simulated state of the game and its score points as follows: a high value is added to the score in a winning state; a high value is subtracted from the score in a losing state (also dependant on the depth of the tree); the score without any modification is returned for non-end-of-game states.

3.2.4 Rolling Horizon Evolutionary Algorithm (RHEA)

RHEA is an evolutionary algorithm (Section 2.2.2) first introduced by Perez et al. [2013] as an alternative approach to the well-performed MCTS. The authors presented promising results when comparing its performance with MCTS in an experiment using the Physical Travelling Salesman Problem (PTSP), focused on navigation in real-time. The algorithm was later applied to video games and GVGP [Gaina et al., 2017].

In RHEA, a plan of action represents an individual. Therefore, each individual of the population contains the sequence of actions to carry out in the game. To determine the fitness of the individual, the current state of the game is simulated using the forward model. A heuristic function is used to obtain the reward of the final state reached by its sequence of actions. The simulation is stopped ahead of time if a terminal state is reached, and the fitness assigned to the individual is the reward obtained in that terminal state. In the vanilla implementation of RHEA, the population of individuals is initialised at random. In each cycle (Fig. 3.6), the population is evolved to generate new individuals for the next generation. This evolution is done by mutation (randomly updating actions in the sequence) and cross-over (combining existing individuals to create new ones). The new individuals are then evaluated to obtain their fitness as described above. The fitness is used to sort the individuals in the population and discard the worst ones before

Algorithm 3 OLETS algorithm; from [Perez-Liebana et al., 2015].
Nomenclature: $n \leftarrow$ node; $n_s(n) \leftarrow$ number of simulations passed through n; $n_e(n) \leftarrow$ number of simulations ended in n; $R_e(n) \leftarrow$ accumulated reward from simulations that ended in n; $C(n) \leftarrow$ set of children of n; $P(n) \leftarrow$ parent of n; $\tau_M(n) \leftarrow$ empirical average reward in n.

```
 1: procedure OLETS($s$,$T$)
 2:     $\tau \leftarrow root$                                   ▷ initialize the tree
 3:     while $elapsed\ time < T$ do
 4:         RunSimulation($\tau, s$)
 5:         return $action = \arg\max_{a \in C(root)} n_s(a)$
 6:
 7: procedure RunSimulation($\tau$,$s_0$)
 8:     $s \leftarrow s_0$                                       ▷ set initial state
 9:     $n \leftarrow root(\tau)$                                ▷ start by pointing at the root
10:     $Exit \leftarrow$ False
11:     while $\neg Final(s) \wedge \neg Exit$ do               ▷ navigating the tree
12:         if $n$ has unexplored actions then
13:             $a \leftarrow$ Random unexplored action
14:             $s \leftarrow ForwardModel(s, a)$
15:             $n \leftarrow NewNode(a, Score(s))$
16:             $Exit \leftarrow$ True
17:         else                                                 ▷ use node scoring to select a branch
18:             $a \leftarrow \arg\max_{a \in C(n)} \text{OLE}(n, a)$
19:             $n \leftarrow a$
20:             $s \leftarrow ForwardModel(s, a)$
21:     $n_e(n) \leftarrow n_e(n) + 1$
22:     $R_e(n) \leftarrow R_e(n) + Score(s)$
23:     while $\neg P(n) = \emptyset$ do                         ▷ update the tree
24:         $n_s(n) \leftarrow n_s(n) + 1$
25:         $\tau_M(n) \leftarrow (R_e(n)/n_s(n)) + ((1 - n_e(n))/n_s(n)) \max_{c \in C(n)} \tau_M(c)$
26:         $n \leftarrow P(n)$
27:
28: procedure OLE($n$,$a$)
29:     return $score = \tau_M(a) + \sqrt{ln(n_s(n))/n_s(a)}$
```

creating a new generation. The end of the algorithm's execution is given by a limit in terms of time, memory budget, or a maximum number of iterations. When the algorithm ends, the individual with the best fitness is selected. The action to be carried out next in the game corresponds with the first action in the plan of this best individual.

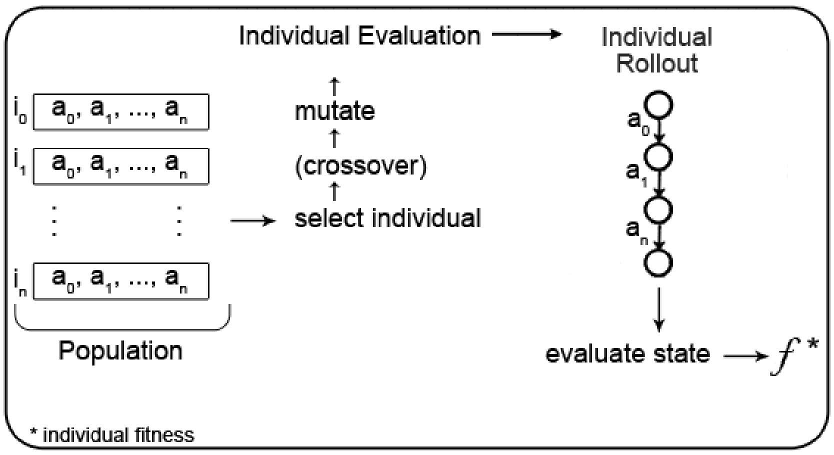


Figure 3.6: RHEA algorithm cycle; from [Perez-Liebana et al., 2019c, Chapter 3].

The parameters that are needed to be set to execute the algorithm are the following: elitism (E), which defines the number of individuals to be carried forward to the next generation without being changed; mutation rate (M), which determines the probability of a gene to be mutated; population size (P); individual length (L). Gaina et al. [2017] proves that the performance of the RHEA algorithm is highly affected by P and L when the rest of the parameters are fixed. A series of enhancements of RHEA are presented and discussed in [Perez-Liebana et al., 2019c, Chapter 3].

The *sampleRHEA* algorithm provided in the GVGAI framework has its parameters set to $P = 10$, $E = 1$, and $M = 1$. It produces as many sequences as possible in the time provided as budget, so the number of individuals is dynamic. Its heuristic rewards win states, heavily penalises losing states, and returns the difference of score.

3.2.5 Random Search (RS)

Random Search (RS) is a particular configuration of RHEA where $P = 24$ and $L = 20$ [Gaina et al., 2017]. Due to the time limit given by the budget, this algorithm only has time to initialise its population at random. Therefore, its decision is based on trying 24 different random action sequences and selecting the plan with the best reward. The first action found in this plan is carried out next in the game.

3.3 MAP-Elites

The Multi-dimensional Archive of Phenotypic Elites (MAP-Elites) is an illumination algorithm that can search in a very high dimensional space created by all possible designs to find the highest performance criterion for each combination of features [Mouret and

Clune, 2015]. An example would be finding the *fastest* robot for each combination of *height*, *weight*, and *energy consumption per meter* features. The algorithm *illuminates* the relationship between performance and dimensions of interest solutions and returns a set of high-performance and diverse solutions. It requires defining the following elements:

Genome/genotype x Candidate solution.

Search space All possible values of x.

Phenotype p_x Assessment of the characteristics of x.

Fitness function f_x Measures the performance to evaluate each candidate x.

Feature space N-dimensions of variation of interests defining the area of the MAP-Elites.

Feature/behaviour function b_x Defines the features that correspond to each x, determining the values that should be assigned to the N-dimensional vector of the map.

The relationship between *genotypes*, *phenotypes*, and the behaviour and performance of a candidate goes as follows:

$$x \rightarrow p_x \rightarrow b_x, f_x$$

It is possible that a direct encoding exists between a characteristic of the candidates and a phenotypical feature, but it is also viable that there is an indirect encoding, where a complex process maps the elements of the candidates with components of the phenotype [Mouret and Clune, 2015].

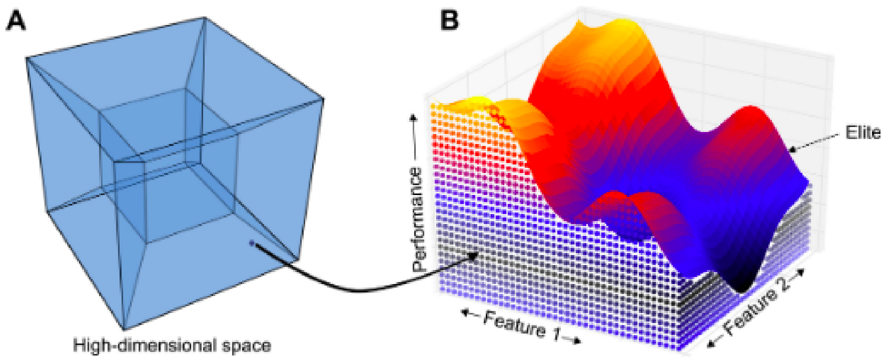


Figure 3.7: MAP-Elites graphic representation; from [Mouret and Clune, 2015].

The original version of the MAP-Elites algorithm, introduced by Mouret and Clune [2015] is presented in Algorithm 4. It starts with the creation of an N-dimensional map of elites and its initialisation by generating random candidate solutions. In each new iteration of the algorithm, until the stop condition is reached, one of the current solutions of the map (elite) is randomly chosen and evolved to create a new candidate solution. This evolution is done via mutation and/or crossover. The new candidate generated is then simulated to obtain its feature description, which defines its position on the map, and its performance. The new elite is automatically assigned to its correspondent cell if this is empty. If there is a current elite occupying it, the performances are compared. The new elite replaces the previous solution only if it has a better performance. When the algorithm finishes, the map contains the elites obtained with the resulting highest performance found for their corresponding cell.

Algorithm 4 Simple, default version of the MAP-Elites algorithm; from [Mouret and Clune, 2015].
Nomenclature: $X \leftarrow$ solutions (map of elites); $P \leftarrow$ solutions' performances; $x \leftarrow$ elite; $x' \leftarrow$ candidate solution; $b' \leftarrow$ feature descriptor of x'; $p' \leftarrow$ performance of x'.

```
1:  procedure MAP-Elites
2:      $(P \leftarrow \emptyset, X \leftarrow \emptyset)$        ▷ create an empty, N-dimensional map of elites
3:      for $iter = 1 \rightarrow I$ do        ▷ repeat for $I$ iterations
4:          if $iter < G$ then        ▷ initialise by generating G random solutions
5:              $x' \leftarrow random\_solution()$
6:          else        ▷ all subsequent solutions are generated from elites in the map
7:              $x \leftarrow random\_selection(X)$        ▷ randomly select an elite $x$ from $X$
8:              $x' \leftarrow random\_variation(x)$        ▷ create a randomly modified copy of $x$
9:          $b' \leftarrow feature\_descriptor(x')$        ▷ simulate $x'$ and record its $b'$
10:         $p' \leftarrow performance(x')$        ▷ record the performance $p'$ of $x'$
11:         if $P(b') = \emptyset$ or $P(b') < p'$ then
12:             $P(b') \leftarrow p'$
13:             $X(b') \leftarrow x'$
        return feature-performance map ($P$ and $X$)
```

3.3.1 Application in games

The simplicity of the MAP-Elites algorithm allows its application to different disciplines, including the field of games.

Khalifa et al. [2018] introduced a constrained version of the MAP-Elites to generate levels for bullet hell games. This constrained version combines MAP-Elites with a Feasible Infeasible 2 Population (FI2Pop) genetic algorithm. It emerged given the necessity of searching and identifying the optima in multiple sub-spaces, useful in the context of Procedural Content Generation (PCG). Each cell of the map contains two populations, representing two intentions: a) maximise its fitness (feasible population); b) satisfy a set of constraints (unfeasible population). Therefore, each chromosome is identified by the pair (cell, population). It can move between cells when its properties change and between populations based on the feasibility of its constraints. The dimensions of the MAP-Elites represent some aspects of the difficulty of the game. In the particular case of the bullet hell games considered, the authors distinguish between entropy (the amount of input required from the player), risk (the presence of bullets close to the player), and distribution (the amount of space the bullets occupy). Alvarez et al. [2019] applied a constrained MAP-Elites to assist a mixed-initiative design of levels in the Evolutionary Dungeon Designer (EDD) and named it Interactive Constrained MAP-Elites (IC MAP-Elites). This continuous version of the algorithm allows the users to influence the evolutionary algorithm that generates the levels in real-time and suggests customised and diverse solutions.

Another application of MAP-Elites in games is to influence gameplay. Fontaine et al. [2019] present the MAP-Elites with Sliding Boundaries (MESB), a modification of the MAP-Elites that recalculates the boundaries of the map every certain number of iterations. They use this algorithm to design and balance *Hearthstone* [Blizzard Entertainment, 2014] decks, discovering high-performance solutions with a diversity of strategies, given by the resulting illumination of the space. Previous work also includes a relationship between game-playing agents and the MAP-Elites algorithm: Balla et al. [2021] use the illuminated space resulting with MAP-Elites to examine the relationship between the parameters of a game and the behaviour of a well-performed agent. Bravi and Lucas [2021] study the behavioural space of the board game *Splendor* [André, 2014] by illuminating the search space with the hyper-parameters of a configurable play-testing agent. Last but not least, regarding agent generation, Canaan et al. [2019] use MAP-Elites to generate a pool of agents to play and perform in the card game *Hanabi* [Bauza, 2014].

I apply the MAP-Elites algorithm in the work described in Chapter 6, and the solution, out of the scope of the thesis, has also been adapted to achieve different play-styles while keeping a competitive level of play in a strategy game [Perez-Liebana et al., 2021].

3.4 Games

I have used different games throughout the experiments carried out in this work. All of them are implemented in VGDL and run in the GVGAI Framework. The games I use in my experiments have mostly been taken from the vast available collection provided by the GVGAI Framework. There are hundreds of games available, and they are very diverse, so it is possible to find games with different characteristics. For each of my experiments, I need to make a selection from these games based on what the work requires. Some of these games have been used without modifications, but others have been adapted to the needs of the experiment. A new game has also been created from scratch. This section includes details about the selection of games and the motivation for creating a new game.

3.4.1 Game selection for Chapter 4

In Chapter 4, the games are used as a benchmark to study and compare the performance of different algorithms. Therefore, the primary motivation behind the game selection for those experiments comes from having a diverse collection of games with various characteristics so the results can be generalised. Previous authors carrying out research in the GVGAI Framework have faced a similar problem, so I base my selection of games on such previous research. [Gaina et al., 2017] combined two classifications of games present in previous work and uniformly sampled the games to come up with a new subset constituted by a total of 20 games, split equally between deterministic and stochastic properties (Table 3.1).

Deterministic games		**Stochastic games**	
Bait	*Camel Race*	*Aliens*	*Butterflies*
Chase	*Escape*	*Chopper*	*Crossfire*
Hungry Birds	*Lemmings*	*Digdug*	*Infection*
Missile Command	*Modality*	*Intersection*	*Roguelike*
Plaque Attack	*Wait for Breakfast*	*Seaquest*	*Survive Zombies*

Table 3.1: List of the 20 games from the GVGAI framework used in the experiments in Chapter 4. This sub-set is based on the selection carried out by [Gaina et al., 2017]

These games have been used in the experiments without modifications, so I introduce and briefly describe them in this section. The timeout of each game, unless otherwise stated, is 2000. Details about their rules, elements and a screenshot of the levels are provided in Appendix A.

Aliens The player controls a ship, and the goal is to kill all the aliens before they reach the bottom of the screen. Details and a screenshot are given in Appendix A.1.

Bait Puzzle game where the goal is to collect a key and reach the exit. Details and a screenshot are given in Appendix A.2.

Butterflies The goal is to capture all the butterflies before the time runs out or all the cocoons transform into butterflies. Details and a screenshot are given in Appendix A.3.

Camel Race The objective is to be the first camel to reach the goal banner. Details and a screenshot are given in Appendix A.4.

Chase The goal is to kill all the white birds by colliding with them. Details and a screenshot are given in Appendix A.5.

Chopper The goal is to destroy the tanks while avoiding the missiles before the time runs out or all the satellites are destroyed. Details and a screenshot are given in Appendix A.6.

Crossfire The goal is to reach the door without being hit by the bombs. Details and a screenshot are given in Appendix A.7.

Digdug The player can break walls to achieve the goal: kill all the monsters and collect all the gems and gold pieces scattered around the level. Details and a screenshot are given in Appendix A.8.

Escape Puzzle game where the objective is to reach the cheese by pushing away and throwing into holes the boxes blocking the path. The timeout is 1000 game ticks. Details and a screenshot are given in Appendix A.9.

Hungry Birds The aim is to reach the goal and *exit* the maze before the health points, which are reduced every game tick, are entirely depleted. Details and a screenshot are given in Appendix A.10.

Infection The goal is to carry a virus and infect all the healthy people before the time runs out. Details and a screenshot are given in Appendix A.11.

Intersection The objective is to avoid the cars and reach the goal sign as many times as possible before the time runs out. The timeout is 1000 game ticks. Details and a screenshot are given in Appendix A.12.

Lemmings The goal is to help the lemmings to reach the door by breaking the walls. Details and a screenshot are given in Appendix A.13.

Missile Command The goal is to eliminate the missiles directed to the cities before they reach them. Details and a screenshot are given in Appendix A.14.

Modality The goal is to push the bush into the hole. The player can walk over two types of surfaces, but they can only cross from one to the other by a specific point in the map. Details and a screenshot are given in Appendix A.15.

Plaque Attack The goal is to destroy all the food by shooting at it before they damage the teeth. Details and a screenshot are given in Appendix A.16.

Roguelike The objective of the game is to reach the goal sign without being killed by the enemies. Locked doors block the path of the player, but they can be opened with keys. Details and a screenshot are given in Appendix A.17.

Seaquest The goal is to rescue divers by bringing them to the surface and stay alive until the time runs out. The timeout is 1000 game ticks. Details and a screenshot are given in Appendix A.18.

Survive Zombies The goal is to survive until the time runs out. The player has health points that lower every time they collide with a zombie. The timeout for this game is 1000 game ticks. Details and a screenshot are given in Appendix A.19.

Wait For Breakfast The goal is to go to the empty table and wait for the waiter to serve the food. The timeout for this game is 1000 game ticks. Details and a screenshot are given in Appendix A.20.

3.4.2 Game selection for Chapters 6 and 7

The experiments included in Chapters 6 and 7 require four games with different characteristics. One of them should have simple rules and be able to reach goals related to *winning* while trying to achieve a high score, *exploring* the level, and *interacting* with different elements of the game. The other three games should be more complex and allow, in addition, having goals related to *killing* enemies, *collecting* items, or both. I look into having at least a game where the player kills the enemies by hitting them and another one where they are killed by shooting at them instead. I am also interested on the end of game being triggered by different rules or achievements. The final selection of games is formed by *Butterflies*, *Digdug*, *Sheriff*, and *Zelda*.

The experiments require that the rules of the game fulfil some constraints given by the heuristics implemented and used by the agents. I have reviewed the VGDL implementation of the games selected and updated their taxonomy when necessary to ensure the experiments are carried out successfully. *Butterflies* fits the restrictions without any modifications. However, *Digdug*, *Sheriff*, and *Zelda* have required updates in their rules, elements, or sprites. I include the details about the reviewed versions of these three games. The timeout of each game, unless otherwise stated, is 2000.

Butterflies The goal is to capture all the butterflies before the time runs out or all the cocoons transform into butterflies. Details and a screenshot are given in Appendix A.3.

Digdug This personalised version of *Digdug* (A.8) addresses the rule issues that did not follow the constraints detailed in Chapter 6, and includes new elements and sprites. The game is formed by an **avatar** that can move up, down, right, and left and use a **shovel**; **monsters** that move randomly; **gems**; **gold**; and **breakable walls**. In contrast to the original game, it defines the **monsters** as NPCs, the **gem** and **gold** as resources, removes the boulder as an element, and introduces the **gold block** (G block), which becomes gold when hit with a shovel. Similarly to the original *Digdug*, the goal of the game is to collect all the resources and kill all the monsters before the time runs out. If the player is killed by a monster when colliding with it, the score decreases by 1 and the game ends. The **wall** and G blocks limit the path of the avatar, but the shovel can destroy them. The avatar can collect the resources (gem and gold) when colliding with them, but only collecting the gem increases the score by 1. The monsters, generated in a **monster spawner**, can be killed when hit with the shovel, increasing the score by 2. Fig 3.8 shows the screenshot of the level used in the experiments, which is similar to the one presented in A.8 but uses the G block instead of the gold element.

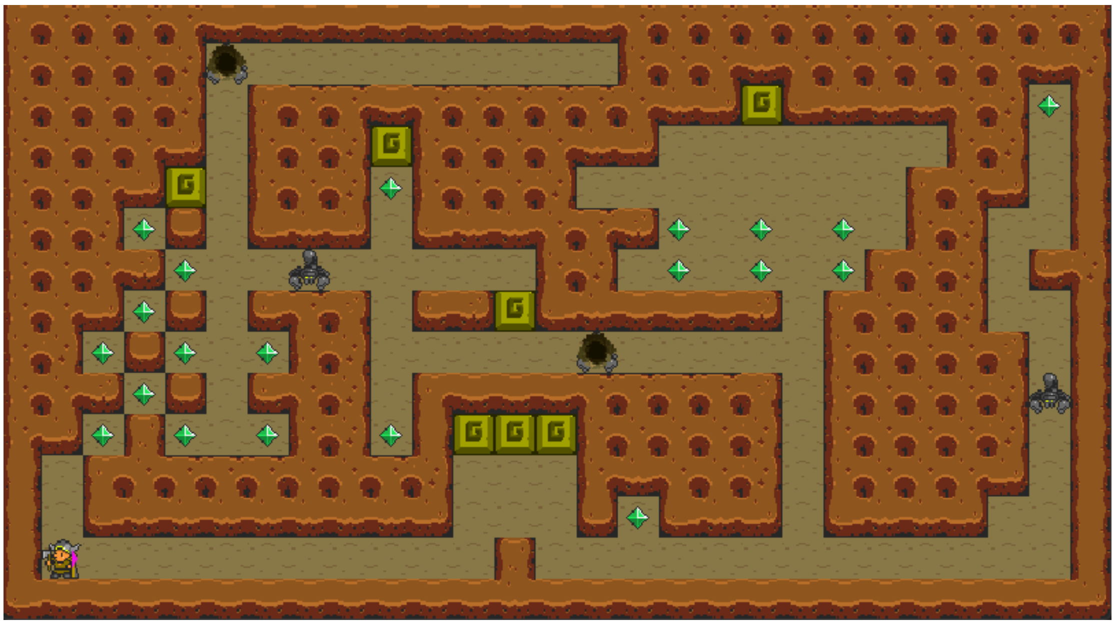


Figure 3.8: Screenshot of *Digdug* at time $t = 0$: 20 gems, 7 gold blocks, 2 monster spawners, 2 initial monsters (12 total possible), 267 breakable walls, 405 locations.

Sheriff This version of the game has similar rules than the original, but uses alternative sprites to the ones provided by the framework. It is constituted by an **avatar** that can move up, down, left, and right and shoot **red bullets**; **bandits** that are NPCs that can shoot **blue bullets** and move anti-clockwise inside the **prison**; **barrels** that are *Immovable* objects that breaks when hit by a bullet; and **walls** that define the limits of the map. The goal is to kill the bandits by shooting at them (+1 score change) and surviving until the time runs out. The player cannot access the prison but can shoot into it, and dies when hit by an enemy's bullet, which decreases the score by 1. The timeout for this game is 1000 game ticks, and a screenshot of the main level is included in Fig. 3.9.

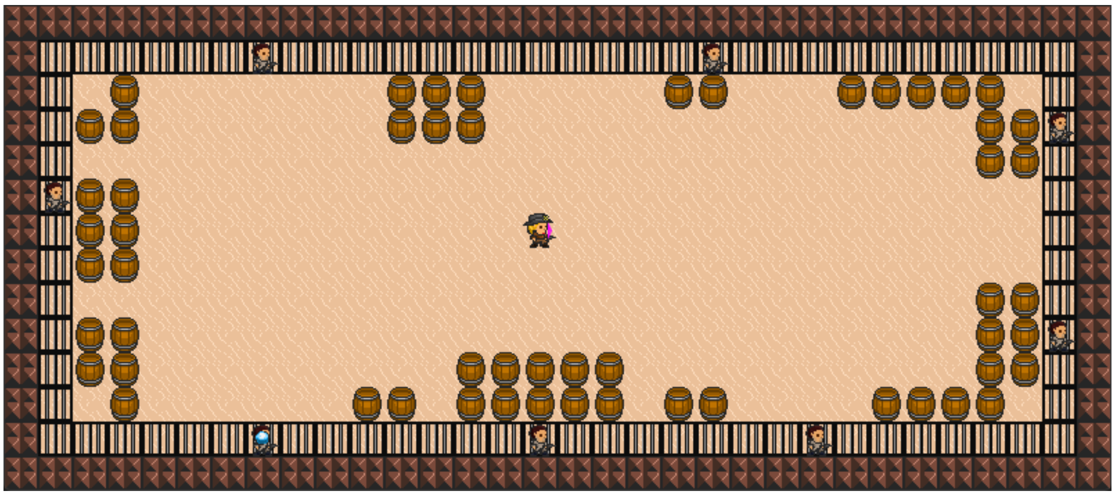


Figure 3.9: Screenshot of *Sheriff* at time $t = 0$: 55 barrels, 8 bandits, 80 blocks of prison, 88 walls, 280 locations.

Zelda The game and level used are personalised in relation to the one provided by the framework. It includes modifications of the rules in order to solve the issues found in the original game. The description included here refers to the updated version. The game is formed by an **avatar** that can move up, down, right, and left and use a sword; three types of **monsters** that are NPCs and move randomly; a **key** that is a resource, a **door** that is an immutable element the player can interact with after collecting the key; and **walls** that shape the limits of the map. The goal of the game is to collect the key and use it to open the door before the time runs out or the player gets killed by the monsters. The three different types of monsters are differentiated by the sprite assigned to them: slow (scorpion), normal (spider), and quick (bat). The player can kill the monsters by hitting them with the sword, which increases the score by 2 independently of the type of monster. The player can also be killed by the monsters when colliding with them, which reduces the score by 1 and ends the game. The player collects the key by colliding with it, increasing the score by 1. If the player collides with the door without the key, nothing happens, and the door blocks its path. Fig 3.10 shows the screenshot of the main level used.

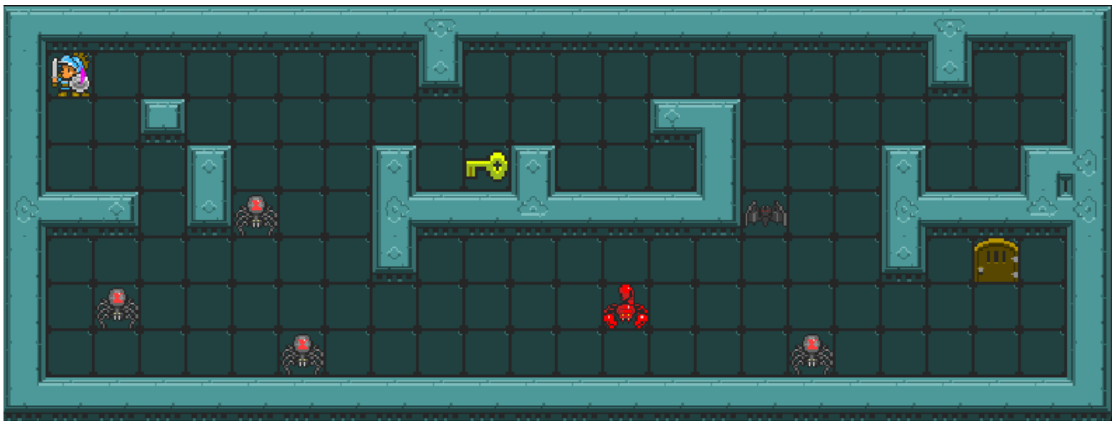


Figure 3.10: Screenshot of *Zelda* at $t = 0$: 1 key, 1 door, 6 monsters, 90 walls, 126 locations.

3.4.3 Original game for Chapter 8: *Skulls and Tombstones*

Skulls and Tombstones is an original game created from scratch for the case study presented in Chapter 8. Like every game in the GVGAI framework, it is implemented in VGDL and can be played by any of the algorithms supported by it. The case study looks at Player Experience constructs, so the game is expected to be played by humans. The games in the GVGAI Framework have not been implemented with human players in mind, so we decided to design and develop a game from scratch instead of using an existing one. Going through this process would allow us to iterate over the game to ensure it is easy to understand and provides enough feedback to the players during gameplay.

It is a 2-player competitive *collect the flag* type of game and consists of the following elements: **two avatars**, one assigned to each player; **trees**, shaping the limits of the map; and various **skulls** and **tombstones** scattered around. *Player 0* is assigned the colour blue and *Player 1* red, used to differentiate the tombstones acquired by each of them. Initially, the tombstones do not belong to any player, so they have no colour assigned. The players need to collect the skulls, one at a time, and bring them to a tomb to turn it into the corresponding colour. A player collects a skull or turns a tombstone by colliding with them. It is possible to turn a tombstone already owned by the other player. The score is updated every time a player acquires a tombstone and represents the number of tombstones owned by each, multiplied by 10. The game runs for 30 seconds (400 game ticks), and when the end of the game is reached, the player controlling the highest number of tombstones wins. In the case study, the game is used as a player-vs-NPC game.

The level (Fig. 3.11) is designed with the idea of distributing the elements around the map, so the layout is aligned with an exploratory behaviour.

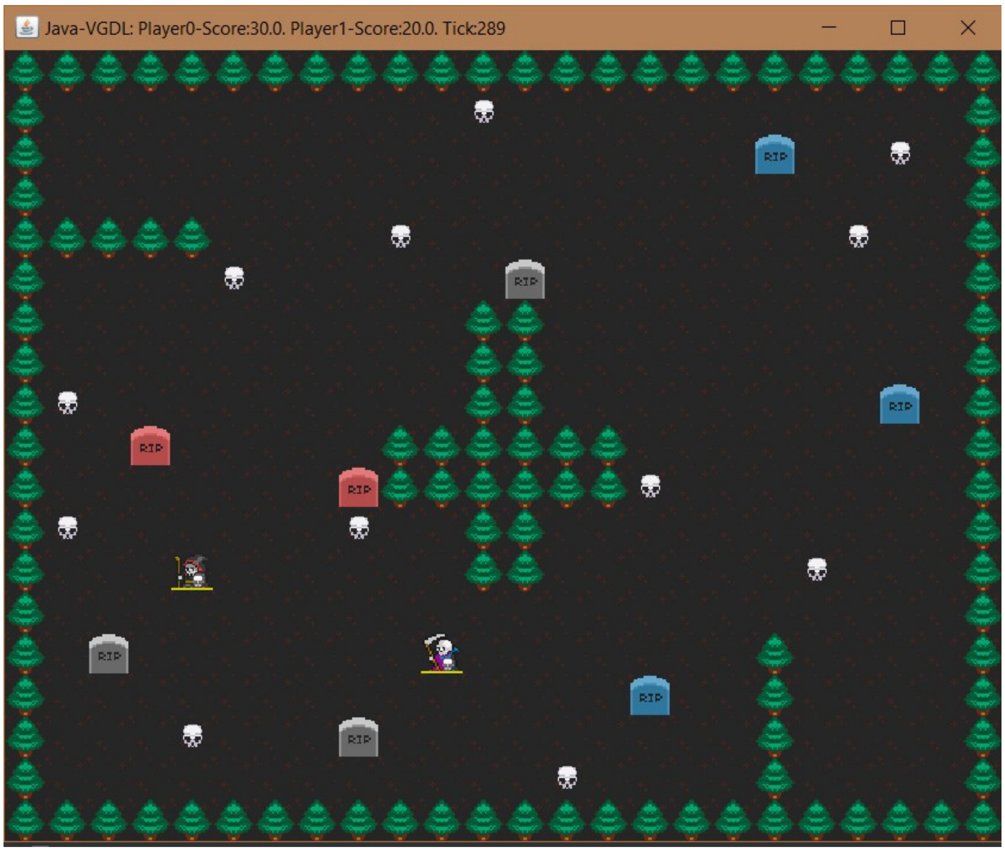


Figure 3.11: Screenshot of *Skulls and Tombstones* at $t = 289$, showing three tombs controlled by Player 0 (blue) and two tombs by Player 1 (red). There are 12 uncollected skulls, and both players are carrying one.

3.5 Summary

This chapter has introduced the tools pertinent to my research so the reader can be familiar with them. It has described the GVGAI Framework, where I carry out all my experiments. It has provided details about its characteristics, the language used to implement the games supported by it (VGDL), and the competitions in which the framework is used. It has also included an overview of the GVGP agents I use in my research: OSLA, MCTS, OLETS, RHEA, and RS; the MAP-Elites algorithm, which is relevant for Chapter 6, and the games used in each block of experiments. The information provided in this chapter is helpful in the following chapters, where my research is presented in detail.

Part II

Beyond Playing to Win

Chapter 4

Foundation: Heuristic Diversification in General Game-Playing

This chapter presents the foundation of the work. I introduce the term of *heuristic diversification*, define a series of game-playing goals, and implement the corresponding heuristics to apply to different controllers for comparison in terms of both performance and behaviour. The heuristics are general within the GVGAI Framework and can be used in any game supported by it. Most of the material presented has been published in the paper [Guerrero-Romero et al., 2017].

4.1 Introduction

Building game-playing agents can be a simple or a complicated task, depending on the approach followed. However, one aspect is common: the programmer typically knows how the game works. Therefore, concepts related to the game are usually included as heuristics, allowing the algorithm to explore the search space in a certain way. On the other hand, when building agents for General Video Game Playing (GVGP), introduced in Section 2.3, these heuristics are less clear, especially if the game being played is unknown a priori. In the particular case of the GVGAI Framework and Competition (Section 3.1), the possibilities of writing heuristics that can guide the search in all games (over 150 in total) are significantly reduced. After years of the GVGAI Competition, there has not been a single approach that has achieved more than 65% of victories across the different game sets, and, in most cases, winners achieve a win rate below 50% [Perez-Liebana et al., 2019c, Chapter 2]. These algorithms, and GVGP algorithms in general, are mainly focused on winning and maximising the score. However, as discussed in Section 2.4, these are not the only motivations driving the players and shaping their gameplay. Thus, why are the goals of these agents being limited to those two?

Procedural personas [Holmgård et al., 2014a, 2018] are the example of successfully applying the concept of existing differentiated archetypes to game-playing AI to provide agents with distinct behaviours. I argue that making use of several agents with different goals (and their pertinent heuristics) is a feasible approach to follow in GVGP. General agents are implemented with various methods and techniques, but their heuristics cannot apply specific information about the rules of the game or environment. However, as presented in Section 2.3.2, these heuristics can contain general goals unrelated to the fact of achieving a victory or increasing the score. Therefore, general agents can potentially be designed to elicit different behaviours to respond and adapt to the diverse situations presented in the games and accomplish particular tasks. I believe this line of research is promising and can enrich current approaches. Several applications can be given to these agents, being possible to extend the field in different ways. As a first step, I need to find a method to diversify the heuristics in GVGP and study the general agents when provided with goals beyond winning. The purpose is to understand how the agents respond and act in the game when driven by each of these goals. Therefore, the first question (RQ1) I look to answer in the research is: *Which general heuristics can be defined and implemented beyond the goal of winning the game, and how does each of these affect the performance and behaviour of existing GVGP agents when it is the only variation in the algorithm?*

I employ five General Video Game Playing (GVGP) algorithms to play a series of games in the GVGAI framework using four heuristics with differentiated objectives: 1) winning and maximising the score; 2) exploring the map; 3) discovering and interacting with the different elements of the game; 4) acquiring knowledge about the game and its rules. For each of these heuristics, I set performance criteria related to their objectives and run experiments to compare the strength of each algorithm. I am also interested in the agents' behaviour when provided by each of the heuristics, so I include an analysis and discussion about it. The use of the GVGAI Framework allows running the experiments in several games without having to adapt the controllers or heuristics for each of them, making it possible to extend the research to as many games as intended. The execution and actions taken by the controllers depend not only on the rules of the game but also on the characteristics of the level. As a result, in some games or levels, the agents may not be able to reach the specific goal. This limitation is expected because generality and adaptability are preferred.

4.2 Definitions

I define the concepts used in this chapter as follows:

Heuristic value Result of the evaluation of a game state, given in relative terms of an arbitrarily high value denoted as H. High and low rewards values are indicated with H^+ and H^- respectively. For the experiments included in this chapter, $H = 10^6$.

Event/Interaction Effect of two sprites being in contact during the execution of the game, being one of the entities involved the *avatar* (player) or an element generated from it. I differentiate between two types: *Collision* and *Action-onto*.

Collision A particular case of event where the entity involved is the player. An example of this kind of interaction is the player collecting a coin.

Action-onto A particular case of event where the entity involved is an item generated by the player. The item has typically been previously generated as a result of the execution of an action of type ACTION. An example of this kind of interaction is a bullet shot by the player hitting an enemy.

Curiosity An interaction between sprites that happens in a position of the map where it has not taken place before.

4.3 Goals and Heuristics Implementation

I define four goals that can be applied to several games, describing four heuristics: *Winning Maximisation Heuristic* (WMH), *Exploration Maximisation Heuristic* (EMH), *Knowledge Discovery Heuristic* (KDH), and *Knowledge Estimation Heuristic* (KEH). Their respective goals are: winning the game and maximising the score when a winning state is not immediately reachable; maximising the exploration of the level; interacting with the game as much as possible; predicting the outcome of the interaction with the different elements of the game, related with both the victory status and score modifications. I describe and present the implementation of these heuristics, which are general within the GVGAI Framework. The evaluation works with the information provided by the Framework API (Section 3.1). Therefore, the calculations of the final reward belonging to a particular state do not consider the specifics of the games or their rules. The heuristics affect the way a state of the game is evaluated, so they guide the search and policies of the agents.

4.3.1 Winning Maximisation Heuristic (WMH)

The **goal** of the Winning Maximisation Heuristic (WMH) is to win the game while achieving the highest score possible.

WMH penalises the end states where the player loses and rewards those where the player wins. In non-game-over states, the heuristic value is the difference between the final score and the score in the previous state. Algorithm 5 presents the pseudocode of the implementation of this heuristic.

Algorithm 5 Winning Maximisation Heuristic (WMH).
Nomenclature: $S' \leftarrow$ simulated game state; $H \leftarrow$ arbitrary high value; $lastScore \leftarrow$ score of the game in the previous state.

```
1: function WMH($S'$)
2:     if $isGameOver(S')$ and $isLoser(S')$ then
3:         return $H^-$
4:     if $isGameOver(S')$ and $isWinner(S')$ then
5:         return $H^+$
6:     $score \leftarrow getScore(S')$
7:     return $score - lastScore$
```

Heuristics focused on winning and maximising the score are very common in GVGP, and most of the sample agents provided in the GVGAI Framework implement a version of them. My research looks at using goals that go beyond these two. However, they represent the basis of my work, so I need to include them for comparison and as a starting point.

4.3.2 Exploration Maximisation Heuristic (EMH)

The **goal** of the Exploration Maximisation Heuristic (EMH) is to maximise the physical exploration of the level.

EMH rewards the agent that visits as many different available locations of the level as possible. At the beginning of the game, an empty exploration matrix is initialised. The exploration matrix corresponds to the 2D map of the level, mapping each of its tiles. Every time step, the information in the matrix is updated to mark the current position of the player as *visited*. The heuristic refers to this data when evaluating subsequent states. The heuristic value is given a positive reward when the player reaches a new position. Negative rewards are given if the agent does not move and remains in the same location in consecutive states. Algorithm 6 presents the pseudocode of the implementation of this heuristic.

Algorithm 6 Exploration Maximisation Heuristic (EMH).
Nomenclature: $S' \leftarrow$ simulated game state; $H \leftarrow$ arbitrary high value; $lastPosition \leftarrow$ position of the avatar in the previous state.

```
1:  function EMH($S'$)
2:      if $isGameOver(S')$ then
3:          return $H^-$
4:      $position \leftarrow getAvatarPosition(S')$
5:      if $isOutOfBounds(position)$ then          ▷ the position reached is outside the map
6:          return $H^-$
7:      if $\neg hasBeenBefore(position)$ then              ▷ the position reached is new
8:          return $H^+/100$
9:      if $position == lastPosition$ then                  ▷ the player has not moved
10:         return $H^-/200$
11:     return $H^-/400$                      ▷ no new position but the player has moved
```

Previous work related to the use of exploration as a goal in GVGP presented in Section 2.3.2, particularly the simple exploratory heuristic implemented by Nielsen et al. [2015], served as an inspiration to design EMH.

4.3.3 Knowledge Discovery Heuristic (KDH)

The **goal** of the Knowledge Discovery Heuristic (KDH) is to interact with the game as much as possible, triggering sprite spawns and interactions with the different elements that constitute the game.

KDH maximises the discovery of the different sprites of the game and encourages to perform interactions with them. It rewards those states where new sprites are acknowledged. I say that the agent *acknowledges* a sprite when its type has been observed during gameplay or the simulations reached with the forward model. At the beginning of the game, the *acknowledgement* of the sprites is initialised with the information available (i.e. what sprites are visible right before the game starts). Every interaction with sprites is recorded in an interaction table, distinguishing between two types: *collisions* and *actions-onto*. When no new sprites emerge, the heuristic prioritises carrying out new interactions. Lastly, the heuristic rewards *curiosity*, defined as those interactions that take place in new locations on the map. Algorithm 7 presents the pseudocode of the implementation of this heuristic.

Algorithm 7 Knowledge Discovery Heuristic (KDH).
Nomenclature: $S' \leftarrow$ simulated game state; $H \leftarrow$ arbitrary high value.

```
function KDH($S'$)
    if $isGameOver(S')$ and $isLoser(S')$ then
        return $H^-$
    if $isGameOver(S')$ and $isWinner(S')$ then
        return $H^-/2$
    $position \leftarrow getAvatarPosition(S')$
    if $isOutOfBounds(position)$ then        ▷ the position reached is outside the map
        return $H^-$
    if $newSpriteAcknowledge(S')$ then        ▷ new sprite appeared in the game
        return $H^+$
    $events \leftarrow getLastGameTickEvents(S')$ ▷ events in the last simulated game tick
    if $events \neq \emptyset$ then
        if $newInteraction(events)$ then        ▷ interaction with a new sprite
            return $H^+/10$
        if $newCollisionCuriosity(events)$ then        ▷ collision in a new position
            return $H^+/200$
        if $newActionOntoCuriosity(events)$ then        ▷ hit in a new position
            return $H^+/400$
    return $H^-/400$
```

4.3.4 Knowledge Estimation Heuristic (KEH)

The **goal** of the Knowledge Estimation Heuristic (KEH) is to predict the outcome of interacting with sprites, looking at changes in the *victory status* and *score modifiers*.

KEH acquires the best possible knowledge of the game dynamics to estimate the advantages and disadvantages of each possible interaction. The goal is to provide an estimation of the winning/losing conditions and score change when interacting with each element of the game. During the gameplay, the following information for the interaction type (*collisions* and *actions-onto*) with every sprite is gathered:

- *Score change.* Some interactions trigger a modification in the score of the game.

KEH accumulates the change in the score presumably derived from each interaction to calculate their average and estimate the score change at the end of the game. Ideally, the resulting prediction would be 0.0 if the interaction does not affect the score or a value (positive or negative) if it does.

- *Win condition.* The interactions between sprites can trigger a termination condition. After an interaction has been detected, the game may finish as a result of that particular interaction meeting a termination requirement. Consequently, KEH collects the total number of wins and defeats encountered for each interaction. This information is used to predict the win condition at the end of the game by simply calculating the average. Ideally, the resulting estimated value would be 0.0 if a specific interaction never produces a game over, or 1.0 or -1.0, in the case it triggers a termination condition that causes the player to win or lose, respectively.

Some sprites in the game may not be subjects of interactions by the avatar or the elements derived from it. As a result, a default value of 0.0 is given for both *win condition* and *score change*. Effectively, this initialisation assumes that, upon lack of interactions with a given sprite, no score or changes on game termination are triggered by this sprite. Rather than maximising the number of interactions with the same sprites, KEH attempts to interact uniformly with all the available ones to estimate the effects of these collisions better and improve upon the default estimations. Algorithm 8 presents the pseudocode of the implementation of this heuristic.

4.4 Controllers

I use five of the single-player algorithms provided by the sample pool of the GVGAI Framework: OSLA (Section 3.2.1), OLMCTS (Section 3.2.2), OLETS (Section 3.2.3), RHEA (Section 3.2.4), and RS (Section 3.2.5). The details about each of them are included in the corresponding sections.

All sample controllers use a heuristic to win and maximise the score by default. I modify the sample agents to isolate the evaluation function, so the heuristic can be assigned externally and be easily interchangeable. These adjustments do not affect the core of the algorithm (presented and described in detail in Section 3.2) but allow a fair comparison when using different heuristics and a quick experimental setup. The *heuristic diversification* I propose makes it possible to provide the goals externally. As a result, it is not required to hard-code or alter the agent's code to update them. The adjustments made are as follows:

1. All sample controllers extend from a newly created *AbstractHeuristicPlayer*, extending from the existent *AbstractPlayer* and replacing it. This class provides the heuristic when instantiating the controller. As a result, the goal and evaluation function are independent of the algorithm.

Algorithm 8 Knowledge Estimation Heuristic (KEH).
Nomenclature: $S' \leftarrow$ simulated game state; $interactionHistory \leftarrow$ all previous interactions before S'; $H \leftarrow$ arbitrary high value; $rewardLeast \leftarrow$ calculates the reward based on the number of times a certain interaction has happened compared to the total of interactions: the fewer times an interaction has happened, the highest the reward is; $balanceInteractionsReward \leftarrow$ balances the penalisation of the reward returning a value between $H^-/200$ and 0.

```
1: function KEH($S'$)
2:     $events \leftarrow getLastGameTickEvents(S')$    ▷ events in the last simulated game tick
3:     $updateInteractionStats(events)$    ▷ gather win condition and score change
4:     if $isGameOver(S')$ and $isLoser(S')$ then
5:         return $H^-$
6:     if $isGameOver(S')$ and $isWinner(S')$ then
7:         return $H^-/2$
8:     $position \leftarrow getAvatarPosition(S')$
9:     if $isOutOfBounds(position)$ then    ▷ the position reached is outside the map
10:        return $H^-$
11:    if $newSpriteAcknowledge(S')$ then    ▷ new sprite appeared in the game
12:        return $H^+$
13:    if $events \neq \emptyset$ then
14:        if $newInteraction(events)$ then    ▷ interaction with a new sprite
15:            return $H^+/10$
16:        else
17:            return $rewardLeast(events, interactionsHistory)$    ▷ $\in [0, H^+/100]$
18:    return $balanceInteractionsReward(interactionsHistory)$    ▷ $\in [H^-/200, 0]$
```

2. A form of cumulative reward is put in place for all algorithms. It allows algorithms with a long look-ahead (OLETS, OLMCTS, RHEA and RS) to keep track of the simulated states of the game at every step during the evaluation process. It also grants using rewards more accurately as algorithms have more information at their disposal than just the one provided at the termination state. Note that this is already in place by default in OSLA, as this agent only simulates one step ahead in the search.

3. Controllers make calls to the *heuristic* object to update its internal data, handle the heuristic accumulation and make a call to *evaluateState* to obtain the heuristic value in a particular state.

4. New methods are included in *ArcadeMachine*, the core of the GVGAI Framework. These methods adapt the existent ones to support the use of *AbstractHeuristicPlayer*. I instantiate the heuristic and provide it to the controller, create a player, and run a a single-player game with it.

I also set a common ground for a fair comparison between the algorithms and heuristics, as follows:

1. The search depth of the algorithms is set to 10. In some cases, this modification is not possible, given that the algorithm or the time provided prevents search from reaching this depth. The depth does not change for OLETS or OSLA.

Each of the five agents (OSLA, OLETS, OLMCTS, RHEA, and RS) is set to run with the four different heuristics presented in Section 4.3: WMH, EMH, KDH, and KEH, for a total of 20 different algorithm-heuristics configurations. The following section presents the games used for these experiments.

4.5 Games

There are hundreds of games available in the GVGAI Framework with different properties and characteristics. It is prohibitively expensive to use all to run the experiments, so I need to select a subset of them in a way that best represents the variety of games available. Previous authors have analysed and made a diverse selection of those games for their research, so I use a similar subset rooted in such selections. To carry out a selection of games from the GVGAI Framework, Gaina et al. [2017] combined two classifications presented in previous work and uniformly sampled them to come up with a diverse subset constituted by 20 games. Their final selection is formed by 10 deterministic and 10 stochastic games, and it was used for their experiments on Rolling Horizon evolutionary methods. I use the same subset of games in the experiments described in this chapter.

Tables 4.1 and 4.2 list the games used. They provide information about their type (deterministic or stochastic), timeout, available open locations accessible to the player, and details about the sprites of the game. These details include the number of initial sprites of that kind present in the level at $t = 0$ and their type:

- *From-avatar.* Sprite generated as a result of an ACTION of the player.
- *NPC.* Non-Player Character.
- *From-NPC.* Sprite generated as a result of an ACTION of an NPC.
- *Immovable.* Static element.
- *Movable.* Dynamic element.
- *Object.* Element the player can interact with.
- *Resource.* Element the player can collect.
- *Spawner.* Portal that generates sprites.

For some of the elements, I include more details: *invisible* (does not appear in the level), *goal* (an objective of the player involves reaching that particular sprite), or *no collision*, which means that interactions of the player with that element do not trigger any rule. Appendix A contains complete information about the games. It details the rules and includes screenshots of the level used for each.

4.6 Performance Comparison

The goal of the experiments is to compare the performance of the algorithms when given distinct behaviours. The performance when using each heuristic is measured based on parameters related to their corresponding goals. These experiments are constituted by five algorithms, four heuristics, and twenty games. Each of the 20 algorithm-heuristics configurations is set to play the level of each of the 20 games a total of 20 times. It results in 400 play-throughs for each agent with a particular heuristic, and, hence, the experiments execute a total of $8,000$ gameplays.

The average of the results obtained by the agents after playing each game several times is used to create rankings. These rankings are generated with the Formula 1 (F1) point system used in the GVGAI single-player competition [Perez-Liebana et al., 2015]. Five controllers are used in this experiment, so each receives 25, 18, 15, 12, or 10 points depending on their performance in each of the games. The final ranking of each heuristic is determined by the total sum of the scores received across the 20 games. This point system has been previously adopted by authors using the GVGAI framework for performance benchmarking [Pérez-Liébana et al., 2016], as it allows a fair perspective on the general performance of the agents when considering a set of different games.

I collect a series of game-playing stats to provide an overview of the overall performance of the agents, detailed for each heuristics in its corresponding section. The

Game	**Type**	**Timeout**	**Locations**	**Sprites**			
				Name	**Type**	**n at t=0**	**Details**
Aliens	Stochastic	2000	30	missile	from-avatar	0	
				alien	NPC	1	
				bomb	from-NPC	0	
				meteorite	immovable	47	
				portal	spawner	1	invisible
Bait	Deterministic	2000	9	door	immovable	1	goal
				key	object	1	
				box	object	2	
				wall	immovable	21	
Butterflies	Stochastic	2000	206	butterfly	NPC	6	
				cocoon	immovable	27	no collision
				wall (tree)	immovable	102	
Camel race	Deterministic	2000	322	camel	NPC	6	no collision
				goal	immovable	7	goal
				wall	immovable	110	
Chase	Deterministic	2000	135	white bird	NPC	7	
				black bird	NPC	0	
				carcass	immovable	0	no collision
				wall (tree)	immovable	129	
Chopper	Stochastic	2000	184	bomb	from-avatar	0	
				tank	NPC	1	
				missile	from-NPC	0	
				satellite	NPC	18	
				cloud	movable	36	no collision
				supply	resource	1	
				supply portal	spawner	2	
				base	spawner	1	
Crossfire	Stochastic	2000	333	turret	NPC	8	
				bomb	from-NPC	3	
				door	immovable	1	goal
				wall	immovable	132	
Digdug	Stochastic	2000	405	shovel	from-avatar	0	
				monster	NPC	2	
				gem	immovable	20	
				gold	immovable	7	
				wall	immovable	267	breakable
				boulder	movable	0	
				falling rock	movable	0	
				entrance	spawner	2	no collision
Escape	Deterministic	1000	74	hole	immovable	3	
				box	object	27	
				cheese	immovable	1	goal
				wall	immovable	45	
Hungry birds	Deterministic	2000	79	worm	immovable	1	
				goal	immovable	1	goal
				wall (tree)	immovable	97	
Infection	Stochastic	2000	187	sword	from-avatar	0	
				doctor	NPC	4	
				healthy person	NPC	17	
				infected person	NPC	0	
				virus	immovable	6	
				entrance	spawner	2	no collision
				wall	immovable	121	
Intersection	Stochastic	1000	243	car	NPC	13	
				goal	immovable	1	goal
				tree	immovable	32	
				wall	immovable	76	

Table 4.1: List of games used in the experiments (1/2). It includes the name, type, timeout, number of available locations, and information about the non-avatar sprites: type, initial number in the level at $t = 0$, and other details if any.

Game	**Type**	**Timeout**	**Locations**	**Sprites**			
				Name	**Type**	**n at t=0**	**Details**
Lemmings	Deterministic	2000	222	shovel	from-avatar	0	
				lemming	NPC	1	
				entrance	spawner	1	no collision
				door	immovable	1	no collision
				hole	immovable	9	
				wall	immovable	121	breakable
Missile Command	Deterministic	2000	242	explosive	from-avatar	0	
				fire	NPC	4	
				city	immovable	3	no collision
				wall	immovable	46	
Modality	Deterministic	2000	15	hole	object	1	
				bush	object	1	
				terrain 1	immovable	7	no collision
				terrain 2	immovable	7	no collision
				crossing point	immovable	1	
				wall	immovable	20	
Plaque Attack	Deterministic	2000	310	toothpase	from-avatar	0	
				clean tooth	immovable	5	no collision
				damaged tooth	immovable	0	
				hamburger	NPC	1	
				hot-dog	NPC	0	
				trolley	spawner	5	no collision
				wall	immovable	218	
Roguelike	Stochastic	2000	266	sword	from-avatar	0	
				spider	NPC	14	
				ghost	NPC	5	
				heart	resource	10	
				gold	resource	14	
				weapon	resource	1	
				city	immovable	3	
				key	resource	1	
				locked door	immovable	1	
				goal	immovable	1	goal
				wall	immovable	196	
Seaquest	Stochastic	1000	189	torpedo	from-avatar	0	
				whale	NPC	0	
				shark	NPC	0	
				piranha	NPC	0	
				diver	NPC	0	
				whirlpool	spawner	8	no collision
				ocean	immovable	168	no collision
				surface	immovable	22	
Survive Zombies	Stochastic	1000	121	zombie	NPC	1	
				priest	NPC	0	no collision
				tomb	spawner	3	no collision
				cloak	spawner	3	
				heart	object	14	
				wall	immovable	85	
Wait for breakfast	Deterministic	1000	50	chair	immovable	11	no collision
				table	immovable	5	
				waiter	NPC	0	
				door	spawner	1	no collision
				exit	immovable	1	

Table 4.2: List of games used in the experiments (2/2). It includes the name, type, timeout, number of available locations, and information about the non-avatar sprites: type, initial number in the level at $t = 0$, and other details if any.

purpose of some of the criteria used for the benchmark is to break ties. As a result, these criteria are not summarised in the stats table (e.g. game-ticks). Other data retrieved is highly dependent on the games and cannot be summarised in a unique table per heuristic (e.g. score for WMH). The stats that cannot be generalised between games are not included unless I have established a way to compare them. In some of the cases, this comparison is game-relative.

For each heuristic, the controllers are sorted according to their results in the games, based on the criteria tailored to the heuristic employed. Therefore, the final results present four rankings, one per heuristic (WMH, EMH, KDH, KEH), determining the best algorithms for each goal. After introducing and discussing all these individual rankings, I include a table that summarises the results of all the rankings for easy comparison between them.

4.6.1 WMH

Data gathered The gameplay data collected for the Winning Maximisation Heuristic includes three values:

- *win condition*: 1 or 0, determined by the game finishing with a victory for the agent or not, respectively.
- *final score*: number of points achieved at the end of the game.
- *timesteps to End of the Game (EoG)*: number of game-ticks played.

Performance criteria Given the results of different controllers in a game, the agent with a higher number of victories is considered better. In the case of a tie, higher scores are better. A lower average of game-ticks to victory is preferred (indicative that the game is won faster), while a higher value is preferred for games lost (which suggests a longer survival time). This ranking system is similar to the one used in the GVGAI Competition.

Results Based on the performance criteria detailed above, each controller is awarded 25, 18, 15, 12 or 10 points per game based on the results of their gameplay. The points received from each game are summed to obtain the total, which is used to get the final ranking. Table 4.3 presents the final ranking for the Winning Maximisation Heuristic detailing the number of points awarded per game to each of the controllers. Table 4.4 summarises the final ranking including the total number of victories achieved by each controller overall games as a reference. Tables and graphs with detailed information per game including the final stats used to distribute the F1 points are included in Appendix C.1.

The performance of RHEA is notably poor compared with the rest of the algorithms, being last in the ranking with a mere 10.00% rate of wins across the games. Its results

	Controller	G-1	G-2	G-3	G-4	G-5	G-6	G-7	G-8	G-9	G-10	G-11	G-12	G-13	G-14	G-15	G-16	G-17	G-18	G-19	G-20	Total
1	OLETS	25	25	15	25	25	25	25	25	18	25	25	25	18	15	25	15	25	25	18	25	449
2	RS	15	15	25	25	18	15	15	18	25	12	15	15	12	18	15	25	18	15	25	15	356
3	OLMCTS	18	18	18	25	15	18	12	15	12	18	18	12	15	25	18	18	15	18	15	10	333
4	OSLA	12	12	12	25	12	10	18	12	15	10	12	18	25	12	12	12	12	12	12	18	283
5	RHEA	10	10	10	25	10	12	10	10	10	15	10	10	10	10	10	10	10	10	10	12	224

Table 4.3: WMH results: Final ranking and F1 points awarded per game. G-1: *Aliens*, G-2: *Bait*, G-3: *Butterflies*, G-4: *Camel Race*, G-5: *Chase*, G-6: *Chopper*, G-7: *Crossfire*, G-8: *Digdug*, G-9: *Escape*, G-10: *Hungry Birds*, G-11: *Infection*, G-12: *Intersection*, G-13: *Lemmings*, G-14: *Missile Command*, G-15: *Modality*, G-16: *Plaque Attack*, G-17: *Roguelike*, G-18: *Seaquest*, G-19: *Survive Zombies*, G-20: *Wait for Breakfast*.

WMH Ranking			
	Controller	**F1 points**	**Total victories**
1	**OLETS**	**449**	**236/400**
2	RS	356	204/400
3	OLMCTS	333	166/400
4	OSLA	283	136/400
5	RHEA	224	40/400

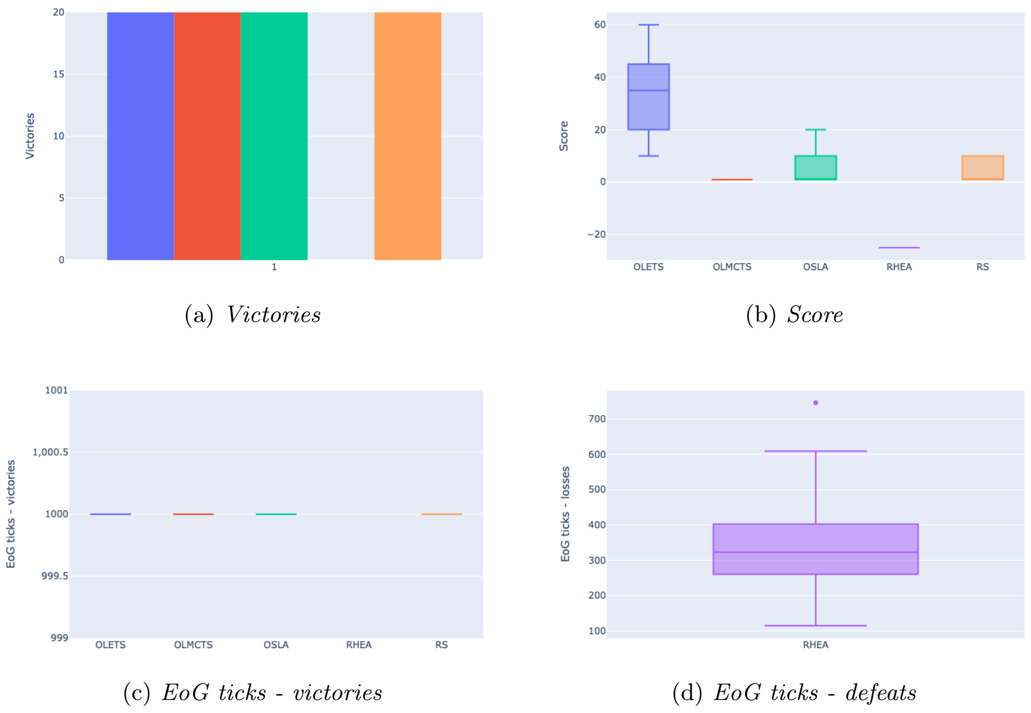
Table 4.4: WMH ranking summary. For reference, it shows the **total number of victories** across games achieved by each controller.

contrast with the 34.00% rate of wins obtained by OSLA (ranked in 4th position) and the 59.00% achieved by OLETS, ranked 1st and with the best stats for the WMH. It is worth mentioning that, specifically for the game *Intersection* (Table 4.5, Fig. 4.1), the difference between RHEA and the other algorithms is evident, as it reaches no victories when the rest of controllers achieve a rate of 100.00% wins in the game. Unlike RHEA, RS performs well, finishing in the 2nd position of the ranking with the second-best numbers. It is interesting that these two algorithms, while following a similar approach, have different performances. This suggests that further tweaking of the default RHEA parameters could land better results.

WMH: *Intersection*					
Points	**Controller**	**% Wins**	**Score**	**EoG victories**	**EoG defeats**
25	**OLETS**	100.00%	34.50 (3.20)	1000.00 (0)	-
18	**OSLA**	100.00%	4.70 (1.39)	1000.00 (0)	-
15	**RS**	100.00%	4.15 (0.96)	1000.00 (0)	-
12	**OLMCTS**	100.00%	1.00 (0)	1000.00 (0)	-
10	**RHEA**	0.00%	-25.00 (0)	-	347.45 (32.89)

Table 4.5: Results for the game *Intersection* showing: points received, controller, rate of wins (% Wins), average of score achieved, average of game-ticks when winning (EoG victories), and average of game-ticks when losing (EoG defeats). The values in parenthesis represent the corresponding *Std. Deviation*.

The resulting tables with performance on a game-per-game basis, as well as the graphs showing the stats of the 20 gameplays, are included in Appendix C.1. This sec-



(a) *Victories* (b) *Score* (c) *EoG ticks - victories* (d) *EoG ticks - defeats*

Figure 4.1: WMH results: *Intersection*. The graphs show the stats of 20 gameplays achieved by each agent, in order: OLETS (blue), OLMCTS (red), OSLA (green), RHEA (purple), and RS (orange).

tion only discusses some interesting results that I have observed.

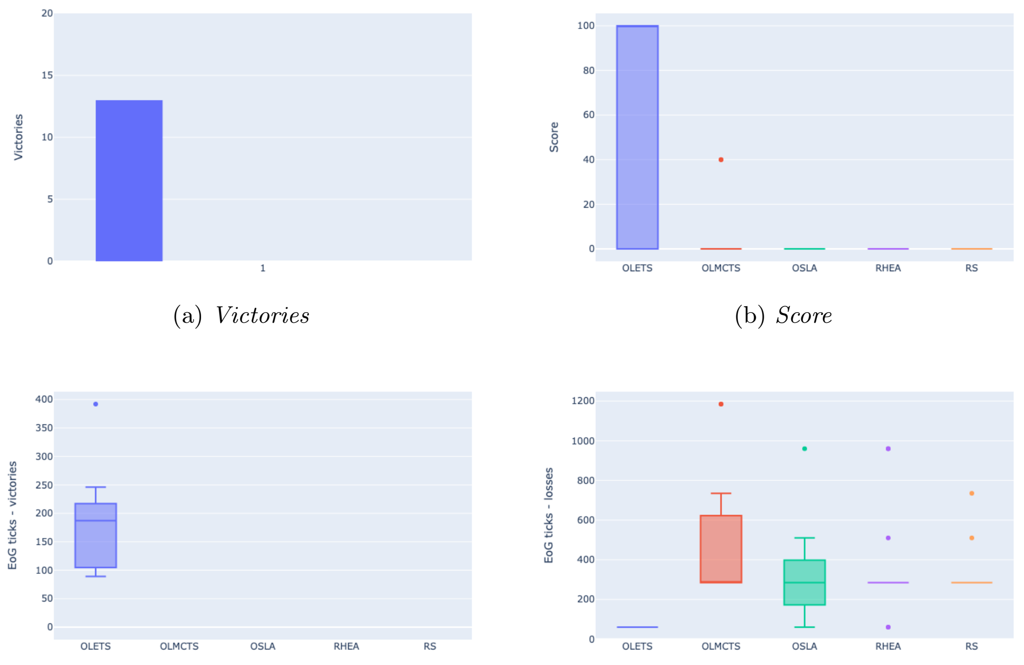
The controllers achieved no victories in *Camel Race*, *Digdug*, *Lemmings*, or *Roguelike*, meaning that none of the agents managed to win these games. The last three corresponds to games with big levels (Figs. A.8, A.13, and A.17) where the agent must carry out accurate actions to trigger the winning condition. Because of the limited resources employed in this heuristic, it is understandable that none of them managed to find the solution. However, it is interesting that *Camel Race* has not been solved, given that to win this game the player just needs to move in a straight line to the right, where the goal is located (Fig. A.4). This suggests that even the simplest game poses a challenge to the agents when the information about the dynamics of the game is restricted.

Hungry Birds (Table 4.6, Fig. 4.2) is very close to falling onto the category of unsolvable games, as most agents obtain a 0.00% rate of wins. However, OLETS managed to win it with an outstanding 65.00% rate of victories. Similarly, *Crossfire* (Table 4.7, Fig. 4.3) was only solved by three of the algorithms (OLETS, OSLA, and RS) with a 55.00% rate of wins for OLETS and a comparably low percentage achieved by the other two (5.00%).

Regarding games with an overall high rate of victories, I can refer to *Aliens*, *Infection*, *Intersection*, and *Modality* (Fig. 4.4). The average between the final rate of wins of the

WMH: *Hungry Birds*					
Points	**Controller**	**% Wins**	**Score**	**EoG victories**	**EoG defeats**
25	**OLETS**	65.00%	65.00 (10.67)	176.38 (22.77)	60.00 (0)
18	**OLMCTS**	0.00%	4.00 (2.68)	-	465.00 (64.82)
15	**RHEA**	0.00%	0.00 (0)	-	341.25 (50.00)
12	**RS**	0.00%	0.00 (0)	-	318.75 (24.00)
10	**OSLA**	0.00%	0.00 (0)	-	307.50 (47.46)

Table 4.6: Results for the game *Hungry Birds* showing: points received, controller, rate of wins (% Wins), average of score achieved, average of game-ticks when winning (EoG victories), and average of game-ticks when losing (EoG defeats). The values in parenthesis represent the corresponding *Std. Deviation.*



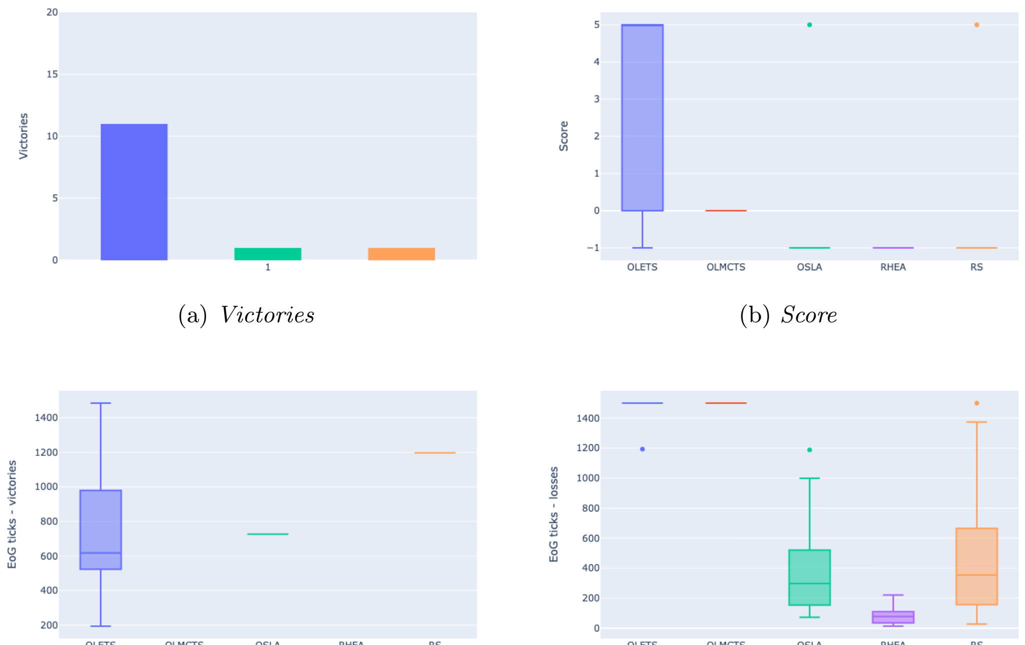
(a) *Victories* (b) *Score*

(c) *EoG ticks - victories* (d) *EoG ticks - defeats*

Figure 4.2: WMH results: *Hungry Birds*. The graphs show the stats of 20 gameplays achieved by each agent, in order: OLETS, OLMCTS, OSLA, RHEA, and RS.

WMH: *Crossfire*					
Points	**Controller**	**% Wins**	**Score**	**EoG victories**	**EoG defeats**
25	**OLETS**	55.00%	2.70 (0.57)	735.27 (103.56)	1465.89 (32.16)
18	**OSLA**	5.00%	-0.70 (0.29)	726.00 (0)	381.00 (66.57)
15	**RS**	5.00%	-0.70 (0.29)	1197.00 (0)	491.84 (96.77)
12	**OLMCTS**	0.0%	0.00 (0)	-	1500.00 (0)
10	**RHEA**	0.00%	-1.00 (0)	-	84.25 (13.59)

Table 4.7: Results for the game *Crossfire* showing: points received, controller, rate of wins (% Wins), average of score achieved, average of game-ticks when winning (EoG victories), and average of game-ticks when losing (EoG defeats). The values in parenthesis represent the corresponding *Std. Deviation.*



(a) *Victories* (b) *Score*

(c) *EoG ticks - victories* (d) *EoG ticks - defeats*

Figure 4.3: WMH results: *Crossfire*. The graphs show the stats of 20 gameplays achieved by each agent, in order: OLETS, OLMCTS, OSLA, RHEA, and RS.

agents in these games results in more than 80%. Still, RHEA does not achieve more than a 55% rate of victories in any of them.

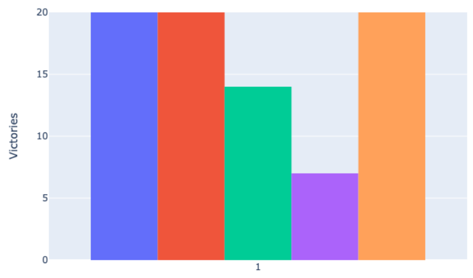
In summary, I conclude that OLETS works well when provided with the Winning Maximisation Heuristic, while RHEA shows an overall low performance.

4.6.2 EMH

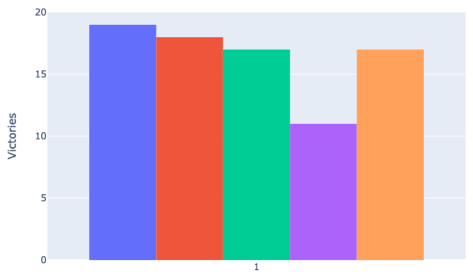
Data gathered The gameplay data collected for the Exploration Maximisation Heuristic includes the following values:

- *different positions visited*: total number of different locations of the map the agent explores. It corresponds to the number of positions of the exploration matrix marked as *visited* (Section 4.3.2).
- *timesteps to the last visit*: the latter game-tick when the player reached a new location (i.e. last time a new exploration happened).

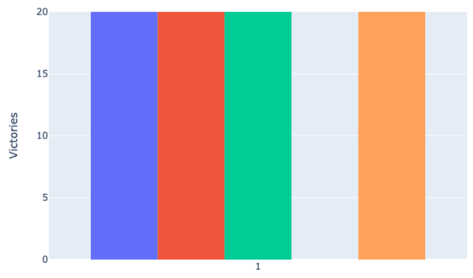
I calculate the final *exploration percentage* of the level using the total of available positions. This information is not provided with the agent's interface, so I have extracted the number of total possible locations corresponding to each game and level by hand. This total is detailed in Tables 4.1 and 4.2.



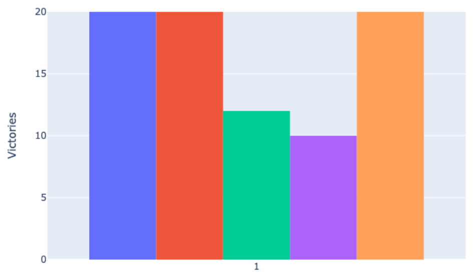
(a) *Aliens*



(b) *Infection*



(c) *Intersection*



(d) *Modality*

Figure 4.4: WMH results: games with resulting (overall) high rate of victories. The graphs show the *number of victories* of 20 gameplays achieved by each agent, in order: OLETS, OLMCTS, OSLA, RHEA, and RS.

Performance criteria The agent with a higher exploration percentage average is considered better. On ties, a lower game-tick for the last new visited position is preferred, as it rewards a faster exploration.

Results Based on the performance criteria detailed above, each controller is awarded 25, 18, 15, 12 or 10 points per game based on the results of their gameplay. The points received from each game are summed to obtain the total, which is used to get the final ranking. Table 4.8 presents the final ranking for the Exploration Maximisation Heuristic detailing the number of points awarded per game to each of the controllers. Table 4.9 summarises the final ranking including the average percentage of exploration overall games as a reference. Tables and graphs with detailed information per game including the final stats used to distribute the F1 points are included in Appendix C.2.

	Controller	**G-1**	**G-2**	**G-3**	**G-4**	**G-5**	**G-6**	**G-7**	**G-8**	**G-9**	**G-10**	**G-11**	**G-12**	**G-13**	**G-14**	**G-15**	**G-16**	**G-17**	**G-18**	**G-19**	**G-20**	**Total**
1	RS	25	18	15	25	15	18	18	25	25	18	25	25	25	25	18	25	18	25	25	15	428
2	OLETS	18	25	25	12	25	15	25	18	18	15	18	18	18	15	15	18	25	18	18	18	377
3	OLMCTS	15	12	18	18	18	25	12	15	10	12	15	12	15	18	12	15	15	15	12	25	309
4	OSLA	12	15	12	15	12	12	15	12	15	25	10	15	12	12	25	12	12	12	15	12	282
5	RHEA	10	10	10	10	10	10	10	10	12	10	12	10	10	10	10	10	10	10	10	10	204

Table 4.8: EMH results: Final ranking and F1 points awarded per game. G-1: *Aliens*, G-2: *Bait*, G-3: *Butterflies*, G-4: *Camel Race*, G-5: *Chase*, G-6: *Chopper*, G-7: *Crossfire*, G-8: *Digdug*, G-9: *Escape*, G-10: *Hungry Birds*, G-11: *Infection*, G-12: *Intersection*, G-13: *Lemmings*, G-14: *Missile Command*, G-15: *Modality*, G-16: *Plaque Attack*, G-17: *Roguelike*, G-18: *Seaquest*, G-19: *Survive Zombies*, G-20: *Wait for Breakfast*.

EMH Results			
	Controller	**F1 Points**	**Total average % explored**
1	RS	**428**	74.94%
2	OLETS	377	**76.86%**
3	OLMCTS	309	65.60%
4	OSLA	282	54.14%
5	RHEA	204	27.56%

Table 4.9: EMH ranking, presenting the **overall games stats**. Shows the overall average of percentage explored obtained by each of the controllers.

The overall average of the percentage of exploration for RS and OLETS is almost the same, being slightly higher for OLETS (74.94% and 76.86%, respectively). However, RS achieves 1^{st} position with a total of 428 points, being OLETS second with 377. OLMCTS ranks third with 309 points and an exploration percentage performance of 65.60%. RHEA has notably poor performance compared with the rest of the algorithms (27.56%) and ranks last, headed by OSLA, in 4^{th} place with an overall exploration of 54.14%.

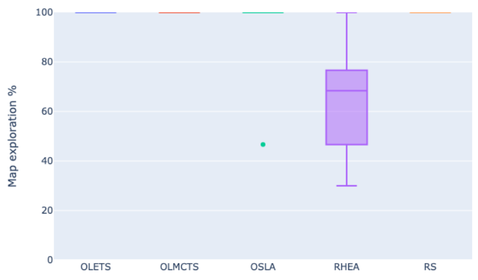
The resulting tables with performance on a game-per-game basis, as well as the graphs showing the stats of the 20 gameplays, are included in Appendix C.2. This section only discusses some interesting results that I have observed.

Only in *Aliens* (Table 4.10, Fig 4.5) some of the agents (RS, OLETS, and OLMCTS) achieve an average of 100.00% of exploration of the level, followed closely by OSLA with 97.33%. In comparison, RHEA achieves a 64%. The movement of the player in this game is horizontally (left and right only), and the level has only a few fully accessible available positions (30), which is a very low value compared with most of the games used in the experiments. RHEA achieving such a little performance in exploration in this game hints about the performance to expect from this agent when applying the EMH.

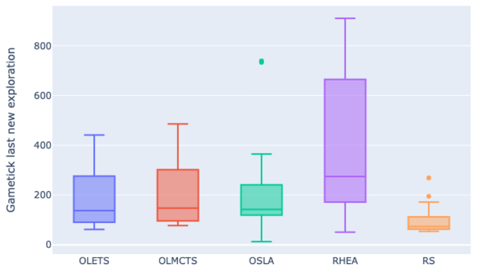
EMH: *Aliens*			
Points	**Controller**	**% Explored**	**Game-ticks last exploration**
25	**RS**	100.00% (0)	98.10 (12.11)
18	**OLETS**	100.00% (0)	184.35 (24.35)
15	**OLMCTS**	100.00% (0)	202.45 (28.23)
12	**OSLA**	97.33% (2.60)	216.25 (42.62)
10	**RHEA**	64.00% (4.37)	395.00 (61.09)

Table 4.10: Results for the game *Aliens* showing: points received, controller, average of percentage explored, average of game-ticks to last exploration. The values in parenthesis represent the corresponding *Std. Deviation.*

Games like *Butterflies*, *Chase*, *Chopper*, *Modality*, and *Survive Zombies* (Fig. 4.6) show a high exploration performance from the agents overall. Calculating the average of the final exploration rate of the agents results in more than 80%. In all these games,



(a) *Map exploration %*



(b) *Game-ticks last new exploration*

Figure 4.5: EMH results: *Aliens*. The graphs show the stats of 20 gameplays achieved by each agent, in order: OLETS, OLMCTS, OSLA, RHEA, and RS.

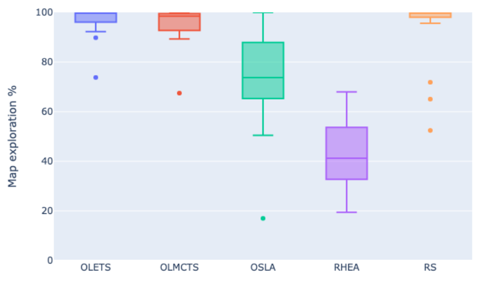
the level played is completely accessible, meaning that the agent does not have to make an action or interact with other sprites to be able to reach the locations available.

Only two of the games result in a significant mediocre performance (less than 25%) when calculating the average of the final percentage of exploration of all the controllers: *Camel Race* and *Roguelike*. *Camel Race* (Table 4.11, Fig 4.7) has a quick game over because the NPC wins the game rapidly and it probably did not give the agents enough time to explore. The maximum average of exploration percentage achieved for this game is obtained by RS with a 23.71%, closely followed by OLMCTS (22.87%), OSLA (20.61%), and OLETS (19.83%). RHEA obtains an average of 7.83%.

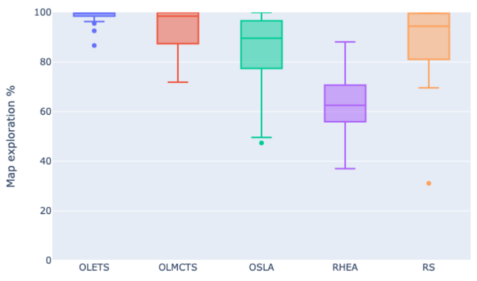
EMH: *Camel Race*			
Points	**Controller**	**% Explored**	**Game-ticks last exploration**
25	**RS**	23.71% (0.27)	78.90 (0.10)
18	**OLMCTS**	22.87% (0.34)	78.80 (0.19)
15	**OSLA**	20.61% (0.67)	76.45 (1.27)
12	**OLETS**	19.83% (0.42)	77.95 (0.80)
10	**RHEA**	7.83% (0.34)	65.15 (2.30)

Table 4.11: Results for the game *Camel Race* showing: points received, controller, average of percentage explored, average of game-ticks to last exploration. The values in parenthesis represent the corresponding *Std. Deviation*.

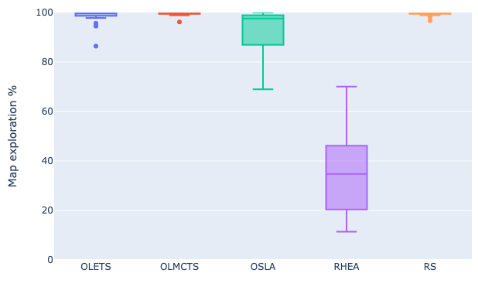
Roguelike (Table 4.12, Fig 4.8) has a big map with two separate zones (Fig. A.17), and the player needs to carry out a specific task (collect a key) to gain access to the second part of the map (73 of the 266 locations). The accessible area covers 72.55% of the whole level. Given that the highest average percentage of exploration achieved by the best performer controller, OLETS, is 42.22%, I speculate that none of the agents managed to gain access to the second area. In addition, the game and this specific level have several enemies that kill the player, so it is quite probable that the agents were killed early in their gameplays, not being able to explore the level enough.



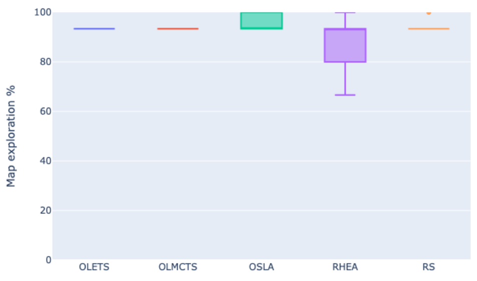
(a) *Butterflies*



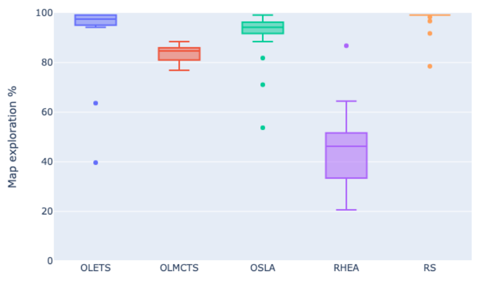
(b) *Chase*



(c) *Chopper*

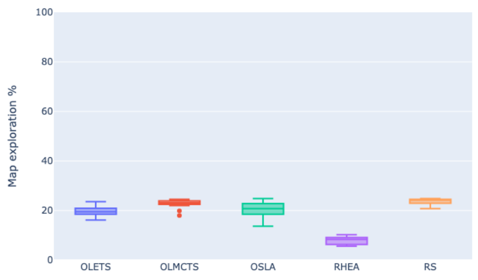


(d) *Modality*

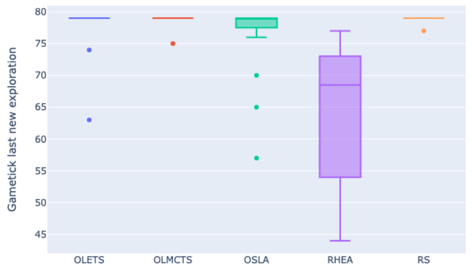


(e) *Survive Zombies*

Figure 4.6: EMH results: games with resulting (overall) high exploration performance. The graphs show the *Map exploration %* of 20 gameplays achieved by each agent, in order: OLETS, OLMCTS, OSLA, RHEA, and RS.



(a) *Map exploration %*

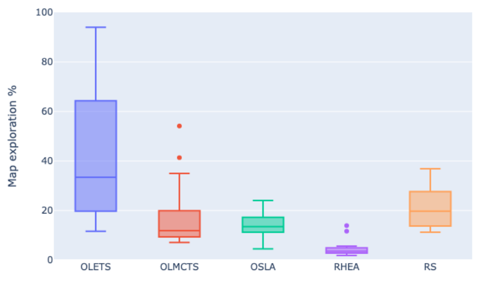


(b) *Game-ticks last new exploration*

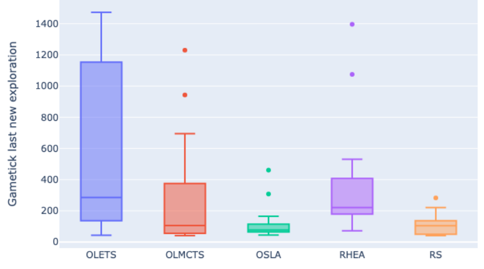
Figure 4.7: EMH results: *Camel Race*. The graphs show the stats of 20 gameplays achieved by each agent, in order: OLETS, OLMCTS, OSLA, RHEA, and RS.

EMH: *Roguelike*			
Points	**Controller**	**% Explored**	**Game-ticks last exploration**
25	**OLETS**	42.22% (5.50)	551.70 (116.54)
18	**RS**	20.71% (1.77)	109.90 (14.47)
15	**OLMCTS**	17.61% (2.86)	268.20 (73.45)
12	**OSLA**	14.12% (1.15)	112.70 (21.89)
10	**RHEA**	4.51% (0.67)	343.90 (73.38)

Table 4.12: Results for the game *Roguelike* showing: points received, controller, average of percentage explored, average of game-ticks to last exploration. The values in parenthesis represent the corresponding *Std. Deviation.*



(a) *Map exploration %*



(b) *Game-ticks last new exploration*

Figure 4.8: EMH results: *Roguelike*. The graphs show the stats of 20 gameplays achieved by each agent, in order: OLETS, OLMCTS, OSLA, RHEA, and RS.

I conclude that for the Exploration Maximisation Heuristic, most agents have a similar overall performance except for RHEA, who shows a poor performance in comparison.

4.6.3 KDH

Data gathered The gameplay data collected for the Knowledge Discovery Heuristic includes the following values:

- *sprites acknowledged*: sprites with different IDs observed during gameplay or the forward model simulations.
- *interactions*: unique interactions with sprites of each type. A distinction between *collisions* and *actions-onto* interactions is made.
- *curiosity*: interactions with sprites in different locations of the map. A distinction between *collisions* and *actions-onto* interactions is made.
- *timesteps to the last acknowledgement*: the latter game-tick when a new sprite was acknowledged (i.e. last time a new acknowledgement happened).
- *timesteps to the last interaction*: the latter game-tick when a new event between two sprites happened (i.e. last time a new interaction took place).

- *timesteps to the last curiosity*: the latter game-tick when an interaction with a sprite in a new location happened (i.e. last time a new curiosity took place).

The total number of known sprites per game and level is undefined, so instead of using a set total, the percentage used as benchmark is based on the relative data of the agents per game. Therefore, in the results, the controller with the highest number of acknowledged sprites is assigned with a 100% *performance* on the 'acknowledged sprites' category. That total number achieved by that agent is the one taken as reference as the *total number of sprites*. Therefore, the final percentage obtained in the results and stats are relative and dependant on the best value obtained during gameplay.

Performance criteria The agent has better performance when obtaining, in order: a higher number of acknowledged sprites, a higher number of unique interactions, a higher curiosity, and a lower game-ticks to last acknowledgement, last interaction and last curiosity.

Results Based on the performance criteria detailed above, each controller is awarded 25, 18, 15, 12 or 10 points per game based on the results of their gameplay. The points received from each game are summed to obtain the total, which is used to get the final ranking. Table 4.13 presents the final ranking for the Knowledge Discovery Heuristic detailing the number of points awarded per game to each of the controllers. Table 4.14 summarises the final ranking including the game-relative percentage of sprites acknowledged, interactions, curiosity collisions and curiosity actions-onto. Tables and graphs with detailed information per game including the final stats used to distribute the F1 points are included in Appendix C.3.

	Controller	G-1	G-2	G-3	G-4	G-5	G-6	G-7	G-8	G-9	G-10	G-11	G-12	G-13	G-14	G-15	G-16	G-17	G-18	G-19	G-20	Total
1	RS	25	15	18	15	25	25	25	18	18	18	18	25	18	18	25	25	18	25	15	25	414
2	RHEA	15	25	12	12	18	12	12	25	25	25	25	12	25	12	15	12	15	10	25	10	342
3	OLMCTS	18	18	25	18	15	10	15	15	15	15	15	10	15	25	18	10	25	18	18	12	330
4	OLETS	12	12	15	25	12	18	10	12	10	12	12	18	12	15	12	15	12	15	12	18	279
5	OSLA	10	10	10	10	10	15	18	10	12	10	10	15	10	10	10	18	10	12	10	15	235

Table 4.13: KDH results: Final ranking and F1 points awarded per game. G-1: *Aliens*, G-2: *Bait*, G-3: *Butterflies*, G-4: *Camel Race*, G-5: *Chase*, G-6: *Chopper*, G-7: *Crossfire*, G-8: *Digdug*, G-9: *Escape*, G-10: *Hungry Birds*, G-11: *Infection*, G-12: *Intersection*, G-13: *Lemmings*, G-14: *Missile Command*, G-15: *Modality*, G-16: *Plaque Attack*, G-17: *Roguelike*, G-18: *Seaquest*, G-19: *Survive Zombies*, G-20: *Wait for Breakfast*.

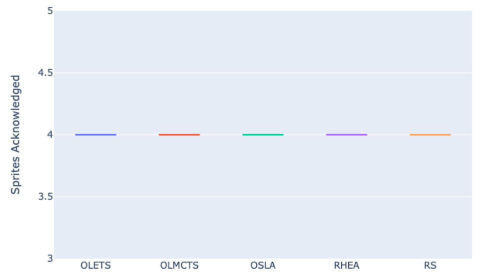
RS obtains the first position achieving a total of 414 points, followed by RHEA (342 points), OLMCTS (330 points), OLETS (279 points), and OSLA (235 points). RS is the best performing agent, achieving a 100% of performance in acknowledgement, meaning that it is the controller that has acknowledged the highest number of sprites in every game. It achieves the highest or second-highest performance for all the stats, which is a remarkable result. The best performance for *collision curiosity* is achieved by OLETS with a total of 90.72%. The algorithm that performs the worst and ranks last is OSLA. Although its performance acknowledging the elements and interacting with them is over

KDH Results						
	Controller	**F1 Points**	**% Ack (Rel.)**	**% Int. (Rel.)**	**% CC (Rel.)**	**% CA (Rel.)**
1	RS	**414**	**100.00%**	**96.18%**	85.46%	**87.42%**
2	RHEA	342	99.66%	95.48%	62.48%	54.44%
3	OLMCTS	330	99.79%	93.53%	84.75%	84.06%
4	OLETS	279	99.86%	88.97%	**90.72%**	77.55%
5	OSLA	235	98.48%	84.99%	56.37%	51.75%

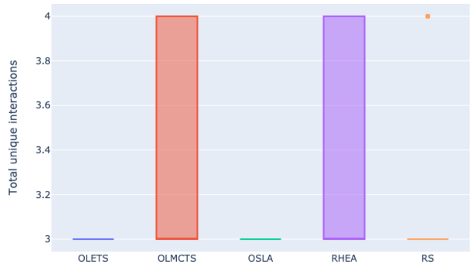
Table 4.14: KDH ranking and **overall games stats**. The stats show game-relative (Rel.) percentages for sprites acknowledged (Ack.), interactions (Int.), Curiosity Collisions (CC) and Curiosity Actions-onto (CA)

84.99%, its curiosity performance is just 56.37% and 51.75%.

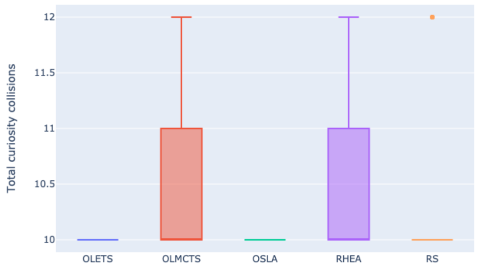
The resulting tables with performance on a game-per-game basis, as well as the graphs showing the stats of the 20 gameplays, are included in Appendix C.3. Only in three of the games, all controllers have interacted with more than 75% of the sprites they acknowledged during gameplay: *Bait* (Fig 4.9), *Digdug* (Fig 4.10), and *Intersection* (Fig 4.11).



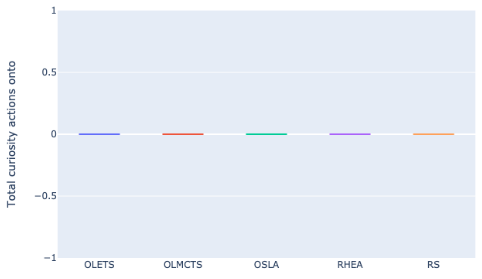
(a) *Sprites acknowledged*



(b) *Unique interactions*



(c) *Curiosity collisions*

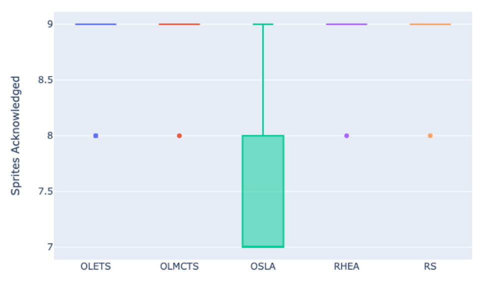


(d) *Curiosity actions-onto*

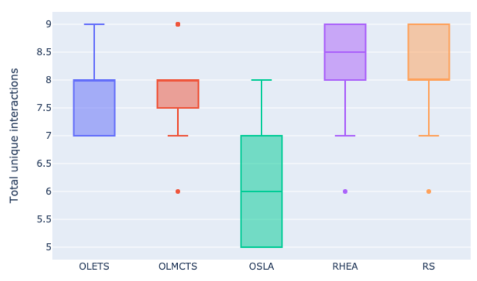
Figure 4.9: KDH results: *Bait*. The graphs show the stats of 20 gameplays achieved by each agent, in order: OLETS, OLMCTS, OSLA, RHEA, and RS.

In *Camel Race*, on the other hand, they only interact with 1 of the 7 acknowledged (Fig 4.12).

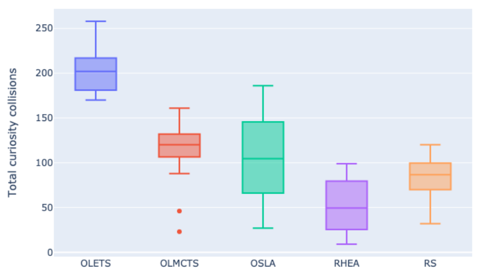
The number of unique interactions with the sprites gives an idea of the number of



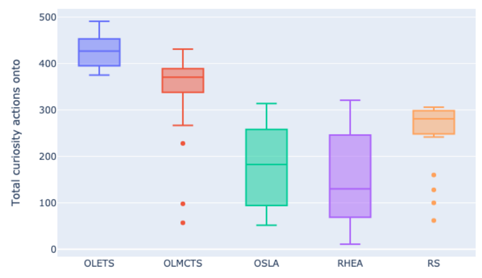
(a) *Sprites acknowledged*



(b) *Unique interactions*

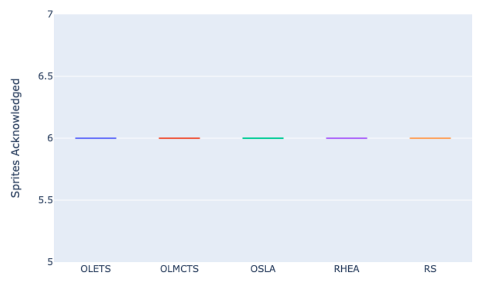


(c) *Curiosity collisions*

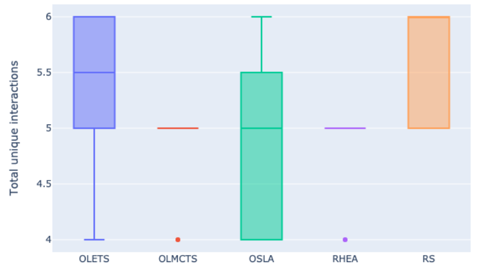


(d) *Curiosity actions-onto*

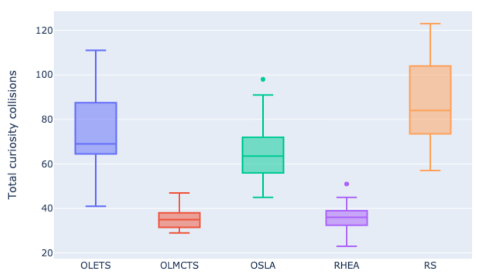
Figure 4.10: KDH results: *Digdug*. The graphs show the stats of 20 gameplays achieved by each agent, in order: OLETS, OLMCTS, OSLA, RHEA, and RS.



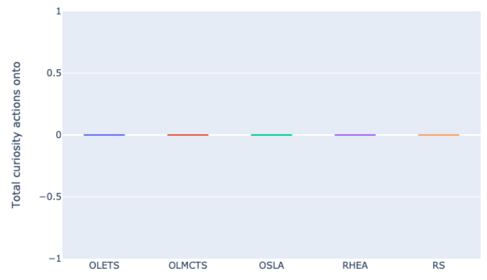
(a) *Sprites acknowledged*



(b) *Unique interactions*

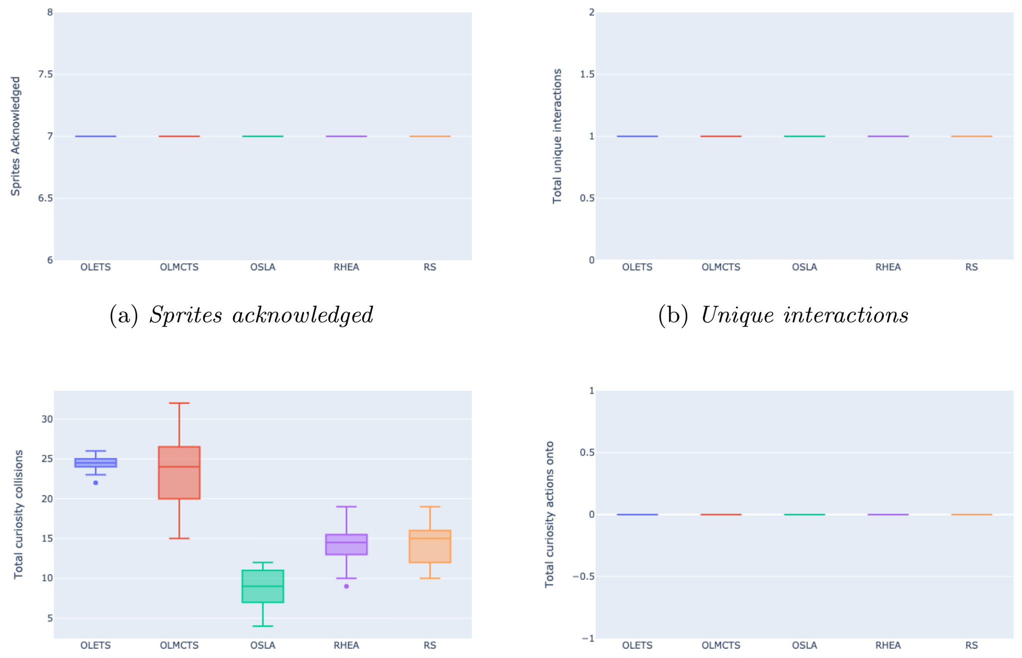


(c) *Curiosity collisions*



(d) *Curiosity actions-onto*

Figure 4.11: KDH results: *Intersection*. The graphs show the stats of 20 gameplays achieved by each agent, in order: OLETS, OLMCTS, OSLA, RHEA, and RS.



(a) *Sprites acknowledged* (b) *Unique interactions*

(c) *Curiosity collisions* (d) *Curiosity actions-onto*

Figure 4.12: KDH results: *Camel Race*. The graphs show the stats of 20 gameplays achieved by each agent, in order: OLETS, OLMCTS, OSLA, RHEA, and RS.

interactive elements in the game. In contrast, the rate of each type of interaction informs how it is possible to interact with them. It can also provide some information about the rules. For example, if the number of sprites interacted with is high but the resulting number of *actions-onto curiosity* is 0, it may be possible that the agent cannot carry out actions.

4.6.4 KEH

Data gathered The gameplay data collected for the Knowledge Estimation Heuristic reflects how well the agent can estimate the dynamics of the game in terms of changes on the victory status and score awarded from interactions with the elements. Therefore, *estimations of the outcomes* of each interaction type (*collisions* and *actions-onto*) are obtained and provided at the end of the game. These estimations must be compared with the true outcome to determine how accurate the predictions are. To determine the level of accuracy, for every game, I have extracted manually the ground truth regarding sprites that cause score change or winning/losing states. For every prediction given by the agent, the square error to the ground truth is calculated. The mean of square errors determines the total prediction error incurred by the agent in a game. The interaction percentage is game-relative, considering the highest value given by a controller for each game as a benchmark to obtain the performance for that game. The value displayed in the table has been obtained with the average of these values for all games.

Performance criteria An agent is considered better based on the quality of their predictions: the best controller is the one with the smallest average of square errors. In the (rare) case of a tie, the number of events that the controller can give a prediction for is used as a tie-breaker, considering the higher, the better.

Results Based on the performance criteria detailed above, each controller is awarded 25, 18, 15, 12 or 10 points per game based on the results of their gameplay. The points received from each game are summed to obtain the total, which is used to get the final ranking. Table 4.15 presents the final ranking for the Knowledge Estimation Heuristic detailing the number of points awarded per game to each of the controllers. Table 4.16 summarises the final ranking including the average of square error accross games and the game-relative percentage of interactions estimated. Tables with detailed information per game including the final stats used to distribute the F1 points are included in Appendix C.4.

	Controller	**G-1**	**G-2**	**G-3**	**G-4**	**G-5**	**G-6**	**G-7**	**G-8**	**G-9**	**G-10**	**G-11**	**G-12**	**G-13**	**G-14**	**G-15**	**G-16**	**G-17**	**G-18**	**G-19**	**G-20**	**Total**
1	OLMCTS	18	15	10	10	25	18	25	15	12	18	25	15	15	18	15	25	15	10	18	25	347
2	RHEA	15	25	18	15	15	25	12	10	25	12	15	12	18	15	10	18	12	18	25	15	330
3	OSLA	12	10	12	18	10	10	18	25	10	10	10	25	25	12	25	10	18	25	10	18	313
4	RS	25	18	25	12	18	12	15	12	15	15	12	18	12	25	12	15	10	15	12	12	310
5	OLETS	10	12	15	25	12	15	10	18	18	25	18	10	10	10	18	12	25	12	15	10	300

Table 4.15: KEH results: Final ranking and F1 points awarded per game. G-1: *Aliens*, G-2: *Bait*, G-3: *Butterflies*, G-4: *Camel Race*, G-5: *Chase*, G-6: *Chopper*, G-7: *Crossfire*, G-8: *Digdug*, G-9: *Escape*, G-10: *Hungry Birds*, G-11: *Infection*, G-12: *Intersection*, G-13: *Lemmings*, G-14: *Missile Command*, G-15: *Modality*, G-16: *Plaque Attack*, G-17: *Roguelike*, G-18: *Seaquest*, G-19: *Survive Zombies*, G-20: *Wait for Breakfast*.

KEH Results				
	Controller	**F1 Points**	**Avg. square error**	**% Int. estimated**
1	OLMCTS	**347**	**0.338**	97.92%
2	RHEA	330	0.505	97.50%
3	OSLA	313	0.617	73.19%
4	RS	310	0.528	**98.33%**
5	OLETS	300	1.086	87.92%

Table 4.16: KEH ranking and **overall games stats**. The stats show the overall average of the *square error average* and the **game-relative** (Rel.) percentage for the interactions estimated (Int. estimated).

OLMCTS ranks first with a total of 347 points, closely followed by RHEA with 330. The last three positions (OSLA, RS, and OLETS) are very close to each other in the number of points achieved: 313, 310 and 300, respectively. In addition, when looking at the overall average square error, results are not very satisfactory, as the average for the agent with best performance overall games (OLMCTS) is 0.338. Overall, none of the agents has a remarkably better performance than the others, as the difference of points between OLMCTS and OLETS is just 47. However, it is worth mentioning that, unless there is a poor overall performance, the estimations for some of the interactions in certain games are very accurate. There are a few cases where the outcome of the interaction with a sprite has been estimated by all the controllers with an average square error of 0.00. Most of these cases are related with estimations for the interactions of the type *collision*.

The resulting tables with performance on a game-per-game basis are included in Appendix C.4. Individual resulting predictions per agent and game are provided in Appendix C.5. This section only discusses some interesting results that I have observed.

In *Aliens*, all five controllers achieved an accurate estimation for the collisions with both the *bomb* (stype 6) and the *alien* (stype 9), predicting an inarguable defeat when the player interacts with any of them. These results are presented in Appendix tables C.81, C.82, C.83, C.84, and C.85. As a result, none of the controllers stand out in the final ranking of this game as the average square error obtained is quite similar between them (Table 4.17).

KEH: *Aliens*			
Points	**Controller**	**Avg. Square Error**	**Interactions Estimated**
25	**RS**	1.11E-01	4
18	**OLMCTS**	1.18E-01	4
15	**RHEA**	1.20E-01	4
12	**OSLA**	1.21E-01	4
10	**OLETS**	1.25E-01	4

Table 4.17: Results for the game *Aliens* showing: average of the square error obtained and total interactions estimated.

Another example of great accuracy in the predictions is found in *Chase*, where a sprite of type *angry bird* can (or not) emerge at some point during the game (Section A.5). The *angry bird* kills the avatar when colliding with it and decreases the score by one point. OLMCTS, RHEA, and RS discovered this element of the game (stype 5) and predicted both the winning condition and score change with an average square error of 0.00 (Table 4.18). See Appendix tables C.103, C.104, and C.105 for all the predictions of those agents for this game.

Finally, RHEA predicted every outcome of every sprite interaction for the game *Escape* with an average square error of 0.00 (Table 4.19), which is a remarkable achievement. The rest of the controllers also achieved accurate predictions, as the final square errors obtained are very small (Table 4.20).

The examples described above show good performance when analysing the data obtained. However, I also encounter less than optimal performance in many of the games studied. In overall terms, the results show the challenge of predicting and understanding a game when its information is limited. None of the controllers has a remarkably bet-

KEH: Predictions Stype 5 (*angry bird*) in *Chase*						
Controller	**Ground Truth**		**Estimations**		**Accuracy Square Error**	
	CW	**CS**	**CW**	**CS**	**CW**	**CS**
OSLA	-1	-1	-	-	-	-
OLETS	-1	-1	-	-	-	-
OLMCTS	-1	-1	-1.00 (0.00)	-1.00 (0.00)	0.00E+00	0.00E+00
RHEA	-1	-1	-1.00 (0.00)	-1.00 (0.00)	0.00E+00	0.00E+00
RS	-1	-1	-1.00 (0.00)	-1.00 (0.00)	0.00E+00	0.00E+00

Table 4.18: Summary predictions of stype 5 (angry bird) for the game *Chase*. Shows the ground truth of every collision, as well as the estimations and square error obtained by each agent for every collision win (CW) and collision score (CS). The values in parenthesis represent the corresponding *Std. Deviation*.

RHEA Predictions: *Escape*												
Stype	**Ground Truth**				**Estimations**				**Accuracy Square Error**			
	CW	**CS**	**AW**	**AS**	**CW**	**CS**	**AW**	**AS**	**CW**	**CS**	**AW**	**AS**
0	0	0	-	-	0.00 (0.00)	0.00 (0.00)	-	-	0.00E+00	0.00E+00	-	-
3	0	0	-	-	0.00 (0.00)	0.00 (0.00)	-	-	0.00E+00	0.00E+00	-	-
4	1	1	-	-	1.00 (0.00)	1.00 (0.00)	-	-	0.00E+00	0.00E+00	-	-
5	-1	-1	-	-	-1.00 (0.00)	-1.00 (0.00)	-	-	0.00E+00	0.00E+00	-	-

Table 4.19: KEH predictions of RHEA for the game *Escape*. Shows the id of the sprite (Stype), the ground truth of every interaction, as well as the estimations and square error obtained by the agent for every type of interaction: collision win (CW), collision score (CS), action-onto win (AW) and action-onto score (AS). The values in parenthesis represent the corresponding *Std. Deviation*.

KEH: *Escape*			
Points	**Controller**	**Avg. Square Error**	**Interactions Estimated**
25	**RHEA**	0.00E+00	4
18	**OLETS**	6.71E-08	4
15	**RS**	5.81E-07	4
12	**OLMCTS**	2.50E-01	3
10	**OSLA**	5.00E-01	1

Table 4.20: Results for the game *Escape* showing: average of the square error obtained and total interactions estimated.

ter performance than the others. However, RHEA performs better with the KEH than with any of the previous heuristics, being able to achieve in some cases more accurate predictions than the rest of the controllers.

4.6.5 Results summary and discussion

The experiments confirm that the performances of the agents are different, so the external heuristic has an effect on the efficiency of the GVGP agents. By analysing the results per game, I have noticed that the performance is influenced by its characteristics and the level considered.

Taking the RS algorithm with EMH as an example, there are clear distinctions in the performance depending on the size of the map, the obstacles, and the rules of the game (Fig. 4.13).

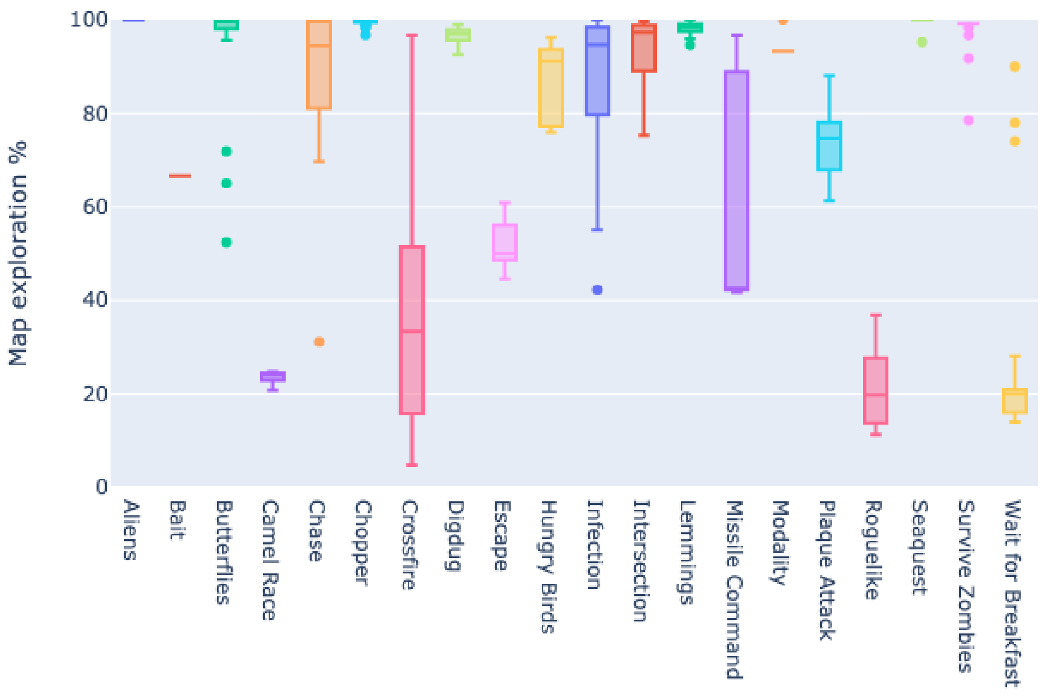


Figure 4.13: EMH results per game: RS. The graphs show the *Map exploration %* of 20 gameplays achieved by RS in each game.

In small or easy accessible maps as the ones presented in *Aliens*, *Butterflies*, *Chase*, *Chopper*, *Digdug*, *Hungry Birds*, *Infection*, *Intersection*, *Lemmings*, *Modality*, *Seaquest*, and *Survive Zombies*, the agent presents an average performance higher than 85%, being higher than 90% for most of them. On the contrary, those games that present obstacles that block the path (*Bait* and *Escape*), have large or not easy accessible maps (*Crossfire* and *Roguelike*), or where the agent plays against time (*Camel Race* and *Wait for Breakfast*), their performance drops.

These observations are interesting. They show how even the most performing agent for a particular goal can be affected by the characteristics of the game and level, attesting to the difficulty of developing general heuristics. However, they are also inspiring. The existence of differences in the resulting performance based on the type of game (or characteristics of the level) could be used to assist in the development and testing of games. I explore this idea in detail in Chapter 5.

Table 4.21 provides an overview of the rankings obtained for each of the heuristics. It displays the position and the total number of points achieved by each controller in each set of experiments. The number of points scored by the controllers in the first position for WMH, EMH, KDH, and KEH is, respectively, 449, 428, 414, and 347. Note that the difference between the points achieved by the agents in the first and last positions is higher for WMH, EMH, and KDH than for KEH. Overall, KEH shows a more uniform distribution of points. This result suggests that the performance of the algorithms using KEH is very similar, with no algorithm showing a clear dominance.

Rankings								
	WMH		**EMH**		**KDH**		**KEH**	
1st	**449**	**OLETS**	**428**	**RS**	**414**	**RS**	**347**	**OLMCTS**
2nd	356	RS	377	OLETS	342	RHEA	330	RHEA
3rd	333	OLMCTS	309	OLMCTS	330	OLMCTS	313	OSLA
4th	283	OSLA	282	OSLA	279	OLETS	310	RS
5th	224	RHEA	204	RHEA	235	OSLA	300	OLETS

Table 4.21: Final ranking for each heuristic showing the total number of F1-points achieved by each controller in each independent set of experiments: WMH, EMH, KDH, and KEH.

The results obtained are very heterogeneous. First, three different controllers (OLETS, RS, and OLMCTS) reach the first position, and three (RHEA, OSLA, and OLETS) rank last in at least one of the experiments. In addition, there is a noticeable difference in the rankings and order of the controllers for each heuristic. RHEA performs poorly for both WMH and EMH, being last and with a significant difference in scores with OSLA, in 4th place. However, it appears on the top of the ranking for KDH and KEH, reaching the 2nd position in both of them with 342 and 330 points. OLETS performs very well for WMH and EMH, but ranks 4th and last in both heuristics involving knowledge. OSLA maintains a similar position over all the heuristics (4th for WMH and EMH, last for KDH, and 3rd for KEH). OLMCTS obtains a medium performance for WMH, EMH, and KDH and reaches the first position for KEH. Finally, RS has generally good performance, being second for WMH, first for EMH and KDH. In its fourth position achieved for KEH, the difference of points in the rank is not as high as in the other rankings.

Heuristic diversification provides GVGP agents with goals beyond winning, but the results demonstrate how it would not be enough just to design and implement the heuristic. It is necessary to understand the impact of the heuristic on the performance of the agent that uses it. Carrying out a similar experiment to the ones presented in this Chapter should help to identify the best choice based on the needs. It would be reasonable to choose the agent with the best results based on the heuristic chosen, or if more than one heuristic is brought together, the agent with steady results in overall experiments. In my case, if combining WMH and EMH, the choice would be between using RS or OLETS, but if knowledge is also included, it would be preferable to use RS or OLMCTS. The heuristics have been presented as independent entities, but it is sensible to think that they can all be helpful in different circumstances. For example, it sounds reasonable to use KDH and KEH at the first stage of play to achieve a better understanding of the game. Once certain conditions have been met, these discoveries can influence the usage of WMH and EMH. It should be possible to design a high-level meta-heuristic algorithm to combine and select different agents and heuristics during gameplay. Having available a set of general algorithms with a variety of objectives (provided by the pertinent heuristic) would allow accommodating to different situations that emerge during gameplay and switching behaviour in response to the environment. I do not look into this in my research, but it is an interesting open future work.

The core of the agents is the same in every experiment, so I conclude that the discrepant rankings are influenced by the heuristic provided and, therefore, the goal assigned to the algorithms. I have been able to confirm that the performances are different, but what about their behaviour? Next, I examine the agents playing the games with each heuristic to study the differences in how they act and react to the game.

4.7 Analysing the Distinct Behaviour

This section provides an extension of the work described in this chapter and serves as an inspiration for the following chapters. I take the algorithms with each of the heuristics implemented and analyse their behaviour when playing the same game. The goal is to identify distinct behaviours during gameplay when exposing the algorithms to different general objectives without modifying their core.

For the analysis included in this section, I only use RS, as it maintains a generally good performance across heuristics (Table 4.21). It resulted in second place for WMH, first for both EMH and KDH, and fourth for KEH, where the difference in points in the ranking is not as high as in the other cases. I consider two games of different characteristics: *Butterflies* and *Digdug*. My objective is to show the disparity between the behaviours of the agent across games when using each of the heuristics.

4.7.1 Behaviour in *Butterflies*

To win the game *Butterflies* (Section A.3), the player must capture all the butterflies before the time runs out or all the cocoons open. Catching a butterfly increases the score by 2, so the more captured, the higher the final score is. The behaviour of the agent based on the goal and heuristic that I have observed when studying each gameplay in real-time is as follows:

- WMH: At the start of the game, the agent stays around the same area and moves randomly. This is probably because the butterflies are out of reach, so the agent is not receiving any reward when looking at future states. The moment the butterflies are close enough to be noticed by the agent, it moves to catch them. When the agent gets close enough to the butterflies to capture them all, it wins the game. On the contrary, if during gameplay, the random movement of the agent and the butterflies causes them to stay afar, the agent cannot catch all the butterflies to win the game.

- EMH: The agent continuously moves until the map is completely visited. When all its surrounding locations have been visited, it moves randomly. The agent is not interested in the score or winning, so it does not care about the butterflies and ignores them. It is possible that during the exploration, the agent catches butterflies by moving to a new location. By looking at its behaviour, I can assert that the agent does not pursue the butterflies and catches them by chance.
- KDH: In *Butterflies*, there are only two interactive elements: the tree and the butterfly. I have noticed that the agent moves around the map close to the trees, interacting with them in different positions. It rarely covers the big spaces in the middle of the map unless there are butterflies around, as it tends to interact with them as well.
- KEH: Unlike in KDH, the agent does not interact with the elements based on their position. In contrast, it aims for a uniform interaction with all the elements in the game (i.e. trees and butterflies). It tends to stay in the same spot near a tree until it notices a butterfly to catch.

The demo available online at [Guerrero Romero, 2018] represents an example of gameplay of the game *Butterflies* by the RS agent with each of the heuristics. It attests to the distinctions in the agent's behaviour. At the end of the demo, the estimations of KEH for each of the elements are also shown. It predicts the score change resulted for each of them and the possible outcome (win/lose) of the game. For the trees (ID 0), it predicts no score change at all (and there is none in the game), while for the butterflies (ID 5), it predicts a score increase of 2, which is accurate.

4.7.2 Behaviour in *Digdug*

The goal of *Digdug* (Section A.8) is to collect all the items (gems and gold coins) and kill the monsters before the time runs out or the player dies. Walls are breakable, and the start point of the player is on the bottom left of the map. Figure 4.14 shows the final states of the same level of the game after the RS agent has played with each of the heuristics. By looking at them, I have noticed the existence of a difference in the behaviour. My interpretation of the behaviour of the agent by analysing the resulting final states of the game is as follows:

- WMH: The winning state is not immediately reachable, so the agent focuses on the area where the score is maximised (by the existence of gems to pick up) and cleans it. The enemies move around, and the rest of the gems are further away on the map. Therefore, states where the agent is rewarded by an increase in the score (collecting a gem or killing a monster) are out of reach. As a result, it is not able to cover the entirety of the map.

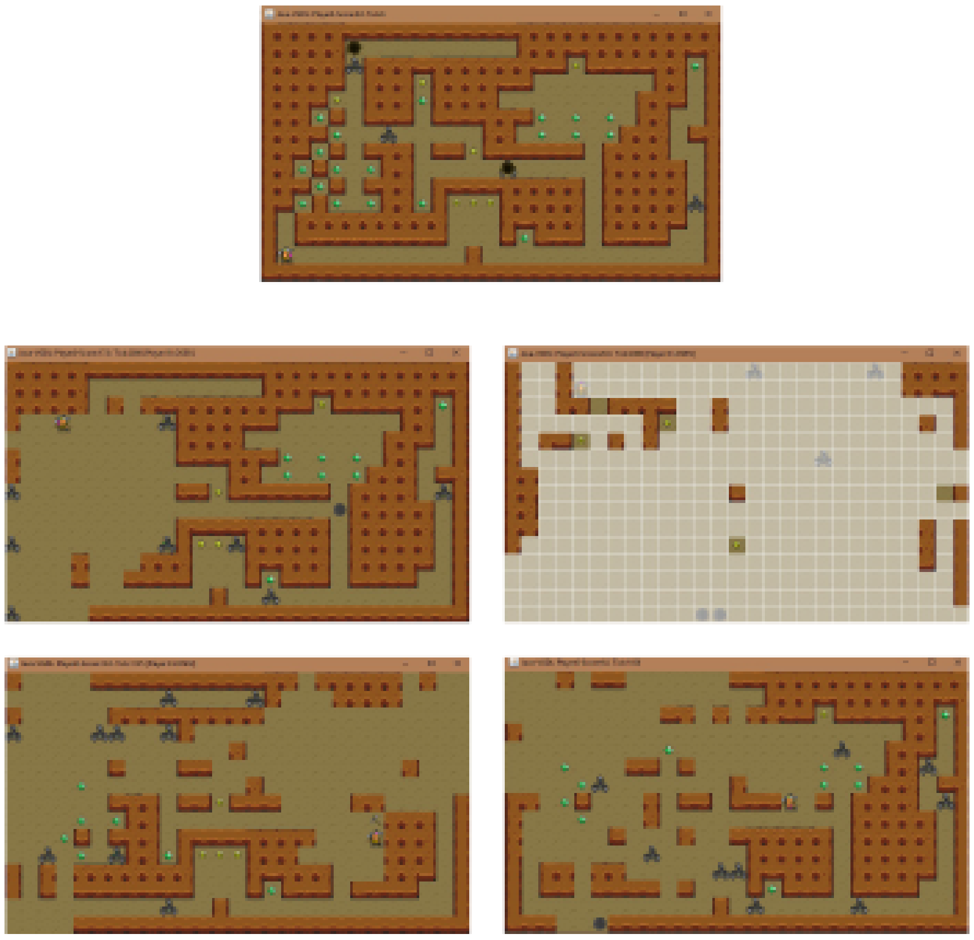


Figure 4.14: . Screenshot of the final state of the *Digdug* level (top) after RS's gameplay using each of the heuristics (in order, up to the right): WMH, EMH, KDH, and KEH.

- EMH: The agent moves around the whole map, managing to cover it almost entirely.
- KDH and KEH: The interaction with different elements (in different locations for the first heuristic) is encouraged. As a result, the agent covers the map slightly more than WMH, driven by the need to interact with the elements scattered around the map.

4.8 Conclusions

This chapter introduces the concept of *heuristic diversification*, which isolates the evaluation function from the core of the planning algorithms so the heuristic can be provided externally and be easily interchangeable. The objective of the work presented is to analyse the performance and behaviour of General Video Game Playing (GVGP) agents when the only difference between them comes from their goals (heuristics). Four different heuristics have been designed and implemented. As a result, in each set of experiments, the goal given to each of the search methods is to explore, interact, or predict rather than win the game. However, the latter is also included in one of the sets as a baseline and point of comparison. Five different controllers are employed for this study (OSLA, OLETS, OLMCTS, RHEA, and RS), and four heuristics are defined: *Winning Maximisation Heuristic* (WMH), *Exploration Maximisation Heuristic* (EMH), *Knowledge Discovery Heuristic* (KHD), and *Knowledge Estimation Heuristic* (KEH).

The essence of the search algorithms is the same, but the results of the experiments show differences based on the heuristic used. First, the performance and final ranking between the group of different agents changes depending on the heuristic assigned. Plus, the behaviour and interactions in the game for each of the given heuristics are considerably different to each other. From the heterogeneous results shown in the rankings, I deduce two important things. First, how challenging is the task of achieving a goal with a good performance for every game, independently of which this goal is, when the characteristics are unknown and generalised. Second, how the performance of a particular agent is also affected and changes depending on the heuristic used, a very interesting and thought-inspiring result.

These observations motivate different lines of research. On the one hand, this work can be taken as a first step in the possibility of enlarging GVGP techniques. These could use and combine different heuristics to gain useful knowledge about the dynamics of the game and improve the performance of the general algorithms. In addition, these agents and heuristics can obtain a relatively good understanding of the game when they play, although there is still room for improvement. This information could assist to play games better but also aid the general Procedural Content Generation (PCG) of levels and games. It could be an option for generators used, for example, in the Level and Rule Generation GVGAI competition. Last but not least, research could look deeper into the application and study of diverse behaviours. The gameplay of the agents presenting different behaviours results in stats that could resemble various types of players and show the achievement of particular tasks. These results could be used to put into practice a general evaluation of levels and games.

Enlarging and improving the winning strength of GVGP techniques by applying *heuristic diversification* is an interesting line of research. The only work I have carried out in this direction is the collaboration with Anderson et al. [2019]. The general heuristics designed were integrated (with modifications) as part of an Ensemble Decision System (EDS) created for GVGP, with successful results.

My main line of work, included in the following chapters, focuses on studying the different behaviours and looking at potential applications in games. I present an approach to elicit differentiated behaviours, identify tasks and agents' proficiency in the game and, ultimately, use the agents to assist in the game development and testing processes. GVGP agents are general and can be used with no modifications in more than one game or when changes are carried out on them, having significant potential. The first step taken, included in the next chapter and inspired by the work presented in this one, is defining a methodology that uses a *team* of general agents with different behaviours. The ultimately objective of my work is to define this *team* and use it to assist in the development and testing processes of games by playing the game automatically and facilitating the generation of various reports. The following chapters introduce proof of concept of the proposed methodology, implemented in the GVGAI Framework and applied to four games with different characteristics. The final chapter extends the application of general agents even further, proposing to use GVGP agents within games. With this objective, I include an exploratory case study that analyses the effect that general agents have on the experience of the players when they are used as NPCs in a player-vs-AI game.

Chapter 5

Vision: Use a Team of Agents for Game Development and Testing

This chapter presents the long-term vision of my research: to use a *team* of general agents with different behaviours to assist in game development and testing. Most of the material presented has been published in the paper [Guerrero-Romero et al., 2018].

5.1 Introduction

The work presented in this chapter is influenced by the results presented in Chapter 4, where I noticed apparent differences in the performance and behaviour of the same agent when provided with distinct goals and motivations. It is also inspired by the literature covered in Chapter 2, which analyses the existence of different types of players and motivations and proposes using simulations of distinct gameplay behaviours for play-testing.

The ultimate goal of my work is to assist in the game development and testing processes by facilitating developers with a method to automatically trigger tests and tools during the development of the game. In this chapter, I present the long-term vision of my research. I propose a methodology consisting of a *team* of GVGP agents with differentiated goals and behaviours to facilitate the evaluation of a game and assist in its development. In the following chapters, I will explore the first steps towards reaching this vision by defining and implementing a technical solution to generate a *team* of these characteristics and presenting a proof of concept. I use the term *team* to refer to the pool of agents because they serve the same purpose, similarly to sports, where it describes a group of people (or athletes) that train together and compete, representing the same club. While in some sports, the competition can involve the athletes playing simultaneously and collaborating (e.g. basketball); in others like fencing, the team members compete individually while still representing the club. I base my perception of the *team* on the latter, meaning that the agents of this *team* are not expected to play the game simultaneously or collaborate between them.

Games evolve during their development process, both in terms of implementation and design. New ideas are put into practice and need to be tested quickly and efficiently. Using human play-testing is a broad practice, but there is no denying that it impacts resources (human and technological), time, and money. Agent-based testing is a suitable alternative for automating the process, and there is a prolific body of work that looks at using game-playing agents to evaluate the game content, detailed in Section 2.5.2. These approaches can provide some advantages for fast and reliable testing. However, the design and implementation of these agents in some cases is very game-dependant and may not be adaptable enough to the changes designers and developers are regularly introducing. The use of general algorithms, on the other hand, provides a level of generalisation, portability, and flexibility that cannot be matched by game-specific ones. This generality, in this context, does not refer to agents that can play every kind of game in every framework, but agents that are general within a particular engine and can play any game or level supported by it. For example, the controllers and heuristics described in Chapter 4 do not use specific information about the rules of the game and, therefore, can play any of the games supported by the GVGAI Framework.

Thus, the research question (RQ2) I am looking at answering now is: *How to define, create, and use a team of GVGP agents with distinct behaviours to assist in the development and evaluation of games?*

I propose a methodology to assist in game development by facilitating the evaluation of any game using a *team* of GVGP agents with differentiated goals. It is rooted in the idea that it is possible to distinguish between different play-styles in the same game based on the motivations of the player, leading to multiple ways to explore and interact with the game. Each of the agents forming the *team* has a specific objective and skill level, which provides a flexibility that would not be possible using just one. The designer or game developer can choose the agents to run based on expected targets of performance and tasks required to accomplish. Each of the specialist agents selected plays the game under evaluation by focusing on their own goals. As a result, a logging system and two types of reports are generated. One of the reports gives information by comparing the performance of the agent with expected results. The other one provides visual feedback by producing graphs illustrating the retrieval of data during gameplay.

This chapter defines the methodology and proposes a list of agents with general objectives that can be present (or not) in a game to constitute the *team*. It also describes a possible logging system and suggests two types of reports to provide to the designer (visual and performance-target based) that can be used to validate the game and level under evaluation. The following chapters will look into defining and implementing particular pieces of this methodology as a proof of concept with the ultimate purpose of making this long-term vision a reality.

5.2 Methodology Description

I propose a methodology to assist in game development by using a *team* constituted by a series of General Video Game Playing (GVGP) algorithms with differentiated goals. Each agent plays and behaves differently within the same game. Game developers and designers can use their automated unique play-through to extract information and check whether they are on the right path or a change is needed to align with the expected outcome. In meta-heuristic approaches, a high-level heuristic is involved in deciding which agent is executed based on the state of the game [Mendes et al., 2016]. In this case, on the other hand, I intend that each of the agents of the team plays the game independently.

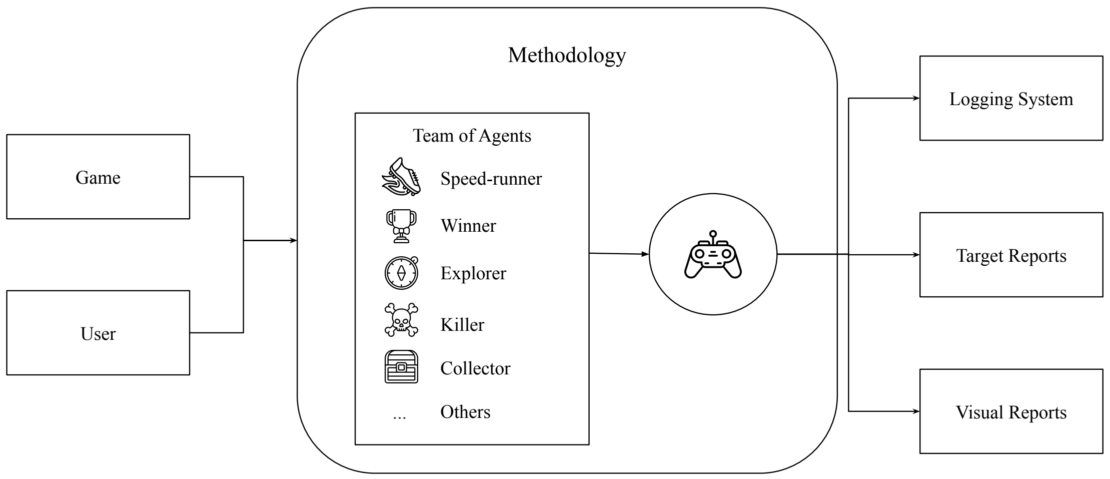


Figure 5.1: . Overview of the envisioned methodology. It requires two inputs: *Game* and *User* to generate: *Logging System*, *Target Reports*, and *Visual Reports*. These outputs are generated from the game-play of a *team* of agents.

The methodology requires a series of entities to work (Fig. 5.1): The **User** is the one responsible for the game (it can be a designer, game developer or any other interested party). They want to make sure that the content (game or level) under development fits the expectations. These expectations can refer to the design, performance, or any other characteristic of the game. The user provides the **Game** and sets up the processes required for its evaluation. Different outputs can be generated during the evaluation of the game. I propose three, two of which are a series of reports. Firstly, the **Target Reports** provide the results of evaluating the game based on the behaviours of the agents, compared to expected targets. These targets should be set before running the tests. Next, the **Visual Reports** provide visual information about the evolution of the data retrieved by the agents during their play-through. This information is presented in a series of graphs. Lastly, the **Logging System** records the logs resulting from the play-through of the agents to provide support for testing and debugging.

The main steps covered by the methodology (Fig. 5.2) are the following:

1. **Team set-up**. It is expected to have a range of agents of different types, range of

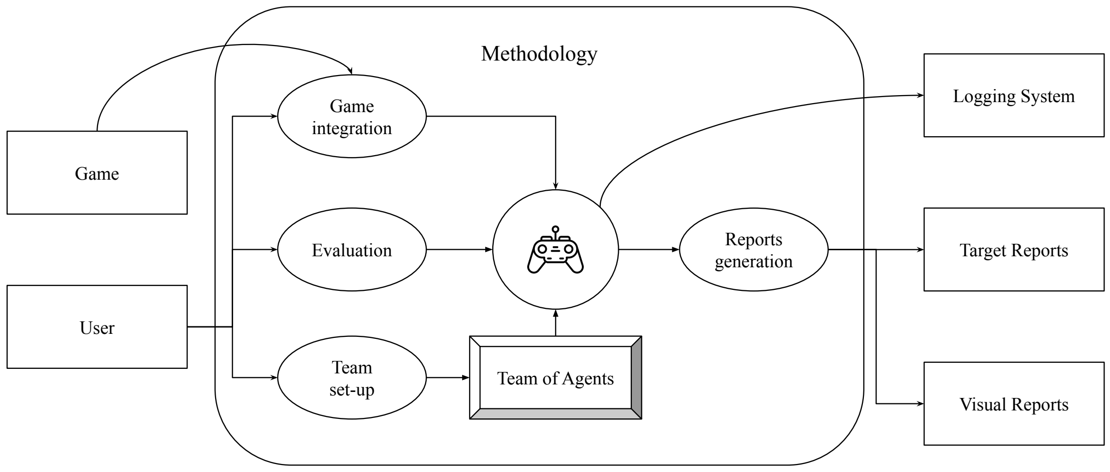


Figure 5.2: . Details of the methodology envisioned.

skills, and ways to interact and react to the game. The user can choose and optimise the ones they believe fit the characteristics of the game and the expectations of the design.

2. **Game integration**. The methodology focuses on being portable and flexible enough to apply to different games from a particular engine. However, it is needed to set the game up to run the algorithms, extract information from their play-through, and record the metrics in the logging system.

3. **Evaluation**. The agents selected by the user play the game a certain number of times. The automated gameplay of these agents logs a series of metrics and errors triggered to have detailed information about the play-through.

4. **Reports generation**. The information extracted from the gameplay of each of the agents is processed to generate reports. I propose two different types, presented in Section 5.4.2. One describes the performance of each of the agents based on their specific goals. If an expected result extracted from the gameplay is set, the error between the expected and real values obtained is calculated and reported. The other report generated is a graph that shows the evolution of the information provided by each of the agents per game-tick.

The team of general agents is meant to respond to changes and updates across multiple dimensions of the game:

Rules The base of every game. Making a change to the rules can trigger unexpected outcomes and affect the game in unanticipated ways. General agents are independent of the rules, so they do not need to be adjusted when a change happens to check that everything is working as it should be. It grants the possibility of carrying out immediate testing to detect anomalies as soon as they appear. It provides flexibility to the methodology, and it is one of the core ideas of the approach proposed.

Levels They shape how the game is presented to the player and where the action takes place. Changes in levels include increasing the difficulty, extending or reducing reachable areas, and modifying the distribution of the elements of the game, like, for example, the proportion of enemies by stages or collectable items dispersed uniformly. Reports from each of the general agents after they have played a level can provide the information needed to check that the design of the level fits the expectations. An example would be analysing the evolution of the number of Non-Player Characters (NPCs) eliminated by the *killer*, introduced in Section 5.3.6. The user should notice peaks and an abrupt increase in the numbers in those stages where a big confrontation is expected. If the play-through graph does not present those peaks, the level should be reviewed and fixed to work as desired.

Non-Player Characters (NPCs) The performance of NPCs and their interactions with the player has an immense impact on the experience. Therefore, any update on their implementation should be tested. Analysing the reports and resulting behaviour of the agents can provide an insight into the effect the changes have on the game. Examples are comparing the number of deaths versus kills obtained by the *killer* (Section 5.3.6) or the difference in the results between two *killers* with a known disparate level of mastery after a change is done to the NPCs. Similar information to the one measured for the *team* can also be logged to track the whereabouts and actions of the NPCs.

Game Parameters Even small updates in the parameters can have a substantial impact on the game. An example is updating the height of player's jump: if set to a low value, they might not be able to reach some areas of the game, affecting the exploration. Analysing the information provided by the agents can hint whether the parameters are set properly. As an example, the percentage of the exploration reached by the *map explorer*, introduced in Section 5.3.7, could increase or decrease abruptly when the height of the jump is modified.

One of the biggest strengths of the methodology I propose is that the general algorithms do not need to be modified every time any of the dimensions listed above change. If agents specific to the game are used instead, every time an element of the game changes, the controllers or heuristics may need to be updated. Another benefit is being able to use the same algorithms, without modifications, in different levels of the same game as they are created. As a result of using the general goals, the heuristics do not need to be updated to fit the specifications of a new level. It allows checking if a new level fulfils the expectations almost immediately after including it in the game. Finally, as Section 2.3 states, there are many types of GVGP algorithms, providing a wide range of options depending on the engine, technology, characteristics, and implementation of the game considered. An example is the availability of a forward model or not. Yet, if the circumstances change during the development of the game, the heuristics used could be easily transferred to another algorithm that fits the new characteristics. The use of GVGP agents does not mean that some game-specific tweaks should not be added to improve the performance of the heuristics, as long as the main general goals are not changed. The strength of the approach is based on the generality, flexibility, and robustness concepts of using General Video Game Playing, as it can adapt to significant changes in the game and its design.

Various pieces and areas of research constitute the methodology. It is impossible to cover all of them in my research and reach a final working solution. However, I develop a technical proof of concept and present an exploratory work to study the viability of the idea (Fig. 5.3). I focus on the generation of the *team* and its potential use for the evaluation of games. This work is described in the following Chapters 6 and 7, where I present an approach to generate the *team* by using *heuristic diversification* and MAP-Elites. The solution generates the heuristics for the controllers and places each agent in a behavioural space, from which it is possible to identify their proficiency. As the heuristics and controllers employed are general, they are able to play the game even if there are changes in the level or rules. However, as the ultimate goal is to use these kinds of agents for automated testing, I also include an experiment to test the portability of their strengths to unknown working levels. Finally, I include exploratory work to analyse what happens when they play 'broken' levels instead. My research is carried out in the GVGAI Framework, so it would require further research to apply the proposed approach as a tool for the games industry, but I believe it introduces a step towards tackling such a complex topic.

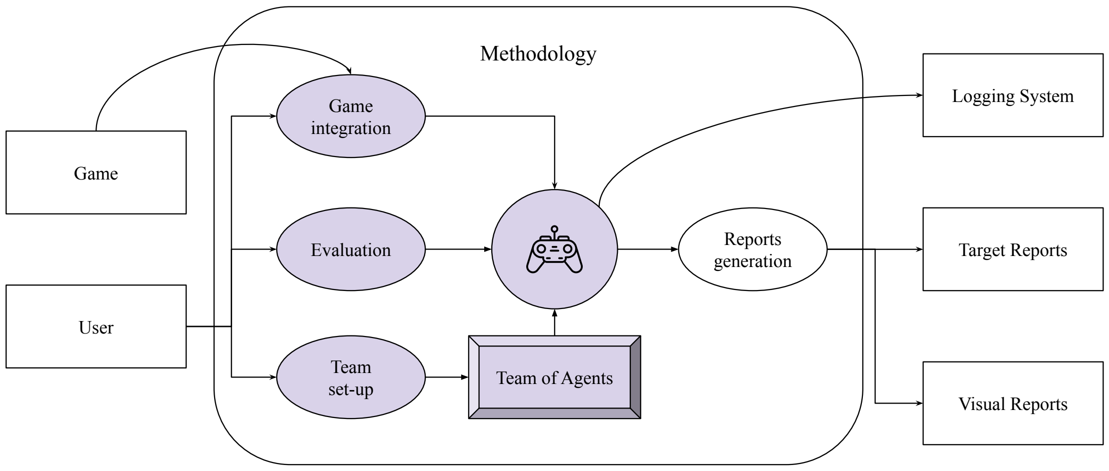


Figure 5.3: . Highlighted the areas that my research focuses on and that I cover in the following chapters. I present a technical proof of concept based on the long-term vision proposed.

5.3 The *Team*

The methodology proposes using a series of general agents with differentiated goals capable of playing a game focusing on their objectives. Each of the agents of the *team* focuses on distinct goals and excels in different tasks. This diversity provides flexibility that is not possible using just one. This section presents a series of general objectives that cover different aspects, which could be present, or not, in a game. The user can accommodate the methodology to adapt their intentions and needs by including the behaviours that fit the particular characteristics of the game. Some of the members of the suggested *team* have been inspired by the work described in Chapter 4, and others by the player-types and motivations present in the literature covered in Section 2.4.

The following is a non-exclusive list of agents proposed to constitute the *team.* I describe their goals and the data from the game that can be gathered for each of them. I use this list in my research as a reference and inspiration to either define heuristics or identify behaviours within the *team.*

5.3.1 Winner

The **goal** is to win the game and to maximise the score when a winning state is not immediately reachable. This is the most common goal in GVGP solutions.

The data gathered by an agent of this type can be the number of wins, the game ticks to victory, and the strategy followed when there is more than one option available.

I implement this objective as a heuristic in Chapter 6 and assign it as one of the *Member Goals* for the *team* generation.

5.3.2 Speed-runner

The **goal** is to finish and win the game as fast as possible.

The data gathered by an agent of this type can be the number of wins and the game ticks to reach the victory.

This proposed agent serves as an inspiration when identifying behaviours from the generated *team* in Chapter 7.

5.3.3 Survivor

The **goal** is to stay alive as much time as possible.

The data gathered by an agent of this type can be the number of wins and loses, the cause of death each time, and the game ticks to game over in each case.

This proposed agent serves as an inspiration when identifying behaviours from the generated *team* in Chapter 7.

5.3.4 Record breaker

The **goal** is to maximise the score and solve puzzles without paying attention to the chances of winning the game.

The data gathered by an agent of this type can be the number of points achieved, puzzles solved, and the game ticks required for each of these tasks.

This proposed agent serves as an inspiration when identifying behaviours from the generated *team* in Chapter 7.

5.3.5 Collector

The **goal** is to assemble the items available in the game.

The data gathered by an agent of this type can be the total number of items collected, the counts per individual type of item, and the game ticks required to collect the different objects present in the game.

I implement this objective as a heuristic in Chapter 6 and assign it as one of the *Member Goals* for the *team* generation. This proposed agent also serves as an inspiration when identifying behaviours from the generated *team* in Chapter 7.

5.3.6 Killer

The **goal** is to remove from the game as many Non-Player Characters (NPCs) as possible.

The data gathered by an agent of this type can be the number of NPCs killed, the number of times the player was killed by an NPC, counts per individual type of NPC encountered, and the game ticks required to kill all the enemies present in the game.

I implement this objective as a heuristic in Chapter 6 and assign it as one of the *Member Goals* for the *team* generation. This proposed agent also serves as an inspiration when identifying behaviours from the generated *team* in Chapter 7.

5.3.7 Map explorer

The **goal** is to physically cover the reachable areas of the level as much as possible.

The data gathered by an agent of this type can be the number of different positions of the map visited, the number of visits to each of these locations, the total percentage of the map explored, and the game ticks required to finish the exploration.

The existent work that includes the implementation of an exploratory heuristic in planning can be used as an example or inspiration to create an agent of this type [Perez Liebana et al., 2015, Nielsen et al., 2015], as well as the EMH implemented in Chapter 4.

I implement this objective as a heuristic in Chapter 6 and assign it as one of the *Member Goals* for the *team* generation. This proposed agent also serves as an inspiration when identifying behaviours from the generated *team* in Chapter 7.

5.3.8 Novelty explorer

The **goal** is to go through as many different game states as possible and to provide this number as a result. It is an alternative for an exploratory agent that considers states instead of locations.

The data gathered by an agent of this type can be the number of different states visited.

This type of agent is related to the *novelty* appraisal frequent in intrinsic motivation [Roohi et al., 2018]. The inspiration for this kind of agent also comes from the work done by Bellemare et al. [2016]. The authors proposed connecting the information gained through the learning process and count-based exploration, which guides the behaviour of the agents to reduce uncertainty. This approach is designed to explore the environments more practically and efficiently.

5.3.9 Competence seeker

The **goal** is related to the amount of information the agent is capable of collecting when a series of actions are performed.

An agent of this type can provide information about the level of expertise gained during its play-through.

It is based on the model of *empowerment* of intrinsic agents, which denotes the degree of control the agent feels having over the environment [Roohi et al., 2018].

5.3.10 Curious

The **goal** is to interact as much as possible with the elements of the game. The agent prioritises those elements that have not been interacted with before and in different locations.

The data gathered by an agent of this type can be the number of elements interacted with, the events triggered when these interactions happened, the locations of the map where the interactions happened, and the number of game ticks required to interact with the different elements of the game.

This type of agent is inspired by the concept of *curiosity* introduced in the development of the Knowledge-based MCTS [Perez et al., 2014], and applied in the design of the KDH and KEH for the *heuristic diversification* experiments described in Chapter 4.

I implement this objective as a heuristic in Chapter 6 and assign it as one of the *Member Goals* for the *team* generation. This proposed agent also serves as an inspiration when identifying behaviours from the generated *team* in Chapter 7.

5.3.11 Scholar

The **goal** is to learn the outcome of the actions available and to obtain as much knowledge about the game as possible.

The data provided by an agent of this type can be the percentage of accuracy of the knowledge gained during gameplay. The generality of this type of agent is improbable because it is needed to have concrete information about the rules and outcomes of the interactions with the game to be able to check the quality of the predictions. However, an agent with this kind of objective is an interesting addition to the *team* as it can be used to detect anomalies during gameplay. There is a high chance that an agent focused on this kind of task finds unexpected rules or bugs on the existent ones that need to be fixed.

This kind of agent is based on the KEH designed and implemented in Chapter 4.

5.3.12 Search space scholar

The **goal** is to learn how to get rewards from the game by building a graph containing this information. So, ultimately, the agent is able to navigate the search space and efficiently move from one state to another based on its motivation.

The data provided by an agent of this type can be the map of states created during its execution, detailed information about the learning graph built, and the strategies followed to go through different states.

This agent is inspired by the use of Influence Maps in GVGP [Park and Kim, 2015]. Although an agent of these characteristics seems to have similarities with *Scholar*, there are key differences between them. The goal of *Scholar* is to mainly learn the outcome of the actions to get information about the rules of the game and obtain accurate predictions of the different events at the end of the game. *Search space scholar*, on the other hand, uses the knowledge to build a *learning* graph while playing the game, which, at the same time, is used to explore the search space profitably.

5.3.13 Risk analyst

The **goal** is to analyse the level of risk during the play-through and take actions to maintain it at a certain level chosen by the user. A low-risk agent would tend to avoid situations where the chances of losing the game are high, like bumping into a hoard of enemies or complex areas. A high-risk agent would gravitate towards the opposite and jump into dangerous situations.

The data gathered by an agent of this type can be the risk percentage predicted at every moment, the number of deaths, NPCs killed, obstacles overcame, and the game ticks until losing the game.

5.3.14 Semantic

The **goal** is to focus on tasks related to linguistics, as coaching the dialogue of the game or making sure the narrative flows and is consistent.

The data gathered by an agent of this type can be the estimated quality of the dialogues, the number of possible outcomes depending on the choices, and the level of consistency of the story.

5.4 Game Evaluation

The agents selected by the user are employed to trigger automated gameplays of the game or level for its evaluation. These gameplays allow to extract information about the game and gather stats related to the behaviour of the agents, producing high-level reports and raw logs provided by a logging system.

5.4.1 Logging system

The *Logging system* keeps track of the information resulting from running each of the agents: location in the map per game tick, list of the actions carried out, elements interacted with, responses triggered, etc. These logs can help to detect anomalies and broken states of the game, even when the development is at an early stage.

As described in Section 2.4.2, in games, it is common to collect metrics from the players. Thus, it should be possible to define a similar approach for the GVGP agents and gather data from their play-through. There is an active line of research looking into defining agent metrics that can be applied to the methodology. Firstly, Nelson [2011] introduces seven strategies to extract information from the game artefact that could be included in the logging system. In addition, Volz et al. [2018] propose the creation of a framework to log games information by gathering general measures previously used to describe gameplay by extracting information from it. They differentiate between the following types of loggable measures: agent-based, interpreted features, direct logging features, and general indirect features. Lastly, the guidelines for a framework of these characteristics are formalised and presented in [Volz and Naujoks, 2020]. The authors detail different benchmarks to use to understand the behaviour of game-playing agents across different games. This generality makes it possible to apply to different environments and general video game playing solutions, and, therefore, it is applicable to this methodology as well. I argue that having a logging system gathering the metrics of game-playing agents and a *team* of agents with contrasting behaviours allow for covering several game states, extracting very diverse information from the game. This diversity grants the ability to trigger and detect errors in the game that may be overlooked otherwise.

5.4.2 Reports

I propose the generation of different types of reports to check the validity of the design of the game. These reports are directed to give detailed information and highlight issues by using the data gathered during gameplay.

Performance-target based reports These reports assist in the evaluation of the game based on expected performances in the resulting stats of the agents.

In the experiments presented in Chapter 4, the results for a similar controller-heuristic pair showed a clear distinction depending on the type of game. I take as an example the results obtained using the Exploration Maximization Heuristic (EMH), focused on maximising the exploration of the level and detailed in Section 4.3.2. In completely accessible levels in games like *Butterflies* and *Chopper*, most of the agents using the EMH obtained an average percentage of performance higher than 90% (Tables C.23 and C.26). Whereas, in games with large maps or where a series of steps are needed to unlock the access to the different areas, the exploration was lower. In *Roguelike*, none of the agents got an average higher than 42.22% (Table C.37). The existence of differences in the resulting performance of particular tasks in the game can be used to the developer's benefit by employing an estimation of the performance that agents should achieve based on the type of game and level under consideration. Before starting the evaluation of the game, the user can choose the agents of the *team* that are considered appropriate and set an estimated desired performance for each of them. The results obtained during gameplay inform whether there is an agreement between the expected values and the reported ones. The recommendation is to play the game several times to avoid outliers in the data and assure the result is a valid representation of the behaviour of the agent. The errors can be calculated by the difference between the targets and the real values. This information makes it possible to check if the design matches the expectations or how distant the values are.

For example, suppose a level is designed to be easily accessible but challenging to win. In that case, the user could assign a high desired value to the exploration reported by the *map explorer* and a low value to the percentage of wins reported by the *winner*. Any discrepancy in the results would inform that there is a mistake in the design and that it should be reviewed.

Visual reports These reports are intended to be easily interpreted by the user in the form of graphs. These graphs are meant to provide an analysis of how the information retrieved by the agents evolves during their play-through. This information can be the number of different positions or states visited, the number of elements interacted with, the number of enemies killed, etc. The user can deduce and conclude interesting information about their game and level by analysing the shape and evolution of the plotted values. A continuous trend means that the agent is capable of getting information without impediments, improving uniformly. On the other hand, if the growth is stuck for a period of time, it means that either there is nothing more to be discovered, all the goals of the agent have been reached, or there is an obstacle (or a series of barriers) preventing the agent from achieving them.

Fig. 5.4 represents a fictional (simplified) play-through graph obtained for the *Map explorer* as an example. The report shows a constant growth to a certain point, where it keeps still for a while, and ends up increasing uniformly again. This shape could be interpreted as follows: the level is divided into two areas, and a particular action from the player is required to progress in the game.

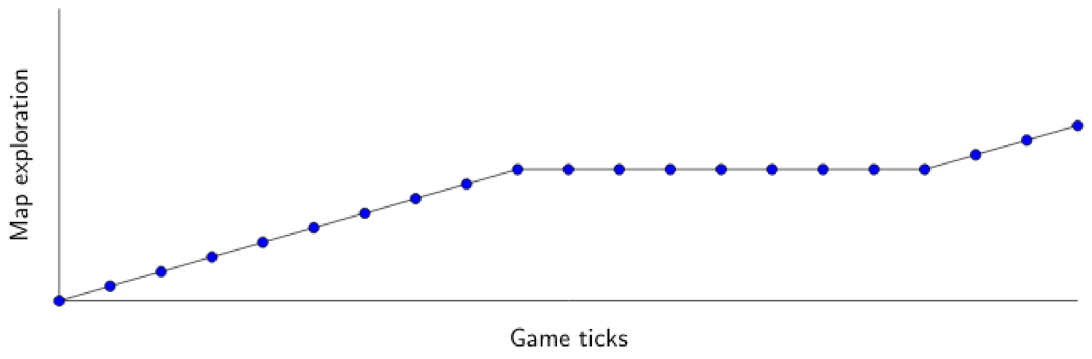


Figure 5.4: Example of a potential visual report generated for a play-through of the *Map Explorer*.

These visual reports can also be used to analyse the distribution of different elements in the level. Fig. 5.5 shows a potential simplified play-through for *Collector* as an example. The growth of the chart shows peaks when the agent visits those areas where there are several items to be collected.

5.5 Variations

The methodology proposed is flexible and open to extensions, so I include some possible variations that are worth considering:

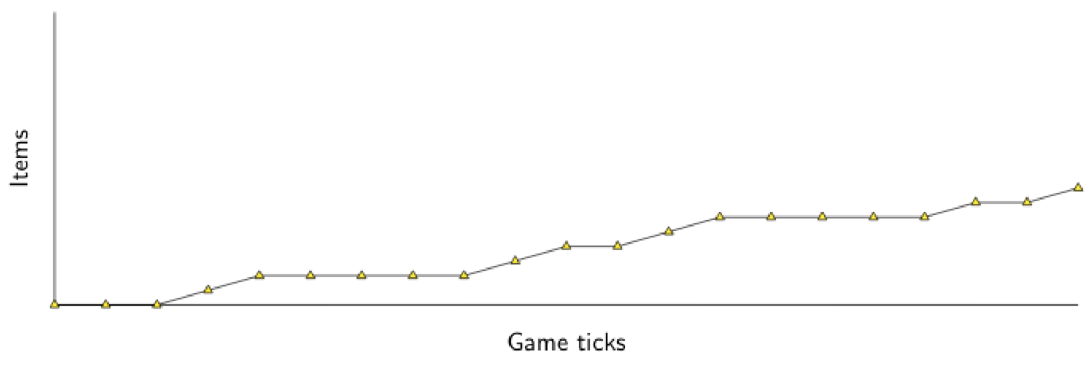


Figure 5.5: Example of a potential visual report generated for a play-through of the *Collector*.

Diversity in skill levels The same algorithms with different parameters have different strengths. The *team* can include several versions of the algorithms with the same objectives but different levels of mastery, based on those parameters.

Granularity Another possible extension is considering the information retrieved by all the agents as a whole and studying the correlations between them. The user can choose and combine data provided by the agents to obtain even greater levels of granularity.

5.6 Limitations

The methodology proposed has an evident strength. However, there are a series of limitations related to the technique, GVGP, and the algorithms themselves that I discuss in this section.

Game complexity The time required to evaluate the game needs to be considered to arrange enough time to analyse the reports and plan the necessary actions to be taken based on the results. The higher the complexity of the game, the longer the evaluation takes, as the agents need more time to run and finish the play-through and provide feedback. A feasible solution would be presenting the game split into stages or levels and analysing small pieces at a time. Moreover, the complexity of the game affects the performance of the algorithms. The use of general objectives come with some limitations when facing complex environments.

Optimisation of the *team* The *team* should be well-tuned to allow the agents to recognise and carry out the actions to reach their goals and obtain results that fit the expectations. This tuning lets the user interpret the feedback accordingly.

Reinforcement Learning (RL) The main limitation of Reinforcement Learning (RL) algorithms is their requirement of computer power, which increases with the complexity of the problem. These algorithms require offline training, and their performance depends on the size and intricacy of the system. The higher the game's complexity, the longer time is needed to train the agents, which translates to higher computational power that is not within everyone's reach. Nevertheless, a long training time does not mean that the learning process is successful, and it is possible that even if having *infinite* computational power, the agents' performance is not as expected. This statement is especially true for games and environments with large branching factors, games that require multiple levels of abstraction and reasoning, or those that include real-world features that limit the application of these techniques. Examples of these are the existence of a continuous state and action space, stochasticity, partial observability (e.g. the fog of war present in multiple strategy games), and multi-agent systems.

Search algorithms Planning algorithms require a forward model to simulate possible future states and take the best action available. Hence, the limitation lies in the challenge of creating a forward model from scratch to include it in the game or the engine used for development. When a forward model is not available, the agents need to work with abstract ones or approximations instead, impacting their performance. Moreover, the budget of processing time and resources impact the performance of these algorithms. The allocated time affects the number of roll-outs available per turn for tree-search algorithms and the number of individuals for genetic algorithms [Perez et al., 2013, Nelson, 2016]. The higher the number of simulations, the more information they get to make decisions.

Parameter optimisation Tree-search and evolutionary algorithms use a series of parameters that have a big impact on their performance and behaviour and, in most cases, need to be optimised. It is especially important in GVGP, where the robustness of the algorithm can benefit its overall performance. As mentioned above, if the number of roll-outs available to the search algorithm is modified, the number of predictions will be reduced or extended. As a result, the information available to make a decision will be affected, influencing their actions and outcomes. In evolutionary algorithms, the size of the population and other parameters present in their definition impact the performance. Therefore, the user needs to find the best set of parameters for the characteristics of the game under evaluation, which can take time. If not enough time is allowed for this optimisation, the expected performance of the agents can suffer a decline.

There is an active body of research looking into the limitations given by parameter optimisation. This optimisation is usually done offline to provide enough time to reach the desired level of performance. However, there has been some recent progress in the area. An example is the implementation of an online adaptive parameter tuning mechanism in GVGP, with promising results [Sironi et al., 2018]. Moreover, the N-Tuple Bandit Evolutionary Algorithm (NTBEA) shows ways to mitigate some of the limitations. Lucas et al. [2018] define NTBEA as a simple, informative, and efficient model capable of being applied to numerous optimisation-related problems. In their research, they use it to optimise the parameters of RHEA.

The challenge of GVGP The task of developing algorithms capable of working through different games is challenging, as it is not possible to use game-specific information to guide them. Given the difficulties of the problem, several approaches have been created and are being investigated to tackle it, so GVGP is still ongoing research. Even considering the latest improvements, the results of the GVGAI Competition show how none of the algorithms is yet good enough to generalise to every kind of game [Perez-Liebana et al., 2019b]. No controller performs uniformly well across all games, as even when the agents perform well in some games, they still show a low percentage of success in others. Furthermore, the variety of the problems increases the complexity of the generalisation, as general algorithms can be applied to several areas in games: from simple single-player games to multi-player collaborative ones, where they need to work together to achieve a common objective. The game's complexity also affects the performance of the algorithms, as general AI has limitations in solving complicated environments. The methodology presented in this chapter would be strengthened if the limitations related to the area of GVGP should be minimised and, therefore, I encourage the research community to tackle them.

5.7 Conclusions

This chapter proposes a new methodology to use General Video Game Playing (GVGP) agents to assist in the game design and testing processes and describes its features. It presents a series of differentiated goals that can be applied to the general agents to play a game in different ways. Having agents focusing on targets beyond simply winning the game leads to specialists with distinct gameplay styles that achieve different sorts of tasks, being able to extract varied information from the game. I propose generating two types of reports and a logging system using the data and metrics gathered from the agents' play-through. These can help to review if the game fulfils its expectations and to identify issues in the early stages of the development. The information retrieved can be used to detect bugs, balance the game, or tweak its parameters. The independence of the rules given by the generality of the agents allows an early integration of the method in the game development process. The approach does not require major modifications when the game is extended, modified, or when new levels are integrated into the game.

The proposal is rooted in previous work in GVGP, automatic play-testing, and AI-assisted game design. It considers the needs of the video game industry for efficient and accurate game testing and highlights interesting areas of open research. Several extensions in the methodology are possible: including agents with different levels of skill, players tackling multiple objectives, or adding collaborative and social-oriented profiles that can fit multi-player games. Moreover, the reports generated can be extended to consider the objectives of the different specialists at once. It should be possible to combine the results obtained to analyse the information produced by multiple agents, study their correlations, and provide a greater level of granularity. I believe that integrating general algorithms in the game development process provides portability that is non-existent in the current approaches. The methodology presents the developers with the option to choose between several algorithms with differentiated behaviours and skills. It provides the flexibility to adapt to the characteristics of the game under evaluation and its modifications.

The methodology proposed is constituted by separate pieces that cover diverse research areas and open different lines of work. In the following chapters, I focus on a portion of it: the area related to eliciting diverse behaviour and automated gameplay (Fig. 5.3). My work does not look into formalising and detailing the metrics obtained from the agents' play-through, so it does not cover the logging system or the generation of the reports. In the following chapters, I investigate the definition of the *team* and the identification of different types of behaviours and tasks within it. I present an approach to generate general agents with differentiated behaviours and implement it in the GVGAI Framework. I apply the procedure to numerous games with different characteristics to verify its flexibility and adaptability. I also determine how to identify various types of agents that accomplish particular tasks and analyse the portability of their strength to levels with different characteristics. Lastly, I discuss the strength of the methodology and its application in game development and testing with a preliminary work that explores the employment of the agents identified to test 'broken' levels. In summary, the following chapters present a technical approach as a proof of concept of the vision, with the ultimately objective of making this methodology applicable in the future.

Chapter 6

Approach: Generate the Team and Elicit a Diverse Gameplay

This chapter presents an approach to generate a *team* of GVGP agents with distinct behaviours. The role of each of these agents is identified by their location in a behavioural space. The agents generated can be used for automated gameplay and have several applications in game development and testing. Most of the material presented has been published in the paper [Guerrero-Romero and Perez-Liebana, 2021] and included in a journal paper submitted to IEEE Transactions on Games that is currently under review.

6.1 Introduction

The work presented in this chapter builds from my previous one presented in Chapters 4 and 5. In Chapter 4, I introduced the concept of *heuristic diversification* in General Video Game Playing (GVGP). I provided GVGP agents with four heuristics that elicit different goals: *winning*, *exploration*, *knowledge discovery*, and *knowledge estimation* and compared their performance when taking as reference features related to each of them. The core of the algorithms was unchanged as the evaluation function was isolated and provided externally. The results showed how 1) the performance between a set of controllers changed depending on the heuristic assigned to them, and 2) the behaviour and interactions in the game for each of these given heuristics were considerably different to each other. These observations inspired me to outline a methodology that uses a *team* of GVGP agents with distinct behaviours to assist in game development and testing (Chapter 5). The generation of a team of these characteristics is the focus of the work presented in this chapter. I define and implement an approach to generate the proposed diversity of behaviours for the agents and integrate it into a framework to assemble a *team* of agents in different games (Fig. 6.1).

I am still looking at answering RQ2: *How to define, create, and use a team of GVGP agents with distinct behaviours to assist in the development and evaluation of games?* However, in order to answer this question, I first need to go through a series of steps in detail and resolve them:

1. *Which general approach, applicable between games of different types, allows to create and provide a team as the one envisioned?*

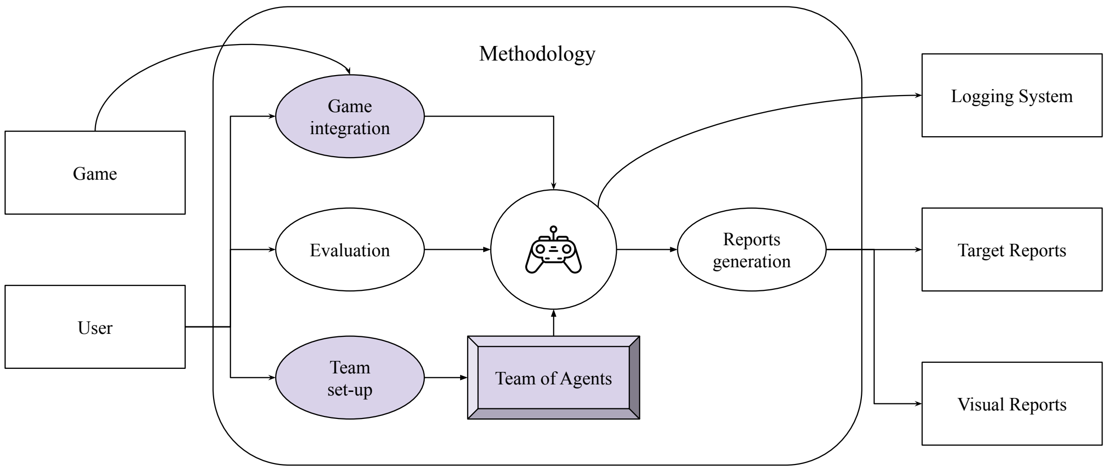


Figure 6.1: Highlighted the areas covered in this chapter from the long-term vision. I present an approach to set up (generate) the *team* and integrate it within the framework and games used.

My proposal requires a diverse range of game-playing agents based on their behaviour and ways to interact with the game. The MAP-Elites (Section 3.3) allows to generate elements in a feature space and, therefore, can provide the diversity of behaviours needed.

2. *How can I define the behaviour of each of the agents so it is easy to describe and generate?*

   For the MAP-Elites, it is necessary to define a series of elements, including the candidate to generate and evolve. This candidate needs to represent the behaviour while being easy to evolve and assign to an agent. The solution lies in applying *heuristic diversification* and defining a heuristic that allows combining a list of different goals with a set-up that fulfils these requirements.

3. *How do I generate these agents using the MAP-Elites?*

   I need to define the distinct elements required for the MAP-Elites and integrate the algorithm in the GVGAI Framework so it can be used for my experiments. The algorithm may require some tweaks to adapt to my needs, so I describe these in detail.

4. *Is this solution general and applicable to games with different characteristics?*

   Once the implementation and integration with the GVGAI Framework are completed, I set up the approach to generate a *team* of agents for several games with different characteristics. I then analyse and discuss the results.

This chapter covers each of the steps listed above. It details the approach and presents the experiments I carry out to apply it in the GVGAI Framework to four games with different characteristics. It also includes a discussion about the results obtained, limitations, extensions, and future steps.

6.2 Definitions

I define the concepts used in this chapter as follows:

Team I use this term to refer to the pool of agents.

Goal The ultimate objective(s) of one of the agents in the *team* when playing the game. Examples of goals are exploring the level, collecting items, or killing enemies.

Heuristic value Result of the evaluation of a game state. It is given in relative terms of an arbitrarily high value denoted as H. High and low rewards values are indicated with H^+ and H^- respectively. For the experiments included in this chapter, the value of H is assigned independently by each of the heuristics, so it varies.

Behaviour The way the agent ultimately reacts and interacts with the game. I consider the agents' behaviour as the results of their play (gameplay stats and features).

Event Effect of two sprites being in contact during the execution of the game, being one of the entities involved the *avatar* (player) or an element generated from it. I also refer to it as *Interaction*, and I differentiate between two types: *Collision* and *Hit*.

Collision A particular case of interaction where the entity involved is the player. An example of this kind of interaction is the player collecting a coin.

Hit A particular case of interaction where the entity involved is an item generated by the player. The item has typically been previously generated as a result of the execution of an action of type ACTION. An example of this kind of interaction is a bullet shot by the player hitting an enemy.

Curiosity An interaction between sprites that happens in a position of the map where it has not taken place before.

6.3 Overview of the Approach

I present a procedure to generate a *team* of agents for a game so they are available to the developer to choose from and that can be used for automated gameplay. My solution applies the MAP-Elites algorithm to generate agents with distinct behaviours. The resulting agents are distributed in the space of features based on the result of their actions when playing a game: *wins*, *score*, *% explored*, *interactions*, *kills*, *items collected*, etc. The criteria used as the performance of the elites does not come from how *well* an agent plays a game, but by the time it takes to reach the End of Game (EoG). Thus, given two agents with similar resulting stats, one is considered better than the other if it manages to reach a game over faster. I present and implement the solution and include details about the agent used, as well as the list of heuristics created to represent differentiated goals within the game: *Winning and score*, *Exploration*, *Curiosity*, *Killing*, and *Collection*. I integrate the algorithm with the GVGAI Framework and execute it for four games and a variety of pairs of features. The final *team* of agents is assembled from the collection of maps generated for each game. The size and diversity of the pool of agents generated allow running automated gameplays eliciting different expected behaviours. The options are limited by the distribution of the team within the space, given by the pair of features or the characteristics of the game. The methodology gives the flexibility to extend the features or modify the range of existing ones to have control over the behavioural space and, therefore, the characteristics of the *team* generated.

6.4 Defining Agents with Behaviours Easy to Generate

The objective is to create and have available a range of agents with differentiated behaviours and identifiable tasks. These behaviours should be provided by their heuristics and not the parameters of the controller, so they can easily vary without having to make updates to their core. I introduce: 1) an agent with interchangeable heuristics; 2) a heuristic called *MemberBehaviour* designed to be plugged into the agent externally and that allows combining different goals; 3) the use of the MAP-Elites algorithm to generate the distinct behaviours of the agents.

6.4.1 Agent: OLMCTS with an interchangeable heuristic

The work presented in this chapter uses a unique controller: the *sampleMCTS* algorithm (Section 3.2.2) provided by the GVGAI Framework, but modified and extended so that the heuristic can be assigned externally. I chose to use OLMCTS because it had a stable mid-range performance with all different heuristics (Table 4.21).

I follow a similar idea to the *heuristic diversification* presented in Chapter 4 in terms of isolating and extracting the evaluation function so it is not tight to the core of the single-player search algorithm. However, in this case, I do not use an accumulation of the heuristic to obtain the final reward. The evaluation of a state comes from comparing the information of the game in the current state and the final state reached with the forward model. Therefore, it is necessary to keep track of the data in each of the future states to use it in the calculation of the reward (Fig. 6.2). The temporary information stored about the future states visited with the forward model depends on the heuristic and its characteristics. For example, the positions visited by the avatar, the number of enemies killed, or the events triggered in each of the states simulated with the forward model.

In this case, the solution cannot simply define a heuristic per goal and swap between them as in Chapter 4. I need to find a solution that combines various goals to produce and have a diverse range of behaviours at my disposal. Hence, I define a *parent* heuristic called *MemberBehaviour*. It is the external heuristic I provide to the controller (OLMCTS

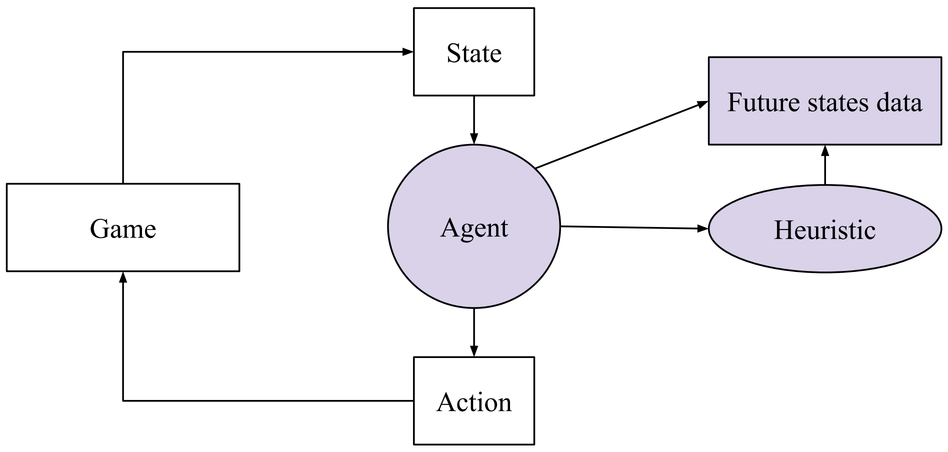


Figure 6.2: Agent with interchangeable heuristic: by *heuristic diversification*, the evaluation function is provided externally. It stores data from the simulated states reached with the forward model and uses it to obtain the heuristic value.

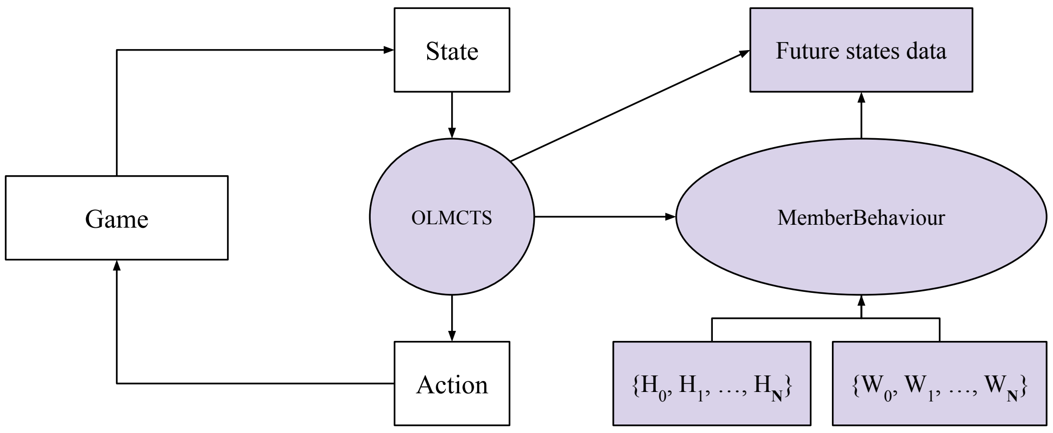


Figure 6.3: OLMCTS agent with *MemberBehaviour*, which allows to provide a list of goals (heuristics) externally and assign a *weight* to each of them.

in this case) and that can receive a list of heuristics and weights (Fig. 6.3). It is described in detail in the section below.

6.4.2 *MemberBehaviour*

I need to provide more diversity to the agent than the one given by simply using one static heuristic or swapping between them. As a result, I create and describe a *parent* heuristic that allows combining different independent goals, called *MemberBehaviour*.

Member goals $\{t_0, ..., t_m\}$ being m the total number of goals available to gather stats about the gameplay of the agent. Each goal collects information related to its characteristics.

Enabled heuristics $\{h_0, ..., h_n\}$ $n <= m$ is the total number of enabled goals, taken from the available ones. Only the enabled goals take part in the evaluation of the state and, therefore, in the calculation of the final heuristic.

Weights $W = \{w_0, ..., w_n\}$ n is the number of enabled heuristics, as each weight is assigned to one of them. The weight gets a value between $[0.0, 1.0]$. This value determines the importance that the corresponding goal is given in the final calculation. Therefore, W ultimately describes the final behaviour of the agent. It is easy to define, generate, and evolve, so it can be used as a candidate in MAP-Elites.

The list of enabled heuristics and the weights assigned to each of them define the behaviour of the agent as they ultimately drive its actions. The final value of the heuristic resulting from evaluating a particular state comes from combining all the enabled ones. The result of each independent heuristic can be in different ranges, so they must be normalised to make sure they are within the same bounds ($[0.0, 1.0]$). The heuristic obtained for each goal is multiplied by its corresponding weight. I get the final heuristic by adding all of these results:

$$H(S) = h_0 w_0 + ... + h_n w_n$$

Fig. 6.4 shows an example of the *MemberBehaviour* with 5 goals available but only 4 of them set as enabled heuristics.

6.4.3 Using MAP-Elites to generate the team

In my approach, I identify the elements required for the MAP-Elites algorithm (Section 3.3) as follows:

Genotype x Vector of weights $W = \{w_0, ..., w_n\}$. The description of this vector is easy to define and, therefore, to generate and evolve. It represents the presence of each of the *enabled* heuristics when assigned to the agent, defining its behaviour, and can be used as a candidate.

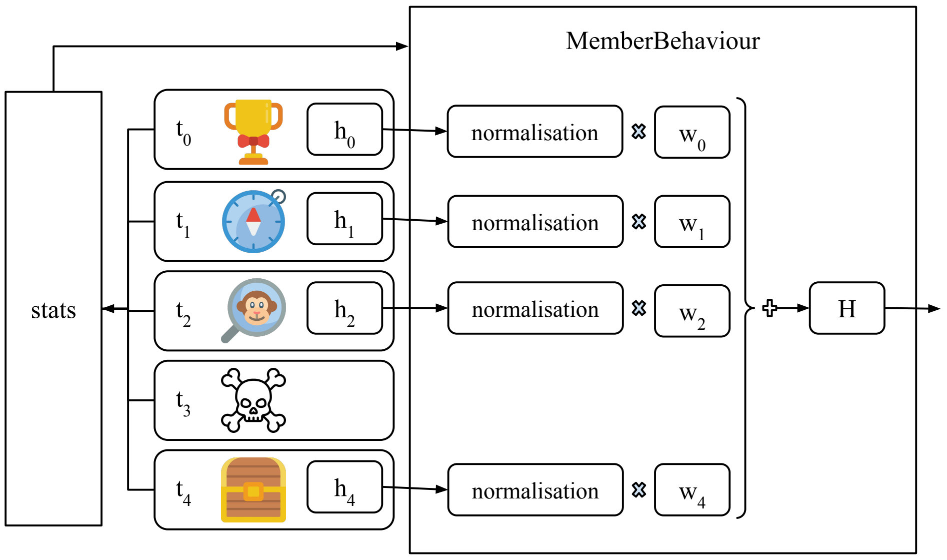


Figure 6.4: Example of *MemberBehaviour*. There are 5 goals but only 4 of them are enabled and take part in the calculation of the heuristic H.

Phenotype p_x Stats generated when providing the agent with W and playing the game several times.

Feature function b_x Characteristics of the gameplay of the agent, taken from the stats: wins, score, exploration percentage, interactions, etc. These illustrate the results from the behaviour of the agent when playing the game, and it is the information I am interested in to get a diverse team. These features are defined by the members and are dependent on the game and its characteristics. Although MAP-Elites supports an N-dimensional feature space, I focus on two-dimensional maps. I consider only pairs of features for simplicity in the setup and readability of the results.

Fitness function f_x The performance of an agent is measured by how quickly the end of the game (EoG) is reached. I establish that between two agents with a similar set of features (stats), one is better than the other if the game ends earlier; i.e. they managed to reach similar results in less time.

Therefore, the relationship between *genotypes*, *phenotypes*, *features*, and *performance* of a candidate existing in the MAP-Elites, applied to my approach, is as follows:

$$W \rightarrow stats \rightarrow features, EoG$$

The pseudocode of this application of the MAP-Elites is included in Algorithm 9.

The initialisation of the MAP-Elites is done in two steps. First, I get a series of candidates eliciting just one of the goals enabled, i.e. the weight assigned to the corresponding heuristic is set to 1.0 while the rest are assigned to 0.0. After this, the second initialisation step generates a series of random candidates (weights are given random

Algorithm 9 Use of the MAP-Elites algorithm to generate the *team*.
Nomenclature: $X \leftarrow$ solutions (map of elites); $P \leftarrow$ solutions' performances; $x \leftarrow$ elite; $W \leftarrow$ agent description; $x' \leftarrow$ candidate solution; $b' \leftarrow$ feature descriptor of x'; $p' \leftarrow$ performance of x'; $\alpha \leftarrow$ *nEnabledHeuristics*, $\beta \leftarrow$ *nRandomInitialisations*.

```
procedure MAP-Elites-Team
    $X \leftarrow MAPElitesInitialisation()$
                        ▷ add to the map $\alpha$ elites with only each of the goals enabled
                        ▷ generate $\beta$ elites with random weights and add them to the map
    for $iter = 1 \leftarrow nAlgorithmIterations$ do
        $x \leftarrow random\_selection(X)$
        $W \leftarrow behaviourWeights(x)$
        $W' \leftarrow evolution(W)$        ▷ evolve elite's weights to generate a new candidate
        $x' \leftarrow createGameplayElite(W')$        ▷ the agent plays the game $nGameRuns$
        $b' \leftarrow x'.featureStats()$        ▷ candidate's stats of the map features
        $p' \leftarrow x'.performanceStats()$        ▷ candidate's stats of the performance criteria
        if $X(b') = \emptyset$ or $p' > P(b')$ then
            $P(b') \leftarrow p'$
            $X(b') \leftarrow x'$
    return $X$, $P$
```

values between $[0.0, 1.0]$). In both cases, the agents are assigned to their corresponding cells. These two steps expect to 1) provide a baseline of behaviours by using each goal independently, and 2) provide a baseline of diversity given by the random assignments. The algorithm then starts its iterations until a certain number is reached, provided as an algorithm parameter in the configuration. In each iteration, a cell is selected uniformly at random. The weights of the elite occupying the cell are evolved with a simple mutation hill climber: one of the weights is randomly selected and updated to a new value between $[0.0, 1.0]$. This mutation generates the behaviour description of a new candidate, which is assigned to its corresponding cell. This simple evolutionary method proved enough to generate a good range of different behaviours.

The assignment, both during initialisation and during the main execution of the algorithm, works as follows. First, the weights are assigned to the agent and it plays the game several times to obtain its stats. Then, these stats are used to get the features that constitute the map and assign the candidate to its corresponding cell. The map is divided into a fixed number of cells given by the two feature dimensions. Each of the features is assigned a *minimum*, *maximum* and *bucket size* value, so the resulting value obtained by the agent is assigned to the bucket that contains the range it belongs to. If the corresponding cell is empty, the agent is directly assigned to it, and the algorithm moves to the next iteration. However, if the cell is occupied by an elite, the performance of both candidates are compared. If the new candidate has better performance than the existing one, it replaces the latter. When the algorithm concludes, the map contains a set of descriptions (W vectors) of a diverse range of agents. Their behaviour is implied by their location and set of features assigned in the aforesaid map. This group of agents is what I call a *team*. An example of the map generated is included in Fig. 6.5. If various maps are generated for the same game and enabled goals, the *team* is constituted by the ensemble of the different maps.

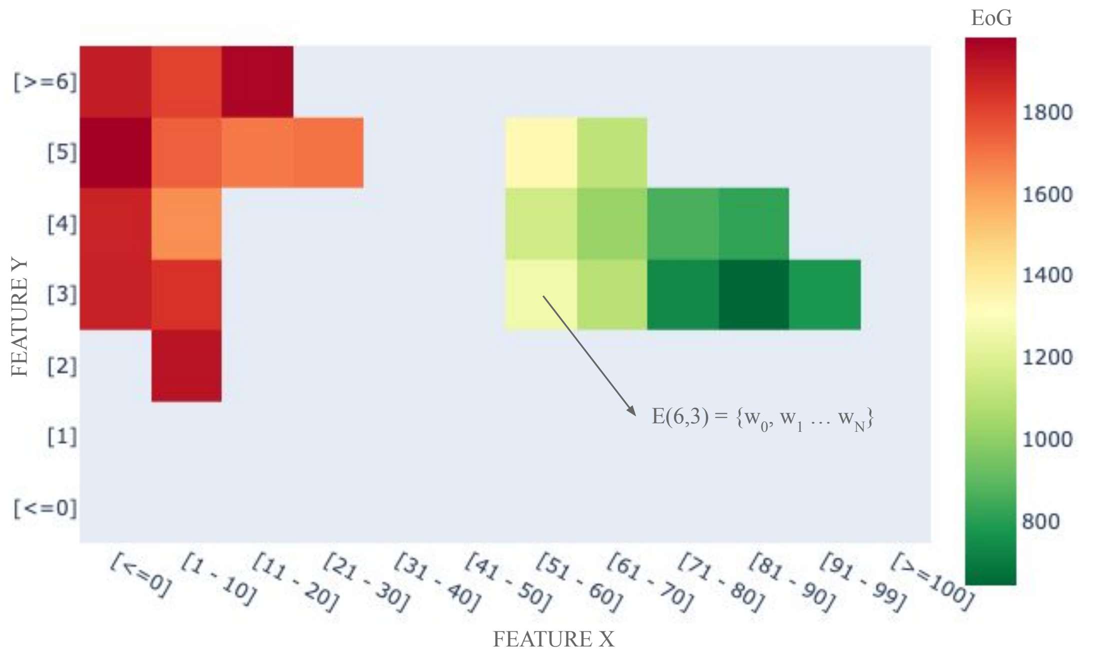


Figure 6.5: Example of a resulting MAP-Elites for a two-dimensional map with Features X and Y. These features are in different ranges and have buckets of different sizes. The map represents a *team* of 23 agents, and each cell contains the description (W) of each of them.

The features that form the map depend on the stats obtained from the gameplay, which are related to the goals defined for the agents. The implementation of the heuristic corresponding to each of these goals is dependent on the framework and games under consideration. I use the GVGAI Framework to carry out my experiments, so I implement and integrate the MAP-Elites algorithm described in this section within it. The next section describes the list of goals identified in different games and their particular implementation in this framework.

6.5 Goals and Heuristics Implementation

I identify 5 goals (heuristics) that can be used to elicit different ways to drive the actions of the agents. These are based on player-types goals and inspired by the list of general goals presented in Section 5.3. The heuristics implemented gather information related to their particular objectives. This data is used to obtain the stats required to assist during the generation of the *team*. These heuristics are general within the GVGAI Framework and can be used in any of its games. Although some of these heuristics may look similar to the ones presented in Chapter 4, their implementation has changed. The key differences result from 1) the variations in the application of the *heuristic diversification* (storing information related to the simulations of future states), and 2) addressing some limitations found in the initial implementation (e.g. only considering the visit to each location once).

Winning and score It prioritises winning the game while maximising the score difference. The heuristic is detailed in Algorithm 10. It is designed to heavily penalise states where the game is lost and reward those where it is won.

Algorithm 10 *Winning and score* heuristic.
Nomenclature: $S' \leftarrow$ simulated game state; $reward \leftarrow$ result of the evaluation of the state; $H \leftarrow$ high value; $currentScore \leftarrow$ current score of the game.

```
1: function WinningAndScore($S'$)
2:     $reward = 0$
3:     if $isGameOver(S')$ and $isLoser(S')$ then
4:         $reward = H^-$
5:     if $isGameOver(S')$ and $isWinner(S')$ then
6:         $reward = H^+$
7:     $score \leftarrow getScore(S')$
8:     $reward \mathrel{+}= (score - currentScore)$
9:     return $reward$
```

This goal is inspired by the *Winner* and *Record breaker* defined in Sections 5.3.1 and 5.3.4, respectively. It collects the following data: winning status (1 for win and 0 for lose), final score, game tick when the score changed last, and game tick when the last positive score change occurred.

Exploration It maximises the physical exploration of the map, which is divided into tiles. In contrast to the Exploration Maximisation Heuristic (EMH) described in Section 4.3.2, which just indicates if a position has been visited or not, *Exploration* takes into consideration the number of times each position has been visited. This heuristic prioritises visiting those positions that have not been visited before. Once they have been visited, it prioritises the ones visited the least number of times. It favours exploring as much as possible, so reaching an EoG state is heavily penalised. Algorithm 11 includes details about this heuristic.

It is necessary to make two clarifications on the design of this heuristic: 1) The number of visits of a location, *nVisits*, is subtracted from the reward. It is calculated by multiplying the total number of visits in the future simulated states by the total number of current visits. Another solution would be adding them instead, but the exponential penalisation given by the multiplication is preferred in this case. 2) At the end of the calculations, I need to adjust the reward when the number of future states simulated is not the maximum one. It is possible that an EoG state has been reached during the simulation and, therefore, the value of the reward obtained is not comparable with other 'full simulations'. I solve this problem by giving the 'missing' future simulated states a penalty corresponding to the maximum number of visits encountered in the map.

This goal is inspired by the *Map explorer* defined in Section 5.3.7. It collects the

Algorithm 11 *Exploration* heuristic.
Nomenclature: $S' \leftarrow$ simulated game state; $reward \leftarrow$ result of the evaluation of the state; $H \leftarrow$ high value; $futureExploredLocations[] \leftarrow$ record of the locations visited in the simulation.

```
1: function EXPLORATION($S'$)
2:     $reward = 0$
3:     if $isGameOver(S')$ then
4:         $reward = H^-$
5:     for $position$ in $futureExploredLocations[]$ do
6:         if $isNewLocation(position)$ then
7:             $reward += H^+$    ▷ Reward highly each non-visited position encountered
8:         else
9:             $nVisits = futureNVisits(position) * currentNVisits(position)$
10:            $reward -= nVisits$    ▷ Reward those visited less frequently
11:    $adjustment \leftarrow calculateRewardAdjustment()$
12:    return $reward - adjustment$
```

following data: the number of different locations visited, the final exploration matrix obtained with details about the number of visits on each position, and the game tick when the last exploration happened.

Curiosity It maximises the discovery and interaction with sprites in the game, prioritising interactions with new sprites. Interactions are defined as the avatar or any sprite generated from the player getting in contact with elements of the game (*collisions* and *hits*, respectively). When no new interactions are possible, it prioritises interactions in new locations of the game, what I define as *curiosity*. The details of the heuristic are included in Algorithm 12.

Algorithm 12 *Curiosity* heuristic.
Nomenclature: $S' \leftarrow$ simulated game state; $reward \leftarrow$ result of the evaluation of the state; $H \leftarrow$ high value; $H_1, H_2, H_3, H_4, H_5 \leftarrow$ modifiers applied to each of the elements considered for the reward; $futureInteractions[] \leftarrow$ record of the events triggered during the simulation

```
1: function CURIOSITY($S'$)
2:     $reward = 0$
3:     if $isGameOver(S')$ then
4:         $reward = H^-$
5:     $reward += H_1 * nNewSpritesDiscovered(futureInteractions[])$
6:     $reward += H_2 * nNewInteractions(futureInteractions[])$
7:     $reward += H_3 * nNewCuriosityInteractions(futureInteractions[])$
8:     $reward += H_4 * nTotalNewCuriosityInteractions(futureInteractions[])$
9:     $reward += H_5 * nTotalInteraction(futureInteractions[])$
10:    return $reward$
```

The heuristic considers and gives different rewards (from high to low value) to the following cases: a) new sprites discovered and new interactions; b) new curiosity interactions; c) the total of different curiosity interactions; d) the number of total interactions. It tries to investigate and interact with the game as much as possible so it penalises reaching EoG states.

This goal is inspired by the *Curious* defined in Section 5.3.10. It collects data related to sprites discovery: total number of different sprites discovered during gameplay, the list of their IDs, and the game-tick when the last discovery happened. It also gathers data related to the interactions in the game: number of unique interactions with sprites, number of curiosity interactions, number of total collisions, number of total hits, and the last game-tick when each of these last three interactions occurred.

Killing It maximises destroying Non-Player Characters (NPCs) and penalises End of Game states. The heuristic considers ‘kills‘ those interactions between sprites generated from the avatar (*hits*) and sprites of the type NPC. VGDL rules can represent killing enemies in different ways. Therefore, when using this heuristic in the GVGAI framework, the following assumptions are made about the rules of the game: a) enemies are killed in one hit; b) enemies are only killed by sprites generated from the avatar; c) the avatar is not able to kill an enemy by colliding with it or by using elements of the terrain. Details about the heuristic are included in Algorithm 13.

Algorithm 13 *Killing* heuristic.
Nomenclature: $S' \leftarrow$ simulated game state; $reward \leftarrow$ result of the evaluation of the state; $H \leftarrow$ high value; $futureInteractions[] \leftarrow$ record of the events triggered during the simulation

```
1: function KILLING($S'$)
2:     $reward = 0$
3:     if $isGameOver(S')$ then
4:         $reward = H^-$
5:     $nKills = nTotalNPCHits(futureInteractions[])$
6:     $reward \mathrel{+}= nKills$
7:     return $reward$
```

This goal is related to the type of player defined as *Killer* (Section 5.3.6). It collects the following data: the total number of enemies killed, their sprite id, and the game-tick when the last kill happened.

Collection It maximises the collection of items and penalises EoG states. The heuristic considers ‘collections‘ those interactions between the avatar (collisions) and sprites of the type *Resource*. VGDL rules can represent collecting items in different ways. Therefore, when using this heuristic in the GVGAI Framework, the following assumptions are made about the game rules: a) Items are of the *Resource* type; b) Items can only be collected by the avatar by colliding with them. Details about the heuristic are included in Algorithm 14.

This goal is related to the type of player defined as *Collector* (Section 5.3.5). It gathers the following information: the total number of items collected, their sprite id, and the game-tick of the last collection.

Algorithm 14 *Collection* heuristic.
Nomenclature: S' ← simulated game state; $reward$ ← result of the evaluation of the state; H ← high value; $futureInteractions[]$ ← record of the events triggered during the simulation

```
1: function COLLECTION($S'$)
2:     $reward = 0$
3:     if $isGameOver(S')$ then
4:         $reward = H^-$
5:     $nItems = nTotalResourceCollisions(futureInteractions[])$
6:     $reward \mathrel{+}= nItems$
7:     return $reward$
```

Given the generality of the heuristics within the GVGAI Framework, it is my responsibility to ensure the VGDL rules of the games are consistent and well-formed to fulfil the assumptions described. Not all of the goals apply to every game, as they depend on its characteristics (e.g. there is no point in guiding the agent to collect resources when the game does not include collectable items). Section 6.6.2 presents the games and establishes which heuristics are included in the experiments for each of them.

6.6 Experiment: *Team* Generation for Different Games

I run a series of experiments to test the approach in games with different characteristics by generating a *team* of agents with distinct behaviours for each of them. The *team* is obtained by assembling the resulting execution of distinct configurations of the MAP-Elites, with different pairs of features. In this section, I describe the games used, the enabled goals, the configurations of the MAP-Elites set, and the resulting maps. The code is in Github [Guerrero Romero, 2021a], while the *jar* executables, configuration files, and resulting JSON data can be found in an OSF repository [Guerrero Romero, 2021d]. All the resulting graphs corresponding to the set of MAP-Elites generated for each game are included in Appendix D.

6.6.1 Experiments configuration

For an easy experimental set-up, each execution of the algorithm is dynamically configurable with an external file; being able to choose the *controller* (OLMCTS) and the following attributes:

gameName, level Game and ID of the level the experiments are executed for.

nGameRuns Number of times the agent (candidate) plays the game with a certain behaviour description (W) to obtain its stats. I use 100 gameplays in every game and experiment. I consider that the most important characteristic to take into consideration for these experiments is the consistency of the stats obtained for the gameplay of the agents generated. It is required that the agents are assigned to the right cell of the map. This consistency is achieved by making the agent play the game a high number of times so the outliers do not skew the resulting average. A low number of data samples per game run (< 100) was not representative enough, while $1,000$ repetitions made the execution time too long.

nRandomInitialisations Number of random candidates generated during the second initialisation step. Set to 10 for every experiment.

nIterations Number of iterations of the MAP-Elites. We have a limited allocated time to run each experiment, so the value set depends on the complexity of the game and the number of heuristics provided. The number used for each game is defined in the following section.

feature X, feature Y I generate 2-dimensional maps for simplicity in the processing and display of the results. For each game, I execute the MAP-Elites with different pairs of features to obtain several maps that, together, assemble the resulting *team*. The list of available features used in my experiments are:

- **Wins** Rate of victories.
- **Score** Total amount of points at the end of the game.
- **Exploration** Different positions of the map visited.
- **Exploration percentage** Percentage of the map visited.
- **Discovery** Different types of sprites discovered.
- **Sprite interactions** Unique interactions with sprites of different types.
- **Curiosity** Unique interactions with sprites of different types in different locations of the map.
- **Collisions** Total avatar interaction with other sprites.
- **Hits** Total from-avatar sprites interactions with sprites.
- **Interactions** Total number of interactions of any type.
- **Kills** Total number of NPCs hit.
- **Items** Total number of resources collected.

6.6.2 Games, levels, and experiments codification

I apply the approach to four GVGAI games with different characteristics: *Butterflies*, *Zelda*, *Digdug*, and *Sheriff*. I have reviewed and updated the VGDL implementation of the games to make sure the rules are well-formed and the assumptions described in the heuristics are met. I have carried out this process to ensure the games can be used successfully with the heuristics designed. The games selected have different complexities in terms of winning conditions, number of sprites, and characteristics. They also define different types of players, allowing a range of goals and generated behaviours. The objective is to study how the methodology works when applied to distinct games and show its flexibility, being able to adapt to different games and goals. All the experiments are conducted on the same level of each game, shown in Fig. 6.6.

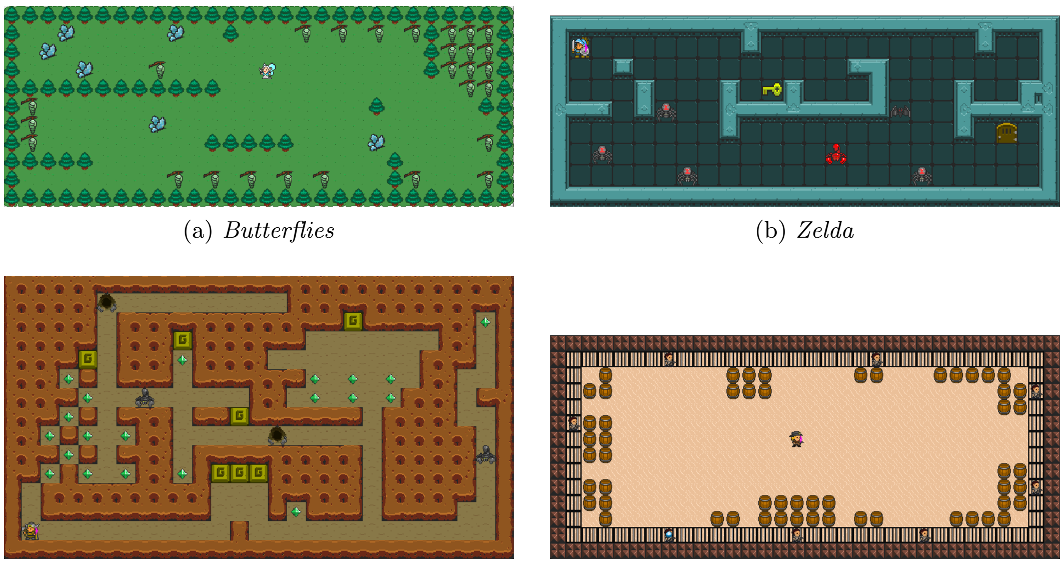

(a) *Butterflies* (b) *Zelda* (c) *Digdug* (d) *Sheriff*

Figure 6.6: Games and levels used to test the approach. I generate MAP-Elites with different pair of features for each to create a diverse *team* of agents.

Butterflies The rules of the game provided by the framework do not require changes, so they are detailed in Section A.3. This game does not allow *killing* NPCs or collecting items, so the corresponding heuristics are not enabled during the experiments. For this game, I run two sets of experiments with two sets of goals to compare the distribution and diversity of the agents when different heuristics are enabled. In both cases, the number of iterations of the MAP-Elites used to generate the agents is 200.

Zelda The rules of the game are detailed in Section 3.4.2. All heuristics are enabled for this game. This game is considered more complex than *Butterflies*, so initially, I set the number of iterations of the MAP-Elites to 250. However, given that the time required to execute the algorithm was higher than the maximum time allowed, the results are obtained after 125 iterations instead.

Digdug The rules of the game are detailed in Section 3.4.2. All heuristics are enabled for this game. Similarly to *Zelda*, the initial number of iterations of the MAP-Elites was 250, but for similar reasons, the results for this game are obtained after 100 iterations instead.

Sheriff The rules of the game are detailed in Section 3.4.2. In contrast to the games listed above, this one is a shooter and the avatar does not have access to the area where the enemies are (the *jail*), as they surround the player. There are *barrels* dispersed in the map that protect the player from the enemies' *bullets*. Although these barrels can be destroyed when hit, they are not collectable items. Therefore, in this case, only four heuristics are enabled. The number of iterations of the MAP-Elites used to generate the agents for this game is 200.

As previously stated, I run various executions of the MAP-Elites with different pairs of features for each game to assemble the corresponding *team*. For simplicity in the tables and results, I give a unique code to each set of experiments carried out, as follows:

B2 *Butterflies* with 2 heuristics enabled: *Winning and score* and *Exploration*.

B3 *Butterflies* with 3 heuristics enabled: *Winning and score*, *Exploration* and *Curiosity*.

Z5 *Zelda* with all 5 heuristics enabled: *Winning and score*, *Exploration*, *Curiosity*, *Killing* and *Collection*.

D5 *Digdug* with all 5 heuristics enabled: *Winning and score*, *Exploration*, *Curiosity*, *Killing* and *Collection*.

S4 *Sheriff* with 4 heuristics enabled: *Winning and score*, *Exploration*, *Curiosity* and *Killing*.

The selection of pairs of features used in each game depends on its characteristics. Table 6.1 shows the full set of experiments and the pair of features used in each of them, generating a total of 68 maps: 10 for B2, 10 for B3, 14 for Z5, 19 for D5, and 15 for S4. The range of the values and bucket size assigned to the features that determine the space of the MAP-Elites varies. It depends on the properties of the level related to the pair of features that form the map. I do not give details about each of them in this chapter, but they can be looked up in the *Features_ GAMELVL.java* file in the repository [Guerrero Romero, 2021a].

6.6.3 Resulting *teams*

The *nGameRuns* value is high (100), so each iteration of MAP-Elites takes a long time. The execution time also depends on the complexity of the game, the average time the game over is reached by the agents, and the number of heuristics enabled (as each of them requires their own calculations). The *nIterations* value is set at 200 for B2, B3, and S4. For Z5 and D5, the time spent on each iteration of the algorithm is higher. The total time required for the executions was higher than the maximum allocated time for

Y / X	Score	Exploration Percentage	Curiosity	Collisions (B2,B3) Interactions (D5,Z5,S4)	Kills	Items
Wins	All	All	All	All	D5 Z5 S4	D5
Exploration Percentage	All		All	All	D5 Z5 S4	D5
Curiosity	All			All	D5 Z5 S4	D5
Collisions (B2, B3) Interactions (D5,Z5,S4)	All				D5 Z5 S4	D5
Kills	S4					D5

Table 6.1: Combinations of feature pairs used for each set of experiments: B2 (10 configurations), B3 (10 configurations), Z5 (14 configurations), D5 (19 configurations), and S4 (15 configurations); for a total of 68 executions of MAP-Elites.

each experiment, so I obtain the *team* after 125 and 100 iterations of the MAP-Elites, respectively.

Running the experiments results in 68 maps for the five sets of experiments: B2, B3, Z5, D5, and S4. A total of 124, 302, 486, 293, and 352 agents are generated for each game and set of goals, respectively (Fig. 6.7). The group of elites generated in each map forms part of the *team* of available agents for future automated gameplay. Going through every map and game is prohibited for the sake of space. Thus, I include overall observations that stand out when going through the teams generated and a particular example for each game. All the resulting maps are included in Appendix D.

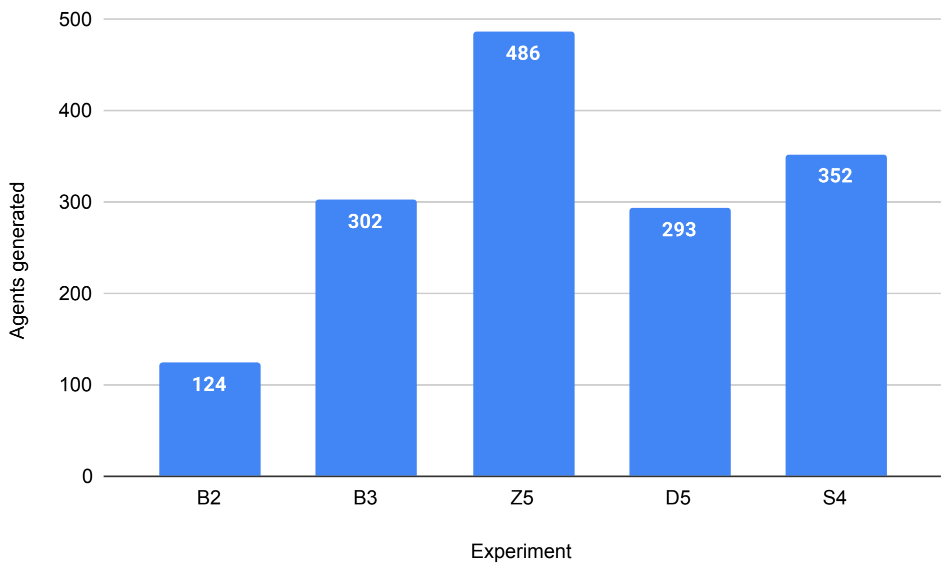


Figure 6.7: Total number of agents generated per experiment: B2 ($nIterations = 200$), B3 ($nIterations = 200$), Z5 ($nIterations = 125$), D5 ($nIterations = 100$), and S4 ($nIterations = 200$).

Agents performance The performance is given by how fast the agent reaches the End of Game (EoG). Not all the agents in the resulting MAP-Elites have a final good performance as, in some cases, they are very slow and obtain an average of EoG ticks very close to the maximum allowed. An example of the heterogeneous performance of the agents generated is shown in Fig. 6.8. My approach does not focus on the performance of the agents, as it merely serves as assistance to replace the agents when there is a collision. The interest comes from the diversity of the solutions generated and, thus, the location of the agents in the space of features. It serves as a reference to identify different behaviours, play-styles, and tasks. However, the final performance value of the elite gives a hint about the length of its average execution time.

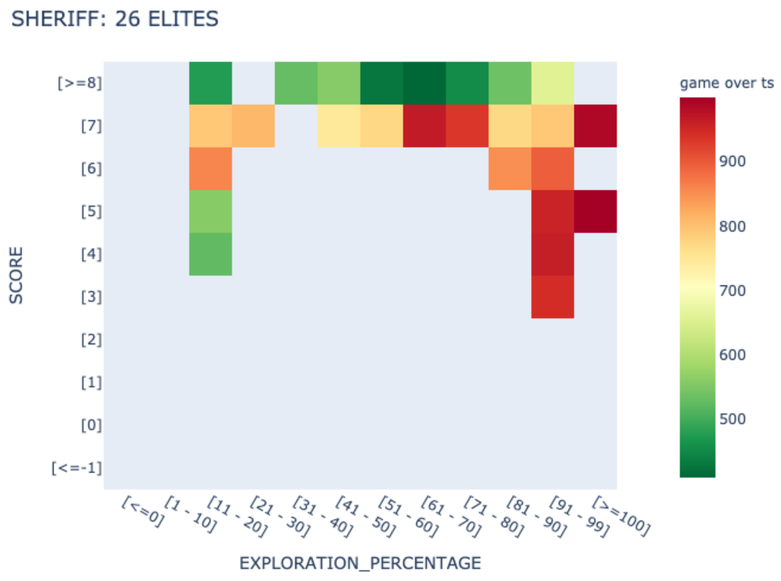
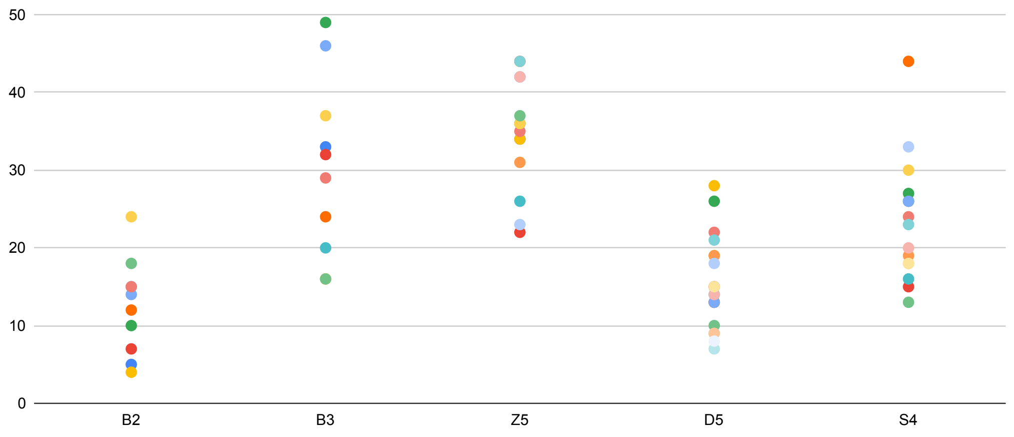


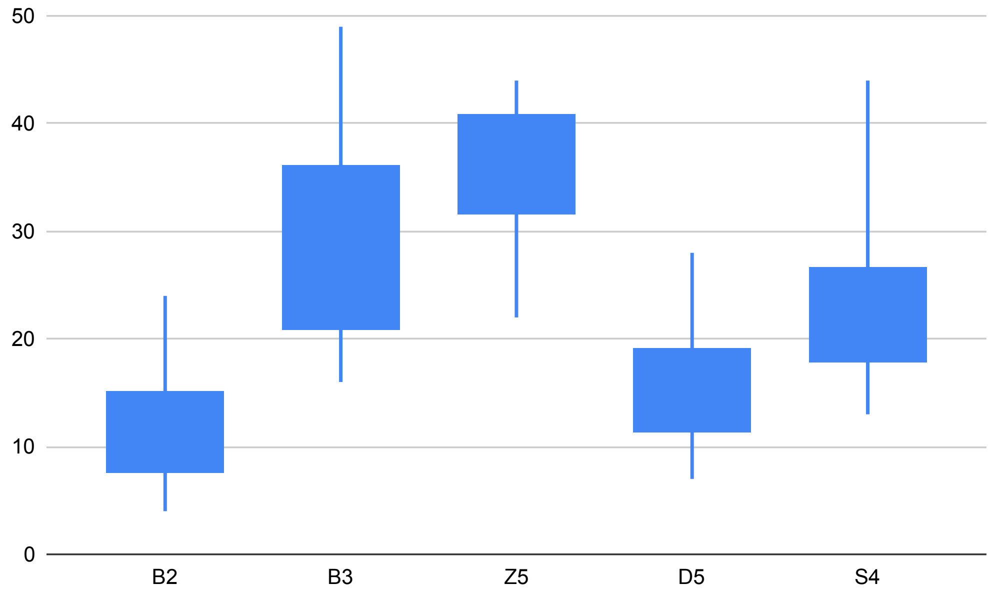
Figure 6.8: *Sheriff* (S4): Resulting MAP-Elites for features *Exploration percentage* and *Score*.

***Team* size** The size of the MAP-Elites generated varies both between games and the pair of features used (Fig. 6.9). For B2, the number of agents generated in each map is found between 4 and 24, for B3, between 16 and 49, for Z5, between 22 and 44, for D5, between 7 and 28, and for S4, between 13 and 44.

As an example, I look at the number of elites generated for the MAP-Elites with pair of features *Exploration percentage* and *Score* in each of the games (Figs. 6.8 and 6.10). In *Butterflies*, B2 generates 15 elites while B3 results in a total of 20. In *Zelda*, the number of agents generated is 42, a higher number than the ones generated for *Sheriff* (26) and *Digdug* (19). The results for D5 are provided after 100 iterations. For a fair comparison, I look at the number of elites generated for Z5 at 100 iterations, which is 39. This result is still significantly higher than 19. Overall, the size of the MAP-Elites generated for B2 and D5 is the smallest. For B2, only two of the three possible goals are applied to the agents, resulting in less diversity of final behaviours. *Digdug* is a game comparably more



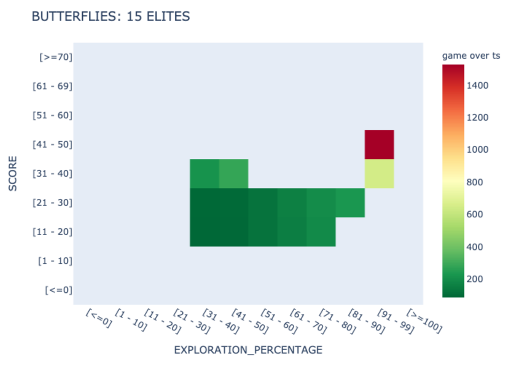
(a) *All map sizes*



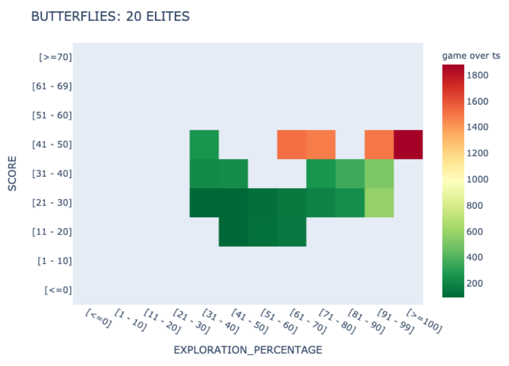
(b) *Distribution of map sizes*

Figure 6.9: Number of agents generated in each MAP-Elites per experiment: B2 (10 maps), B3 (10 maps), Z5 (14 maps), D5 (19 maps), and S4 (15 maps).

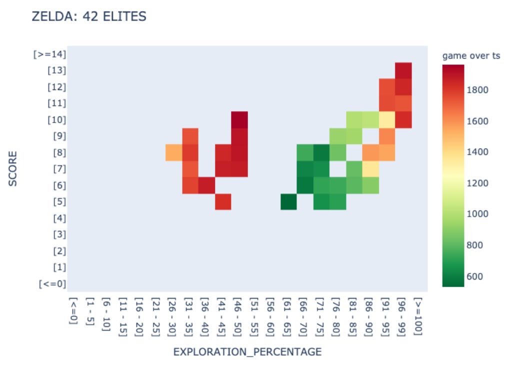
complex than the others. I speculate that in this game, either the MAP-Elites requires more iterations to reach diversity, or it is not possible to elicit further behaviours. As a result, the pool of agents available is smaller.



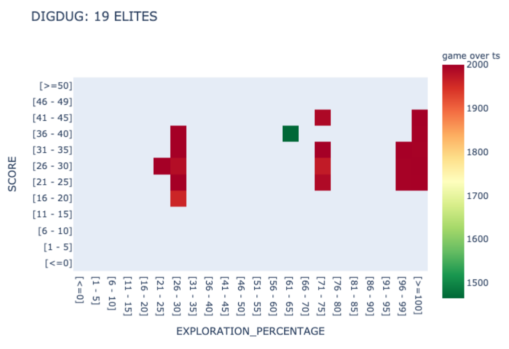
(a) *Butterflies* (B2)



(b) *Butterflies* (B3)



(c) *Zelda* (Z5)



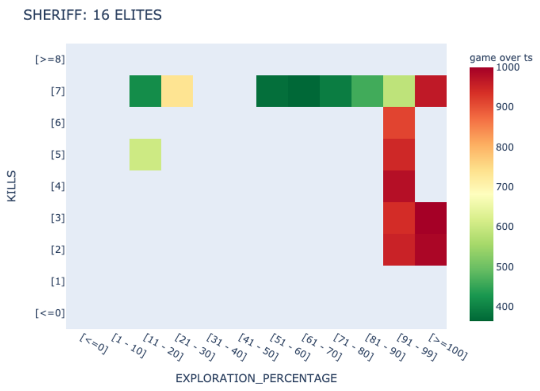
(d) *Digdug* (D5)

Figure 6.10: Resulting MAP-Elites for features *Exploration percentage* and *Score*.

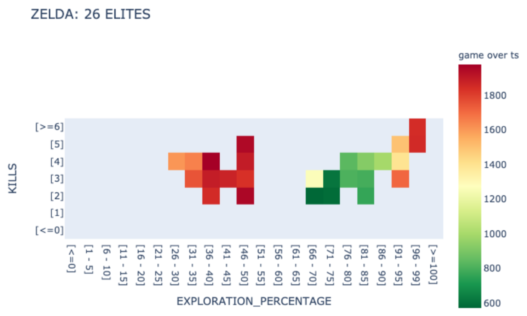
Agents distribution The distribution of the agents in the feature space is different between games for similar pair of features. This disposition provides information about the expected behaviour of the agents based on their location and the values obtained for each feature. However, their alignment also gives hints about the games themselves. I include two examples:

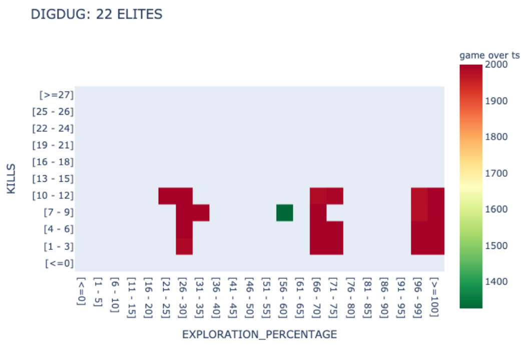
First, when looking at the set of MAP-Elites generated for *Exploration percentage* and *Score* features in B2, B3, Z5, and D5 (Fig. 6.10), all the agents generated achieve an average exploration percentage higher than 21% and, within the dimensions of the game, are found in a certain range of scores. However, the alignment of the agents is very different between games. While in *Butterflies* (Figs. 6.10a and 6.10b) and *Sheriff* (Fig. 6.8) there is a continuity in the occupied cells, the agents are *clustered* in blocks in *Zelda* and *Digdug* (Figs. 6.10c and 6.10d), having no coverage for certain exploration ranges. This aggregation is even more noticeable for the latter, where just a few exploration ranges ($21 - 30\%$, $61 - 65\%$, $71 - 75\%$, and $>= 96\%$) are included.

Second, I look at the maps generated for the *Exploration percentage* and *Kills* features for S4, Z5, and D5 (Fig. 6.11). For S4 (Fig. 6.11a) it is possible to identify agents with different behaviours: from an agent that does not move much but gets a high number of kills, to another that gets its kills while moving around the map achieving a high exploration rate. There is not much diversity on agents with a different number of kills for low exploration rates, as they achieve a minimum average of 5. On the contrary, the resulting map for Z5 (Fig. 6.11b) and D5 (Fig. 6.11c) shows different ranges of kills along different exploration rates. There are also differences in the alignments of the agents generated: for Z5, the distribution of the agents is almost linear and diverse (just missing agents in the $50 - 65\%$ exploration rate), while the ones generated for D5 are gathered in clusters. Each of these results makes sense when looking at the rules of the games: in *Sheriff* the agent kills the bandits by shooting bullets, so it does not need to move around to hit them. Mobility grants the agent the ability to get cover, go closer to the enemies to kill them, or dodge the bullets when it tends not to kill them, allowing more diversity in the range of kills achieved in that case. In both *Zelda* and *Digdug*, the enemies are killed by hitting them close range, so it is necessary to reach them on the map to kill them. In *Zelda*, all enemies move freely, being able to kill different numbers of them while reaching various levels of exploration. In *Digdug*, on the contrary, the enemies are also limited by the wall, so the agent needs to explore it (and break walls) to reach them. The clusters give an idea of the disposition of the enemies in located areas.



(a) *Sheriff* (S4)





(b) *Zelda* (Z5) (c) *Digdug* (D5)

Figure 6.11: Resulting MAP-Elites for features *Exploration percentage* and *Kills*.

Diverse goals results in a more diverse *team* When comparing the size of the MAP-Elites generated for B2 and B3 (Fig. 6.12), I can infer that the inclusion of the new goal (*Curiosity*) in B3 results in a more diverse *team* of agents for a similar setup and number of iterations of the MAP-Elites (200). A total number of 124 agents are generated for B2 and 302 for B3.

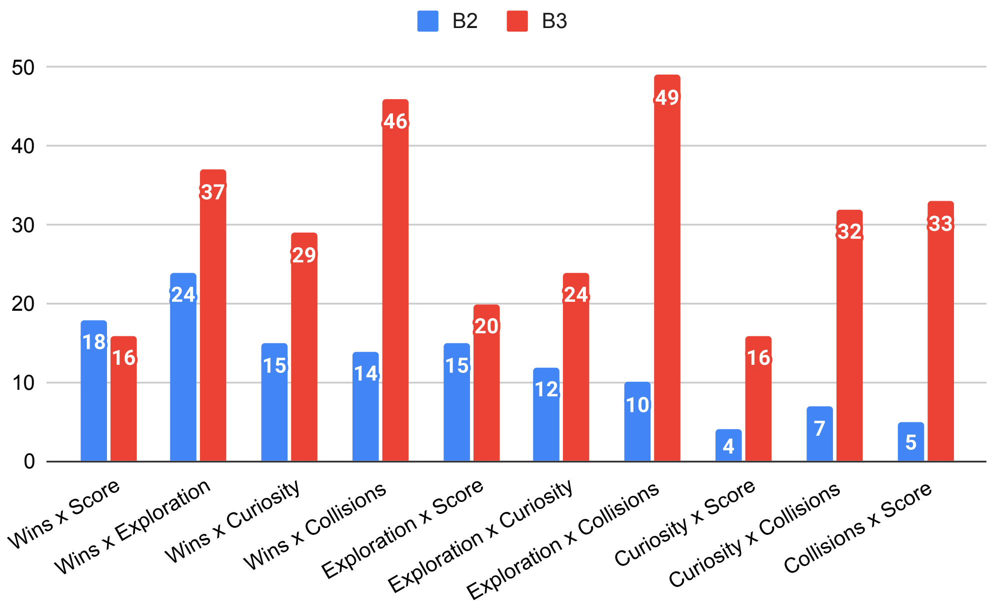
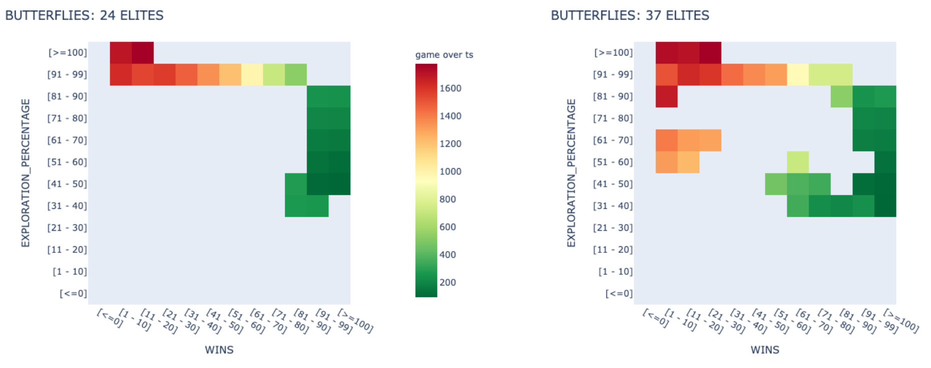


Figure 6.12: *Butterflies*: Total number of agents generated per MAP-Elites for B2 and B3 (*nIterations* = 200). *Exploration percentage* appears as *Exploration* in the graph.

This diversity is clear in the maps generated for the pair of features related to curiosity and interactions: *Exploration percentage* x *Collisions* results in 10 final elites for B2 and 49 for B3; and *Curiosity* x *Score* in 4 and 16 respectively. However, this increased diversity is true even for pair of features not directly related to the new goal, as shown in the MAP-Elites for *Wins* and *Exploration percentage* (Fig. 6.13). Including *Curiosity*, in this case, allows generating agents that obtain different rates of exploration for a higher range of winning rates, providing more diversity and flexibility on the selection of agents to suit different needs. Another example is the map resulting for the pair of features *Exploration percentage* and *Score* (Fig. 6.10a and 6.10b). Including the additional goal increases the choices of agents when looking for a high score ($41-50$): B2 only produces an agent capable of reaching this high score, tight to a high exploration percentage ($91-99\%$). B3, in contrast, generates a total of 5 agents related to this high score, each of them achieving different ranges of exploration ($31-40\%$, $61-70\%$, $71-80\%$, $91-99\%$, and 100%). I conclude that by having more heuristics enabled in a game, it is possible to generate more behaviours and create a more diverse *team*.

Particular example of B3 Fig. 6.14 presents the resulting 49 agents for B3 when the feature pair is *Exploration Percentage* and *Collisions*. The figure shows how the



(a) B2 (b) B3

Figure 6.13: *Butterflies*: Resulting MAP-Elites for features *Wins* and *Exploration Percentage*.

higher the number of collisions, the higher the exploration percentage and the average EoG time of the agents. It is possible to select agents to cover different rates of the map. However, when looking for agents able to obtain more than 300 collisions, this selection gets reduced to agents that also attain a very high exploration ($> 90\%$). Yet, if the target is to obtain more than 1,000 collisions, two additional agents are found in a lower exploration range. Their curiosity value average is 100.64 and 103.76. There are 102 trees and 33 butterflies in the game. Therefore, the behaviour I expect from these two agents is interacting with the trees, sticking to the borders of the map, and avoiding getting into the *open* areas, unless they get the chance to interact with a butterfly at reach.

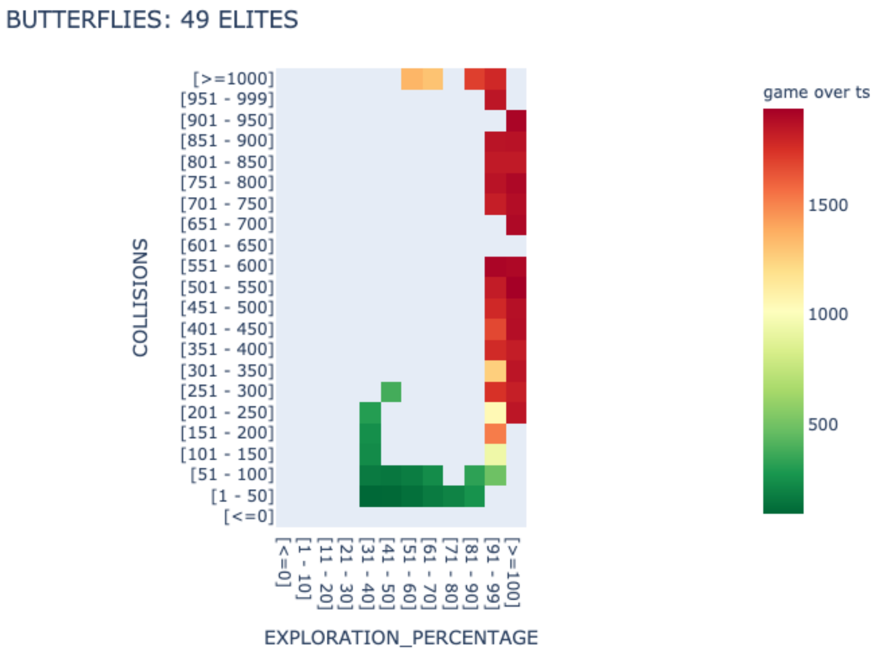


Figure 6.14: *Butterflies* (B3): Resulting MAP-Elites for features *Exploration percentage* and *Collisions*.

Particular example of Z5 Fig. 6.15 shows the resulting MAP-Elites for Z5 when the feature pair is *Interactions* and *Kills*. The solution generates 35 agents capable of killing at least 2 monsters while interacting with the elements of the game in different capacities. The range of the average of iterations elicit by this *team* goes from $[1 - 100]$ to $[1401 - 1500]$. The pool of agents with the most diverse proficiency at killing monsters is tight to the lowest number of iterations ($[1 - 200]$). When requiring an agent to reach higher iterations, it is expected that the agent ends up killing at least 4 or 5 monsters when playing the game. If it is desired that an agent kills all monsters when playing, the pool is reduced to 5 agents, none of them able to reach an average of interactions higher than 800.

Table 6.2 gives details about five of the agents in this map (highlighted in Fig. 6.15). The features in the MAP-Elites work as guidance on diversifying the behavioural space. The resulting stats of these agents are different, exhibiting diversity in the way each of them relates to the game. This information helps to have a better understanding of what to expect from their gameplay. While E1 is expected to win and E5 to lose, the latter would achieve a slightly higher score than the former and would interact with elements of the game much more. Given this information, plus E5's low exploration skills and the fact that it rarely picks the key (the only item in the game), I speculate that this agent rarely reaches the area where the key is. It probably stays at the left and bottom zones of the map.

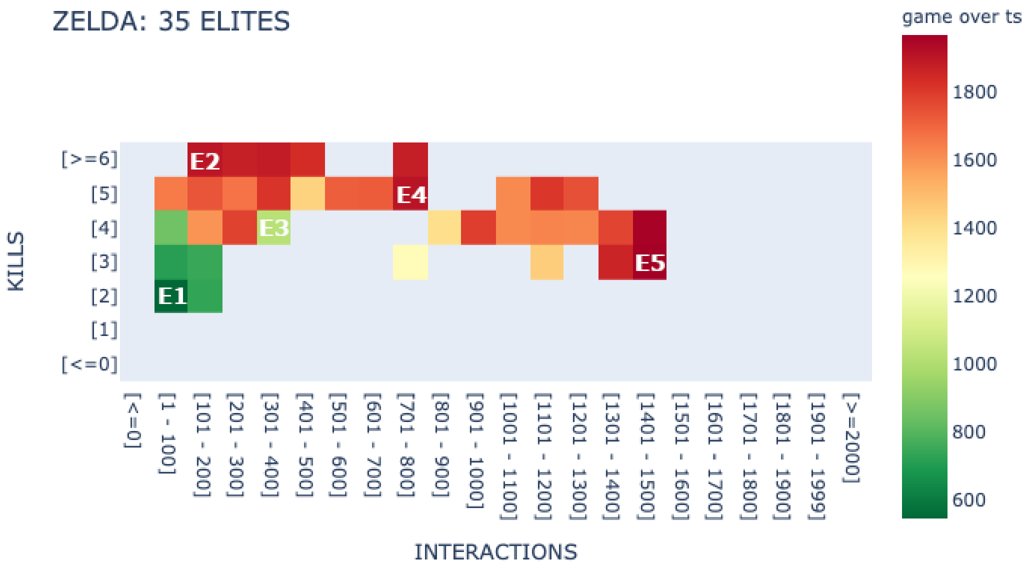


Figure 6.15: *Zelda* (Z5): Resulting MAP-Elites for features *Interactions* and *Kills*. Table 6.2 includes details about the agents highlighted in the graph (*En*).

Particular example of D5 Fig. 6.16 shows the resulting MAP-Elites for D5 for the feature pair *Kills* and *Items*. Only one of the 18 agents available in the pool has a comparably low average of game ticks (1400). This particular agent is in the range of $[22 - 24]$ items and $[7 - 9]$ kills, so it is on the top tier of the agents encountered in this

	E1	E2	E3	E4	E5
Weights	$[0.66, 0.13, 0.01, 0.04, 0.16]$	$[0.23, 0.13, 0.01, 0.64, 0.16]$	$[1.0, 0.13, 0.68, 0.58, 0.16]$	$[0.52, 0.13, 0.68, 0.64, 0.16]$	$[0.0, 0.0, 0.8, 0.0, 0.77]$
X: ***Interactions***	37.91	150.07	314.52	713.64	1411.24
Y: ***Kills***	1.95	5.53	3.62	5.38	3.32
EoG ticks	544.32	1905.71	1023.71	1912.60	1970.68
Gameplay stats (average of 100 plays)					
Win rate	91.99%	6.00%	70.00%	8.00%	0.00%
Score	5.72	12.12	8.8	11.81	7.18
Exploration percentage	70.35%	95.80%	78.00%	96.01%	34.61%
Sprite interactions	3.33	4.99	4.46	5.07	3.75
Curiosity	25.70	80.65	61.82	90.95	37.61
Collisions	35.96	144.54	310.90	708.26	1407.92
Hits	1.95	5.53	3.62	5.38	3.32
Kills	1.95	5.53	3.62	5.38	3.32
Items	0.97	0.98	0.98	0.99	0.60

Table 6.2: *Zelda* (Z5): Details of agents highlighted (*En*) in Fig 6.15 resulting from the MAP-Elites generated for the pair of features *Interactions* and *Kills*. It includes the description of the *weights* of each heuristic (*Winning and score*, *Exploration*, *Curiosity*, *Killing*, *Collection*) as well as the associated value of their features, End of Game (EoG) ticks and average stats resulting when an agent with this description plays the game 100 times.

space, although it is not the one that reaches the most number of monster kills and items collected. All the agents can collect at least 7 items and kill between 7 and 12 monsters, but they rarely win the game (the highest win rate find in the team is 0.05%); so they are either killed by a monster or play for its whole duration. In general, most of the solutions of the MAP-Elites generated for *Digdug* show an average of EoG ticks very close to the maximum allowed (2000), the lowest value being 1200. Therefore, automated gameplays for this game are expected to be slow, independently of the behavioural description of the agent chosen.

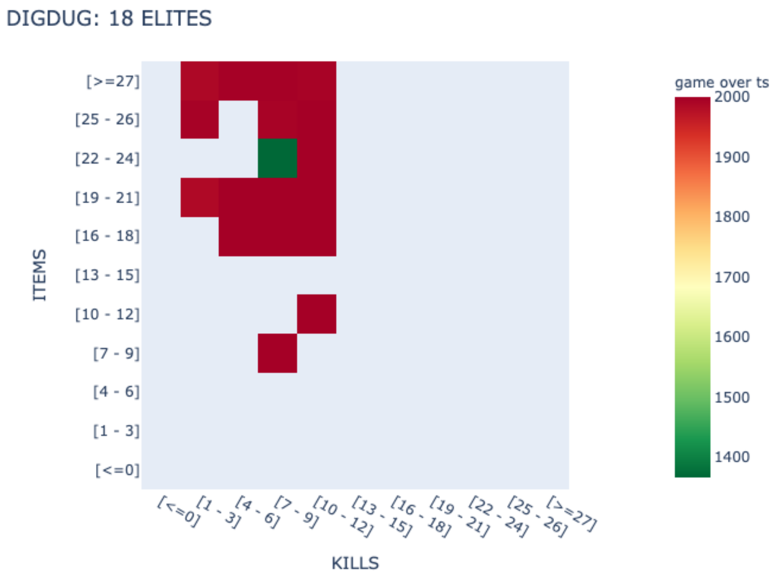


Figure 6.16: *Digdug* (D5): Resulting MAP-Elites for features *Kills* and *Items*.

Particular example of S4 Fig. 6.17 shows the resulting MAP-Elites for the features *Wins* and *Kills* for S4. The solutions generate the description of 20 agents with different rates of wins. One of the agents loses most of the time ($1 - 10\%$ of wins) and kills only 1 bandit on average, while the rest of them have different rates of wins and average of kills. There are a total of 8 bandits in the level, and the game finishes when all of them are killed or by surviving for 1000 game ticks. None of the agents averages 8 kills, which is an odd result, as many of them win the game 100% of the time with a low average of EoG ticks, which imply that they win by killing the enemies and not by surviving. Therefore, there may be an issue with the implementation of the *Killing* heuristic. This problem is probably related to failing to log the information about the events in the last game tick. However, I do not believe this is a critical problem, and I consider that the pool of agents generated can still be used with the expected purpose. In the case of extending the work or carrying out new experiments to generate further MAP-Elites, the heuristic should be reviewed to fix this issue. Nevertheless, I can identify different types of agents: those that win the game by either being proficient at killing the bandits or by surviving and avoiding being shot, with varying ranges of rates of kills. There are also some agents that, having an average win rate ($21 - 70\%$), can kill most of the bandits and survive for a while. This kind of behaviour can become useful in some aspects of testing or tasks achievement, where the agent is not expected to win but manages to survive for a long time and kill most of the enemies.

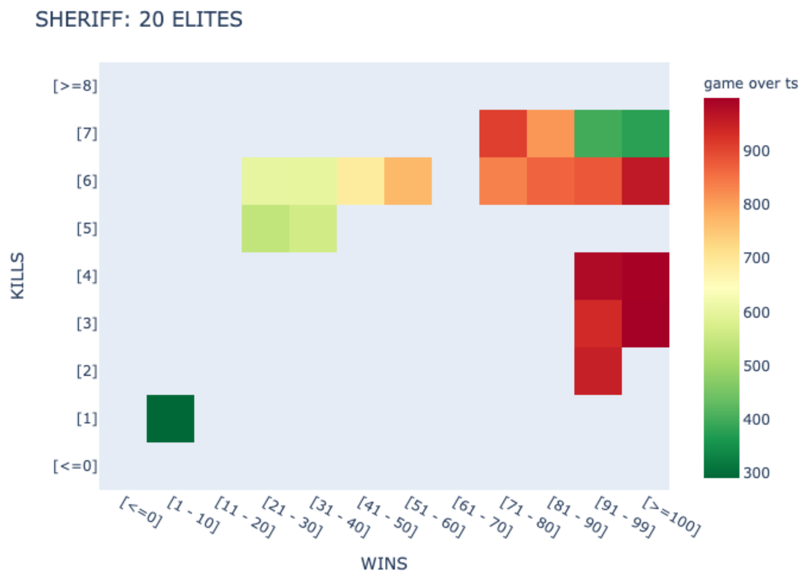


Figure 6.17: *Sheriff* (S4): Resulting MAP-Elites for features *Wins* and *Kills*.

In summary, the collection of MAP-Elites generated shows a variety of agents dispersed in a behavioural space, leading to the existence of agents with different behaviours to run automated gameplay. It is possible to identify different play styles and tasks in the *team* to fit various needs and requirements.

6.6.4 Summary and discussion

The motivation of the experiment described in this section was to generate agents in games with different characteristics to prove the generality of the approach proposed in Section 6.4. The results provide a *team* of agents, which is assembled from the group of maps generated with MAP-Elites and available to play each game automatically. Each agent of the pool is expected to interact and behave differently in the game. However, not all of them would be useful for developers or designers if this approach were to be applied in the game development process. They would not be expected to run all the agents generated to test the game but a selection from the ones available based on their needs. Therefore, for the approach to be applicable in the future as a design tool, it is first needed to find a way to find those agents that may be of interest based on the game under development. The agents are located in behavioural spaces based on the outcomes of their gameplay, so this distribution should allow identifying agents with target abilities based on their location in the corresponding two-dimensional map. The following chapter describes such identification by selecting 6 agents of different characteristics for each game.

6.7 Interactive Tool: Automated Gameplay Visualisation

I am interested in the characteristics of each of the agents and studying their behaviour when playing the game. The graphs generated give information about the distribution of the agents in the feature space, and it is possible to identify what to expect from them by looking at their correspondent cells. However, static images have a limitation when looking at a more extensive analysis. Moreover, the results are produced in JSON files, so going through the data of the agents assigned to each cell of the map to get detailed information about them is a tedious process. To make the processing and reading of the results easy, I have created an interactive tool, accessible online [Guerrero Romero, 2021e].

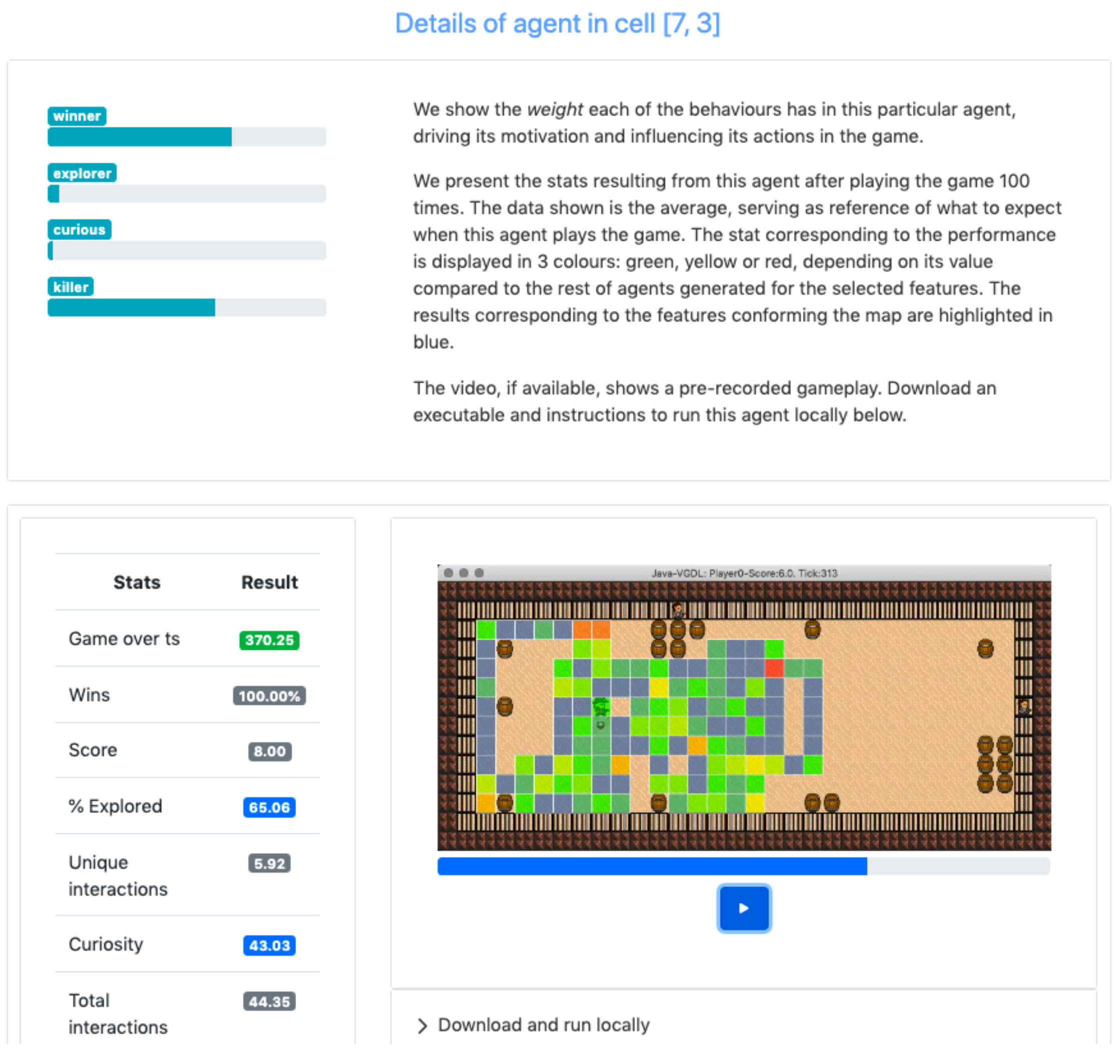
This interactive tool complements and extends my work. It makes it possible to access the results of the experiments in real-time. It generates interactive maps to retrieve detailed information about each of the agents: resulting stats, performance, and an example of their gameplay. It contains the following features:

Welcome page Includes an overview of the demo, definitions, and step-by-step instructions on how to use the tool. It provides details about the goals, games, and features used to generate the MAP-Elites.

Data selection Choose the game and pair of features to load and visualise the corresponding MAP-Elites generated. The data is stored in JSON files, and the user can choose between the available files, displayed in a dropdown field.

***Team* visualisation** Presents an overview of the game, the heuristics enabled for the agents, the features used, and a heatmap graph that represents the MAP-Elites generated. This graph is interactive, and each cell is given a colour based on the agent's performance (EoG time). The user can hover over each of the elements presented on the screen to get detailed information about them.

Agent details and gameplay Selecting a cell from the map gives details about the corresponding agent. It shows its behavioural description (weights assigned to each of its heuristics), the stats resulting from the 100 gameplays of the level, and a pre-recorded video exemplifying its gameplay. A screenshot of this component of the tool is displayed in Fig. 6.18.



Details of agent in cell [7, 3]

We show the *weight* each of the behaviours has in this particular agent, driving its motivation and influencing its actions in the game.

We present the stats resulting from this agent after playing the game 100 times. The data shown is the average, serving as reference of what to expect when this agent plays the game. The stat corresponding to the performance is displayed in 3 colours: green, yellow or red, depending on its value compared to the rest of agents generated for the selected features. The results corresponding to the features conforming the map are highlighted in blue.

The video, if available, shows a pre-recorded gameplay. Download an executable and instructions to run this agent locally below.

Stats	Result
Game over ts	370.25
Wins	100.00%
Score	8.00
% Explored	65.06
Unique interactions	5.92
Curiosity	43.03
Total interactions	44.35

Figure 6.18: Screenshot of the interactive tool showing the information and gameplay of one of the agents generated for *Sheriff*. It is accessible online [Guerrero Romero, 2021e].

Download files to run the agents locally I provide a standalone executable to run the agents locally. I have generated a JSON for each agent description that serves as its configuration and allows triggering automated gameplay of it. The files and instructions are available to download.

This tool is considered relevant for both academics and a more general audience. It provides an interactive visualisation of the results and allows witnessing the distinct behaviours of the agents first-hand. I use it to navigate the maps generated for each game and identify behaviours corresponding to different players and tasks, presented in the following chapter.

6.8 Conclusions

The work presented in this chapter applies MAP-Elites to generate and assemble a *team* of agents with distinct behaviours so they can be used for automated gameplay. The solution provides a pool of agents in a feature space and their location gives an idea of what to expect when they play the game. The features of the map are defined by the results of the actions of the agents: *wins*, *score*, *exploration*, *kills*, *items collected*, etc. The performance of the agents is not defined by how *well* they perform on the game in terms of score or wins. It is based on the time it takes the agents to play the game when obtaining a similar range of values for the features that define their location on the map.

I implement and integrate the approach in the GVGAI Framework and use the OLM-CTS as the controller to play the games. The *sampleMCTS* provided by the framework has been modified by *heuristic diversification* to provide a list of *heuristics* and corresponding *weights* externally. The heuristics implemented are based on objectives that can be found on different games: *Winning and score*, *Exploration*, *Curiosity*, *Killing*, and *Collection*. The list of features and heuristics used in this work can be adapted and extended based on the game under consideration and the needs of the user. The generality of the algorithm and heuristics within the framework allows to execute the methodology in different games. The MAP-Elites is applied to generate agents for 4 games of different characteristics and complexity. In one of the games, two different sets of heuristics are enabled, distinguishing between 5 blocks of experiments: B2, B3, Z5, D5, and S4. For simplicity in the codification and presentation of the results, I use a 2-dimensional feature space, so several independent executions of the MAP-Elites are required to cover different behavioural spaces. A total of 68 different configurations of the MAP-Elites have been executed. These executions result in a pool of agents of various sizes per game and set of experiments: 124 (B2), 302 (B3), 486 (Z5), 293 (D4), and 352 (S4). The size and diversity of the agents generated by the MAP-Elites should make it possible to find agents in each of the games fulfiling different needs. The options are limited by the distribution of the agents within the space, caused by the pair of features, the characteristics of the game, and the *enabled heuristics* set for the agent. These agents could be used for automated gameplay with different objectives. The resulting stats of the agents used as reference refer to the level of the game used for their generation.

The generality of the heuristics in the solution allows running the agents generated at any level of the game. However, whether the behaviour identified to those agents is portable to new levels is an open question that needs to be answered if I ultimately want to use the *team* of agents to assist in the development and testing of games. Chapter 7 focuses on answering this question by identifying different *behaviour-type* agents in each of the games and comparing the tendency of their stats in alternative levels.

Chapter 7

Application: Team Portability and Level Testing

This chapter applies the *team* of GVGP agents generated. It identifies *behaviour-type* agents that reach certain targets, assess their portability to new levels, and presents exploratory work to use them to test levels and spot issues in them. Most of the material presented in this chapter has been included in a journal paper submitted to IEEE Transactions on Games that is currently under review.

7.1 Introduction

This chapter is an immediate continuation of Chapter 6, where I presented an approach to generate a *team* of GVGP agents with differentiated behaviours. I created a pool of agents distributed in various feature spaces, assembling such a *team* for four different games. However, I still need to address how to identify distinct behaviours and accomplishments to ultimately use those agents to assist in the game development and testing processes. Therefore, I am still looking at answering RQ2: *How to define, create, and use a team of GVGP agents with distinct behaviours to assist in the development and evaluation of games?*

Within the methodology envisioned in Chapter 5, this chapter focuses on the *team* of agents and on finding an approach to use them to assist in the evaluation of the game (Fig. 7.1). Following the MAP-Elites procedure described in Chapter 6, the *team* of agents has been generated at a particular level of the game. Before proposing a method to use the agents for its evaluation, I need to make sure that each of those agents generated is actually portable to other levels of the game. The generality of the GVGP agents allows them to play any level, but the question is if their strength and expected behaviour are transferred as well. This work first identifies a series of agents for each game, characterised by particular aspects and resulting features that I find interesting. Then, I apply these agents in an experiment to test their portability to new levels. Lastly, I introduce an exploratory work that envisions the use of the agents to test the design and validity of new levels. I use those same agents to test a 'broken' level in each of the games as proof of concept.

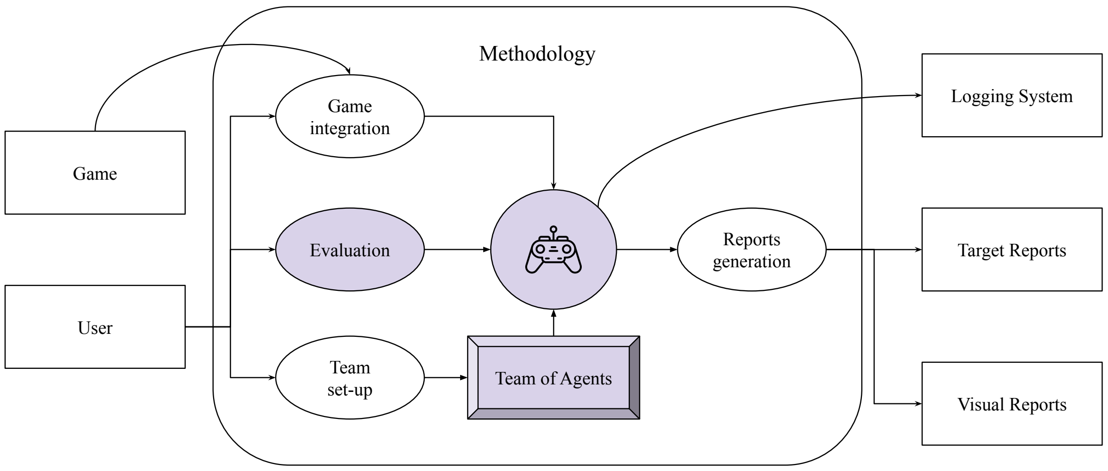


Figure 7.1: Highlighted the areas that I cover in this chapter from the long-term vision. It presents an approach to identify different members of the *team*, use them to play a level automatically, and, ultimately, assist on its evaluation.

I take advantage of my previous work and carry out these experiments in the GVGAI Framework using the same four games as in Chapter 6: *Butterflies*, *Zelda*, *Digdug*, and *Sheriff*. Fig. 7.2 shows the particular levels that I have previously used to generate the agents. I utilise the interactive tool described in Section 6.7 to navigate through the different maps that assemble the *team* to find different *behaviour-type* agents to use in my experiments. As a reference, the full pool of agents available for each game is included in Appendix D. For the portability and level testing experiments, I define new levels for each game. Their details and screenshot are included in the corresponding sections.

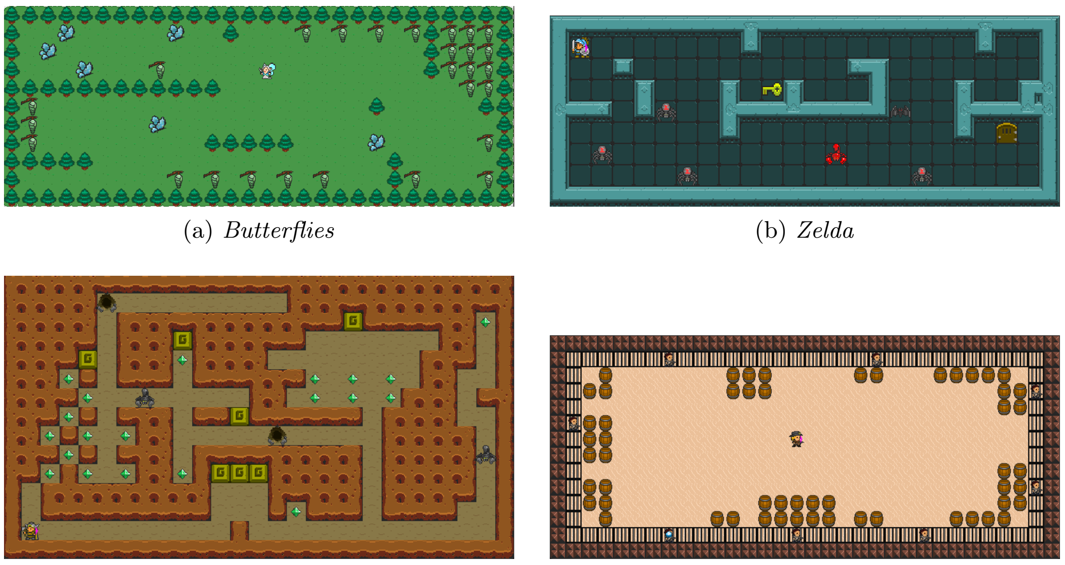

(a) *Butterflies* (b) *Zelda* (c) *Digdug* (d) *Sheriff*

Figure 7.2: Screenshot of the levels used to generate the agents for each game.

7.2 Identification of *Behaviour-type* Agents from the *Team*

I identify 6 agents for each game that correspond to behaviours and tasks that interest me. These agents are used in both the portability and testing experiments included in this chapter. These *behaviour-types* agents have been selected from the resulting maps that assemble the *team* for each game (Section 6.6.3). I use the concept of *behaviour-type* agents instead of *player-types* or *personas* as in the existent literature (Sections 2.4.1, 2.4.2, and 2.4.3) because those concepts are either related to game design or with agents mimicking or modelling human behaviour. I look at the behaviour of the agents as the results (stats and features) of their gameplay, without necessarily resembling players when playing the game. I have used the interactive tool described in Section 6.7 to navigate the collection of maps generated and identify different agents based on their features and location in the space.

The description of the different *behaviour-type* agents identified across the games, in alphabetical order, are the following.

Barrels shooter It only applies to *Sheriff.* Agent with a high rate of *interactions* and *hits* but a low rate of *kills*, so it targets the barrels instead of the bandits. It is identified in the map with the corresponding features.

Collector (high/low) Agent that gathers a high or low number of *items.* It is identified in one of the maps with the corresponding feature.

Curiosity (high/low) Agent with a high or low resulting *curiosity.* It is identified from one of the maps with the corresponding feature.

Explorer (high/low) Agent with a high or low resulting physical *exploration* of the level. It is identified from one of the maps with the corresponding feature.

Interactions (high/low) Agent with a high or low resulting *interactions* with elements of the game. It is identified from one of the maps with the corresponding feature.

Killer (high/low) Agent with a high or low resulting *kills* of enemies. It is identified from one of the maps with the corresponding feature.

Scorer (High/Low) It is identified from one of the maps with the corresponding feature.

Speed-runner Agent with a high victory rate that tends to finish the game fast. It is identified by looking through the agents with a very high *win rate* in the maps with such a feature to find one with one of the lowest EoG ticks.

Survivor It only applies to *Sheriff*. Agent that survives until the time runs out, winning the game. It is identified in the map with the corresponding feature by looking at a resulting high EoG ticks.

Walls breaker It only applies to *Digdug*. Agent that focuses on breaking the walls and tends to follow them instead of moving through open areas. It is identified by having a high *curiosity* but a mid-range *exploration* rate in the map with the corresponding features. In contrast to the *Walls interaction*, introduced below, this agent is expected to break the walls, so although the *hits* are not used for its identification, they are also a relevant gameplay result.

Walls interaction It only applies to *Butterflies*. Agent that focuses on interacting with the trees and butterflies. It is expected to move close to the walls bypassing open areas. It is identified by having high *curiosity* but average *exploration* features in the map with the corresponding features, and confirmed by observing its gameplay.

Some of the agents identified are a combination of two of these *behaviour-types*. The group of agents chosen is different between games (Table 7.1), but each of the final 6 agents per game is given a similar nomenclatures to facilitate the readability of the results: E1, E2, E3, E4, E5, and E6. The selection is different in each game because the tasks and goals I want the agents to elicit depends on its characteristics and rules. For each game, I include the MAP-Elites from the *team* where the agents are identified from. I also include a table showing an overview of the *behaviour-type* agents for each game and corresponding nomenclature given in it. This table gives details about their behaviour attributes, the weights that describe their heuristic and drive their actions, and information about where to find them. This information includes the pair of features that created the map and cell they are assigned to, as well as the resulting values of the corresponding stats and end of game (EoG) ticks.

7.2.1 *Butterflies*

All the agents selected for *Butterflies* are taken from the results of B3, as I have previously determined that the diversity of behaviour in the *team* is richer than the one resulting for B2.

The rules in *Butterflies* are pretty simple compared with the rest of the games, as there are no items to collect or enemies to kill. Given the rules' simplicity, there were few possible behaviours. It is quite an interesting game, given that a higher score does not imply victory, as the game is won as soon as all the butterflies are captured. It is possible to finish it before all the available butterflies in the level are spawned. Therefore, two of the agents in the selection of this game are related to the score feature. There are two different interactive game elements: butterflies and trees (the walls), so I also considered the interactions an interesting behaviour for the agents. Related to this, I

Behaviour-type	***Butterflies***	***Zelda***	***Digdug***	***Sheriff***
Barrels Shooter				X
High collector			X	
Low collector			X	
High curiosity	X			X
High explorer	X	X	X	X
Low explorer		X	X	X
Low interactions				X
High killer		X	X	X
Low killer		X	X	X
High scorer	X	X	X	
Low scorer	X			
Speed-runner	X	X		X
Survivor				X
Walls breaker			X	
Walls interaction	X			
High collector + high killer			X	
High collector + low killer			X	
Low collector + high killer			X	
High curiosity + low interactions				X
Low explorer + high scorer			X	
High killer + high explorer		X		X
High killer + low explorer		X		X
Survivor + low killer				X

Table 7.1: *Behaviour-type* agents overview per game. Some of the agents identified are a combination of two individual *behaviour-type* agents (highlighted in the table), so they are expected to show both traits. The table includes both the individual and combined *behaviour-type* agents for reference.

made a distinction between two different types of agents: one that interacts with the elements of the game in different locations (curiosity trait), and another that will mainly focus on interacting with the walls, avoiding open areas of the level. Finally, I was also interested in an agent capable of finishing the game as quickly as possible or exploring the level instead of looking at capturing butterflies. I list the *behaviour-type* agents identified for this game with the corresponding nomenclature that is used for each of them in this chapter, detailed them in Table 7.2, and include the corresponding map and cell they are located at.

- E1: *Low scorer* (Fig. 7.3).
- E2: *High scorer* (Fig. 7.3).
- E3: *High curiosity* (Fig. 7.4).
- E4: *Speed-runner* (Fig. 7.5).
- E5: *High explorer* (Fig. 7.6a).
- E6: *Walls interaction* This agent should focus on interacting with the trees and butterflies, so it is expected to move close to the walls bypassing open areas. I look into curiosity and exploration to identify it (Fig. 7.6b).

Butterflies (B3)						
	E1	**E2**	**E3**	**E4**	**E5**	**E6**
Behaviour-type	***Low scorer***	***High scorer***	***High curiosity***	***Speed-runner***	***High explorer***	***Walls interaction***
Map features	*Collisions* *Score*	*Collisions* *Score*	*Curiosity* *Collisions*	*Wins* *Exploration %*	*Exploration %* *Collisions*	*Exploration %* *Curiosity*
Cell id	$[1, 2]$	$[8, 6]$	$[7, 21]$	$[11, 4]$	$[11, 11]$	$[7, 6]$
Features value	8.09 19.18	370.36 50.72	121.92 1398.23	100% 37.72%	99.92% 540.19	65.63% 108.15
EoG	95.69	1860.99	1675.32	87.93	1939.34	1479.75
Weights W	$\{0.92, 0.52, 0\}$	$\{0.1, 0.45, 0.67\}$	$\{0, 0, 0.21\}$	$\{0.94, 0.08, 0.02\}$	$\{0.02, 0.29, 0.68\}$	$\{0.04, 0, 0.9\}$

Table 7.2: Full selection of agents from the *team* for *Butterflies* identified based on their behaviour. It includes their attribute, the MAP-Elites and cell assigned to, the values for each feature, and the corresponding End of Game (EoG) ticks. It also presents the weights that describe the heuristic assigned to the agents.

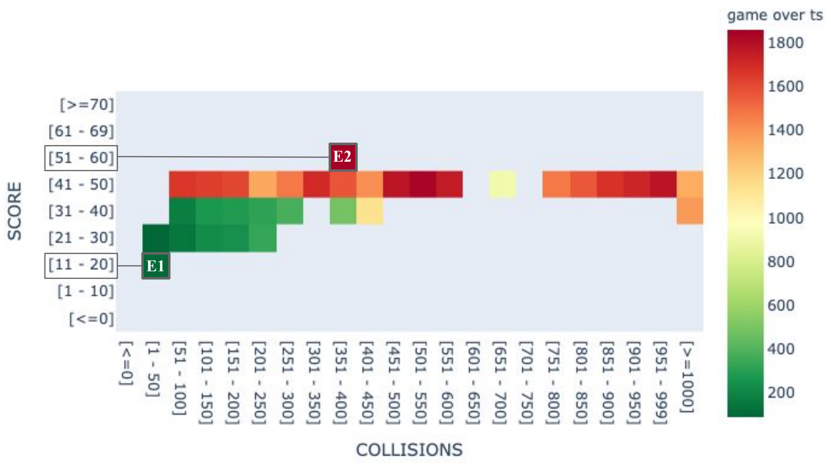


Figure 7.3: *Butterflies behaviour-type* agents identification: *Low scorer* (E1), with the lowest score feature across the maps, and *High scorer* (E2), with the highest score feature across the maps.

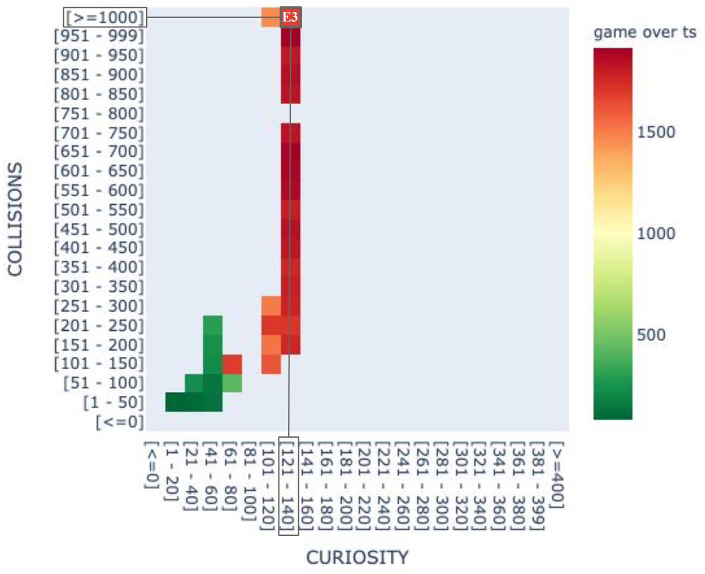


Figure 7.4: *Butterflies behaviour-type* agent identification: *High curiosity* (E3), with the highest pair of collisions and curiosity values.

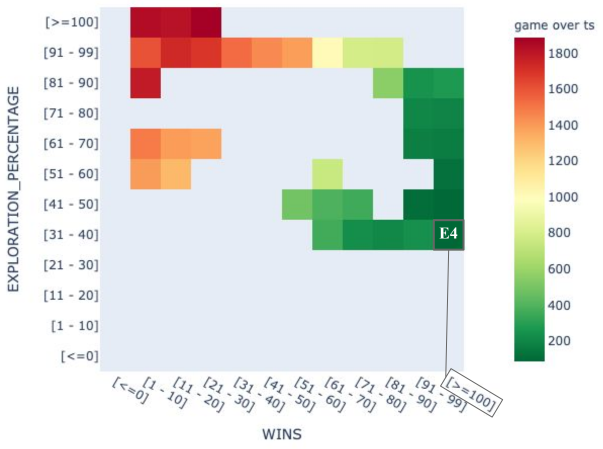
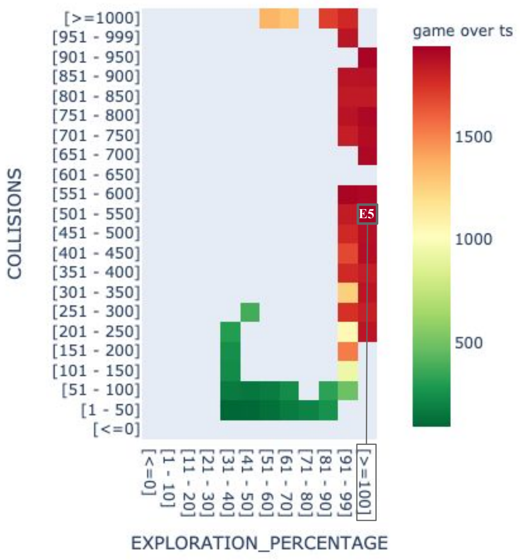
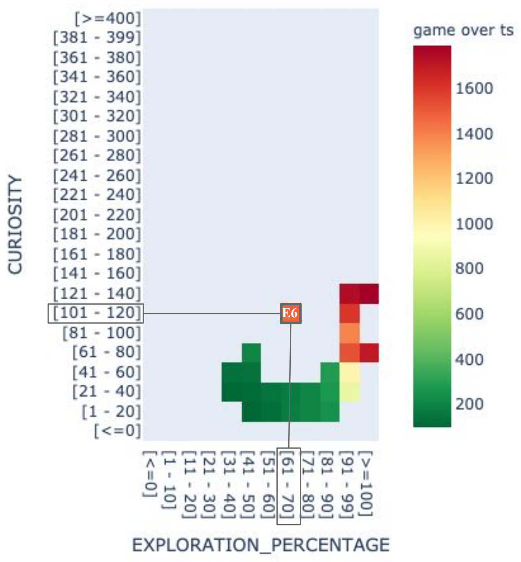


Figure 7.5: *Butterflies behaviour-type* agent identification: *Speed-runner* (E4), with ≃ 100% win rate and one of the lowest EoG ticks.



(a) *High explorer* (E5)



(b) *Walls interaction* (E6)

Figure 7.6: *Butterflies behaviour-type* agents identification: *High explorer*, with a high exploration rate (≃ 100%), and *Walls interaction*, with high curiosity but mid-range exploration rate.

7.2.2 *Zelda*

Zelda is a *dungeon game* where the objective is collecting a key and opening the door to escape. There are enemies, but it is the choice of the player to engage with them or not. Therefore, I was interested in finding agents that would focus on killing enemies in different ways: one that would barely engage with the monsters, and another two that would engage with them and kill them, but in two different ways, either exploring the level and finding them or by covering less part of the level but killing them when they appear in their path. As the score is also given by the key and finishing the level, it was another interesting behaviour (only a high score in this case). Similarly to the previous game, finishing the game fast or exploring the level instead were behaviours I wanted to see in the agents. The only collectable item in the game is the key, so I did not find it very interesting to find agents focused on collection. Even if it was possible to choose an agent with this kind of behaviour, I did not find it important for my experiments.

I list the *behaviour-type* agents identified for this game with the corresponding nomenclature that is used for each of them in this chapter, detailed them in Table 7.3, and include the corresponding map and cell they are located at.

- E1: *High scorer* (Fig. 7.7).
- E2: *Speed-runner* (Fig. 7.7).
- E3: *High explorer* (Fig. 7.8).
- E4: *Low killer* (Fig. 7.9).
- E5: *High killer and explorer* (Fig. 7.10).
- E6: *High killer and low explorer.* This agent is expected to kill the monsters when they approach, instead of 'finding them' (Fig. 7.10).

Zelda (Z5)						
	E1	**E2**	**E3**	**E4**	**E5**	**E6**
Behaviour-type	***High scorer***	***Speed-runner***	***High explorer***	***Low killer***	***High killer + high explorer***	***High killer + low explorer***
Map features	*Wins* *Score*	*Wins* *Score*	*Exploration %* *Curiosity*	*Interactions* *Kills*	*Exploration %* *Kills*	*Exploration %* *Kills*
Cell id	$[0, 13]$	$[10, 6]$	$[20, 7]$	$[1, 2]$	$[20, 6]$	$[10, 5]$
Features value	0.00% 12.85	99.00% 6.37	97.00% 62.66	37.91 1.95	95.64% 5.50	46.92% 4.64
EoG	1966.42	450.05	1902.58	544.32	1849.54	1951.38
***Weights** W*	$\{0, 0.62, 0, 0.59, 0\}$	$\{0.96, 0.54, 0.61, 0, 0\}$	$\{0.92, 0.61, 0.01, 0.93, 0.36\}$	$\{0.66, 0.13, 0.01, 0.04, 0.16\}$	$\{0.08, 0.55, 0.04, 0.84, 0.51\}$	$\{0, 0, 0, 1, 0\}$

Table 7.3: Full selection of agents from the *team* for *Zelda* identified based on their behaviour. It includes their attribute, the MAP-Elites and cell assigned to, the values for each feature, and the corresponding End of Game (EoG) ticks. It also presents the weights that describe the heuristic assigned to the agents.

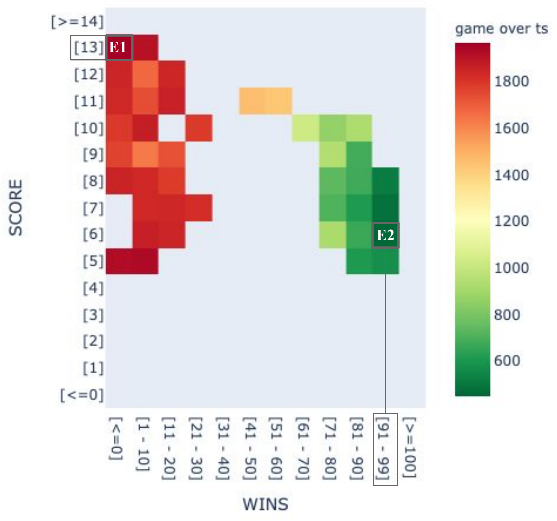


Figure 7.7: *Zelda behaviour-type* agents identification: *High scorer* (E1), with the highest score feature across the maps, and *Speed-runner* (E2), with one of the highest win rate and lowest EoG ticks.

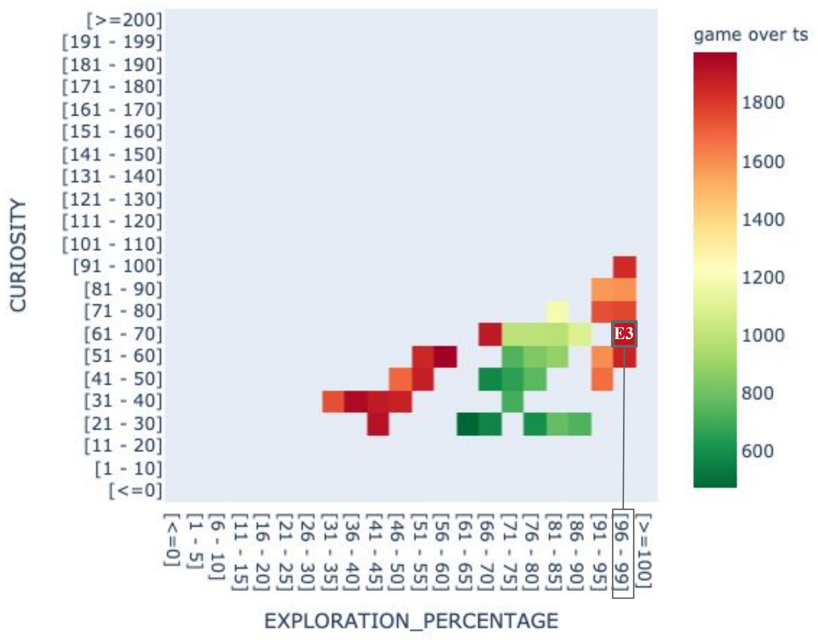


Figure 7.8: *Zelda behaviour-type* agents identification: *High explorer* (E3), with one of the highest exploration rate across the maps.

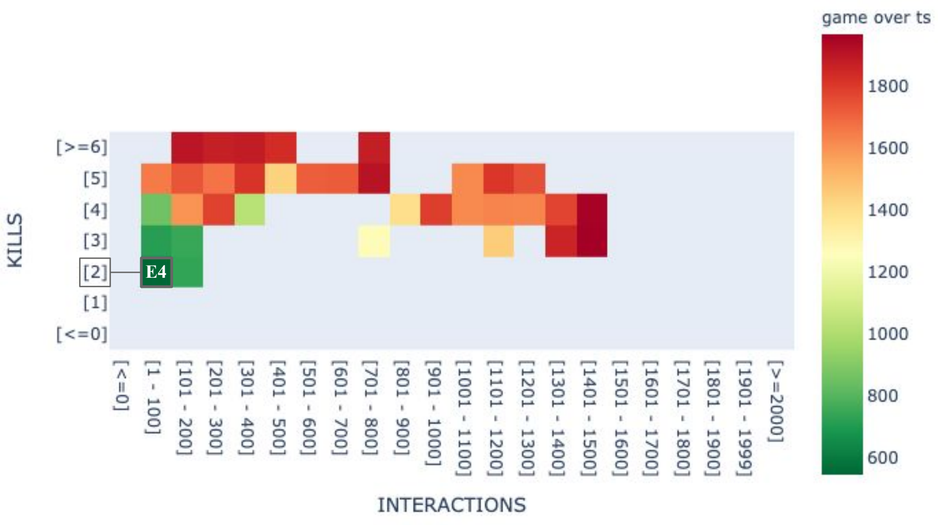


Figure 7.9: *Zelda behaviour-type* agents identification: *Low killer* (E4), with one of the lowest average of kills across the maps.

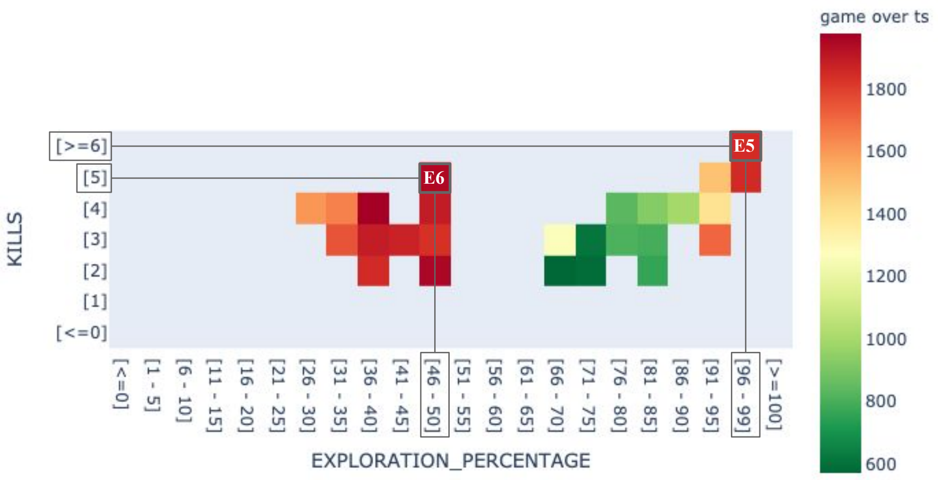


Figure 7.10: *Zelda behaviour-type* agents identification: *High killer and explorer* (E5), with a high average of kills and exploration rate, and *High killer and low explorer* (E6), with a high number of kills but low exploration rate, not being able to cover on average even half of the map

7.2.3 *Digdug*

Digdug is the only game from the set with (various) collectable items. Therefore, most of the agents I identify for this game are related to this behaviour. Killing monsters is also a big part of the game, so I thought it would be interesting contrasting these two behaviours and see what would happen when an agent would either focus on both or, on the contrary, only on one over the other. Similarly to the previous games, I consider exploration a necessary behaviour. The score is given by the collection and kills. Both items and enemies are dispersed in the level, so I found it interesting to look for an agent capable of achieving a high score without having to cover a big part of the map. Finally, the walls in this game are breakable, so they are an essential element, and I wanted to find an agent that mainly focuses on breaking them. It was possible to find agents with other behaviours or further combinations of the ones identified, but I considered those were the most relevant for my experiments.

I list the *behaviour-type* agents identified for this game with the corresponding nomenclature that is used for each of them in this chapter, detailed them in Table 7.4, and include the corresponding map and cell they are located at.

- E1: *High collector and killer* (Fig. 7.11).
- E2: *High collector and low killer* (Fig. 7.11).
- E3: *Low collector and high killer* (Fig. 7.11).
- E4: *Walls breaker* (Fig. 7.12). This agent should focus on breaking the walls, so it is expected to follow them instead of moving through open areas. I look into curiosity and exploration. Although the resulting number of *hits* is not used for the identification, it is also relevant information, as the walls break when the agent hits them.
- E5: *High explorer* (Fig. 7.12).
- E6: *Low explorer and high scorer* (Fig. 7.13).

Digdug (D5)						
	E1	**E2**	**E3**	**E4**	**E5**	**E6**
Behaviour-type	***High collector + high killer***	***High collector + low killer***	***Low collector + high killer***	***Walls breaker***	***High explorer***	***Low explorer + high scorer***
Map features	*Kills* *Items*	*Kills* *Items*	*Kills* *Items*	*Exploration %* *Curiosity*	*Exploration %* *Curiosity*	*Exploration %* *Score*
Cell id	$[4, 10]$	$[1, 10]$	$[4, 4]$	$[14, 19]$	$[21, 13]$	$[6, 8]$
Features value	10.55 26.90	2.84 26.98	9.91 9.83	67.14% 450.88	100.00% 324.97	29.77% 35.77
EoG	1995.89	1989.92	1999.99	1783.33	1999.01	1999.93
Weights W	$\{0, 0, 0.7, 0.92, 1\}$	$\{0, 1, 0, 0, 1\}$	$\{0, 0, 0, 1, 0\}$	$\{0, 0, 0.69, 0, 0.01\}$	$\{0, 0.7, 0.86, 0, 0.01\}$	$\{0, 0, 0, 0.49, 0.33\}$

Table 7.4: Full selection of agents from the *team* for *Digdug* identified based on their behaviour. It includes their attribute, the MAP-Elites and cell assigned to, the values for each feature, and the corresponding End of Game (EoG) ticks. It also presents the weights that describe the heuristic assigned to the agents.

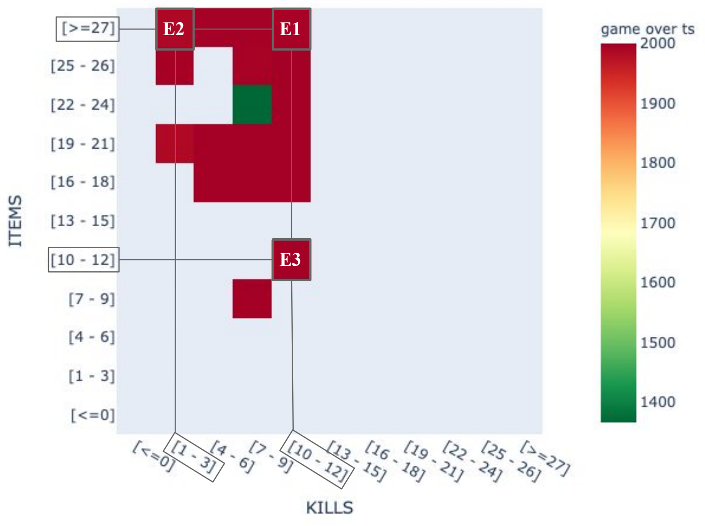


Figure 7.11: *Digdug behaviour-type* agents identification: *High collector and killer* (E1), with a high number of items collected and enemies killed, *High collector and low killer* (E2), with a high number of items collected but a low number of enemies killed, and *Low collector and high killer* (E3), with a low number of items collected but a high number of enemies killed.

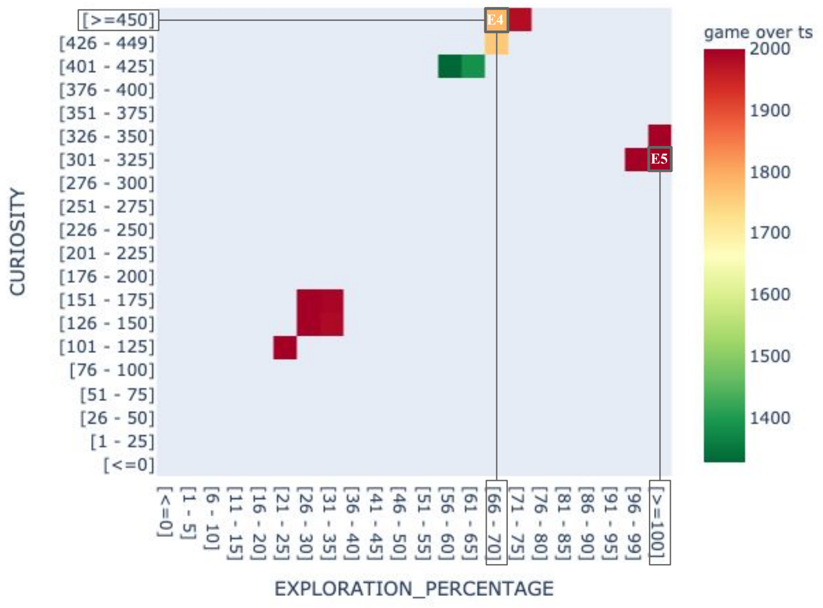


Figure 7.12: *Digdug behaviour-type* agents identification: *Walls breaker* (E4), with a high curiosity but a mid-range exploration rate, and *High explorer* (E5), with one of the highest exploration rate (100%).

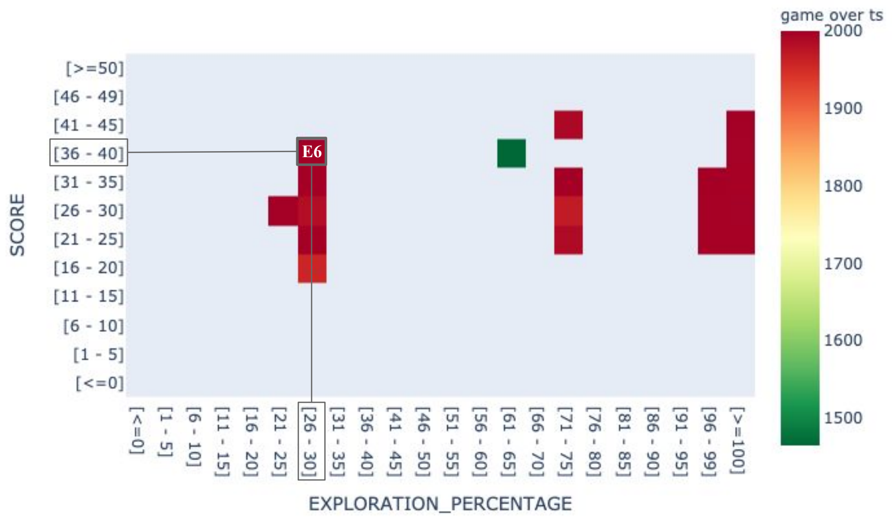


Figure 7.13: *Digdug behaviour-type* agent identification: *Low explorer and high scorer* (E6), with a low exploration rate but a high score.

7.2.4 *Sheriff*

Sheriff, in contrast to the previous games, is a shooter, so the enemies and items are destroyed by hitting them from a distance. Therefore, most of the agents I identified for this game are related to killing enemies or shooting at different game elements. Furthermore, there are two ways of winning the game: killing all the enemies or surviving until the time runs out. I chose to find one agent related to each one of these possibilities.

I list the *behaviour-type* agents identified for this game with the corresponding nomenclature that is used for each of them in this chapter, detailed them in Table 7.5, and include the corresponding map and cell they are located at.

- E1: *Survivor and low killer.* This agent wins the game by reaching the EoG, so it avoids being killed by the enemies and does not focus on killing them either. (Fig. 7.14).
- E2: *High killer and explorer.* This agent kills a lot of enemies while moving a lot around the map (Fig. 7.15).
- E3: *High killer and low explorer* This agent kills a lot of enemies but stays in a small area of the level, so it could be considered a *'Sharp shooter'* (Fig. 7.15).
- E4: *Speed-runner* (Fig. 7.16).
- E5: *Barrels shooter* (Fig. 7.17).
- E6: *High curiosity and low interactions* (Fig. 7.18).

Sheriff (S4)						
	E1	**E2**	**E3**	**E4**	**E5**	**E6**
Behaviour-type	***Survivor + low killer***	***High killer + high explorer***	***High killer + low explorer***	***Speed-runner***	***Barrels shooter***	***High curiosity + low interactions***
Map features	*Wins* *Kills*	*Exploration %* *Kills*	*Exploration %* *Kills*	*Wins* *Score*	*Interactions* *Kills*	*Curiosity* *Interactions*
Cell id	$[11, 3]$	$[11, 7]$	$[2, 7]$	$[11, 9]$	$[8, 5]$	$[6, 3]$
Features value	100% 3.19	99.50% 7.00	14.04% 6.99	100.00% 7.99	350.80 (33.18 hits) 5.03	101.29 127.83
EoG	999.00	966.77	412.30	359.17	549.71	819.76
Weights W	$\{0.84, 1, 0, 0\}$	$\{0, 0.26, 0, 0.46\}$	$\{0.09, 0, 0, 1\}$	$\{0.57, 0.05, 0.03, 0.92\}$	$\{0, 0, 0.19, 0\}$	$\{0.28, 0.25, 0.9, 0.46\}$

Table 7.5: Full selection of agents from the *team* for *Sheriff* identified based on their behaviour. It includes their attribute, the MAP-Elites and cell assigned to, the values for each feature, and the corresponding End of Game (EoG) ticks. It also presents the weights that describe the heuristic assigned to the agents.

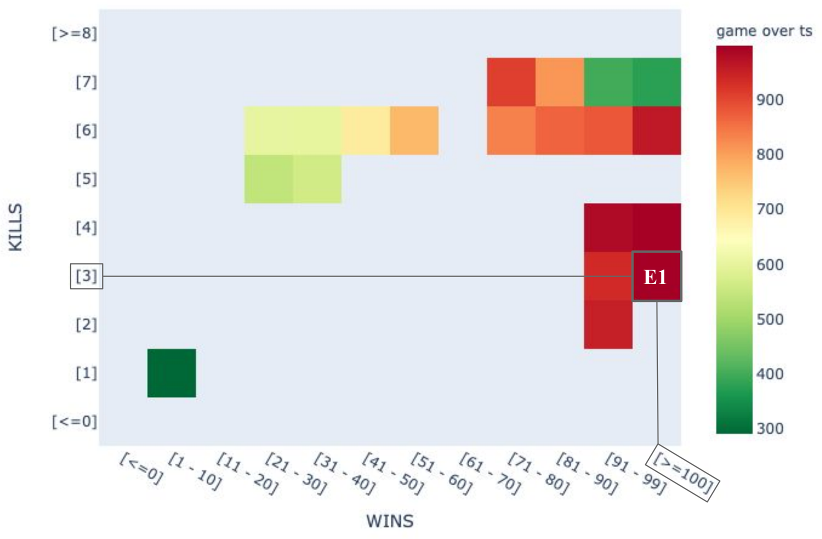


Figure 7.14: *Sheriff behaviour-type* agent identification: *Survivor and low killer* (E1), with a low number of enemies killed but high win rate (100%), achieved by reaching the timeout without dying (EoG $\simeq 1000$).

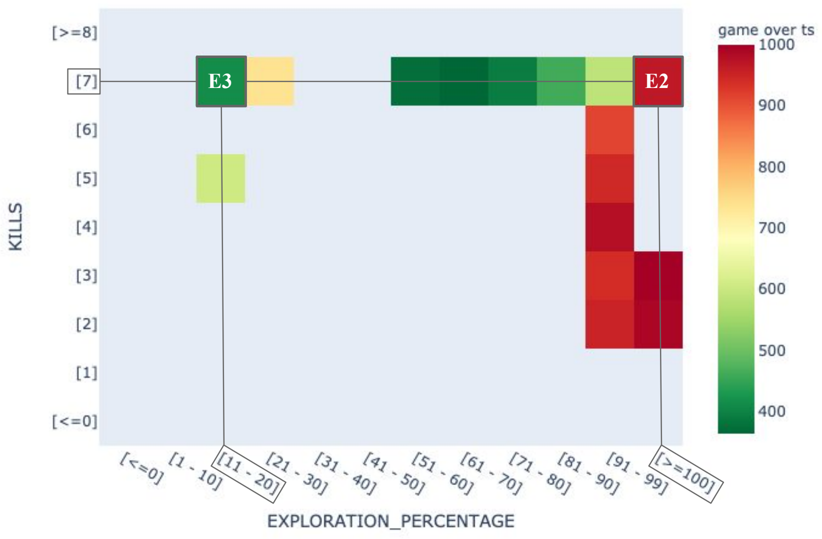


Figure 7.15: *Sheriff behaviour-type* agents identification: *High killer and explorer*, with the highest value pair of exploration rate and average of kills, (E2) and *High killer and low explorer* (E3), with the highest number of kills and lowest exploration rate.

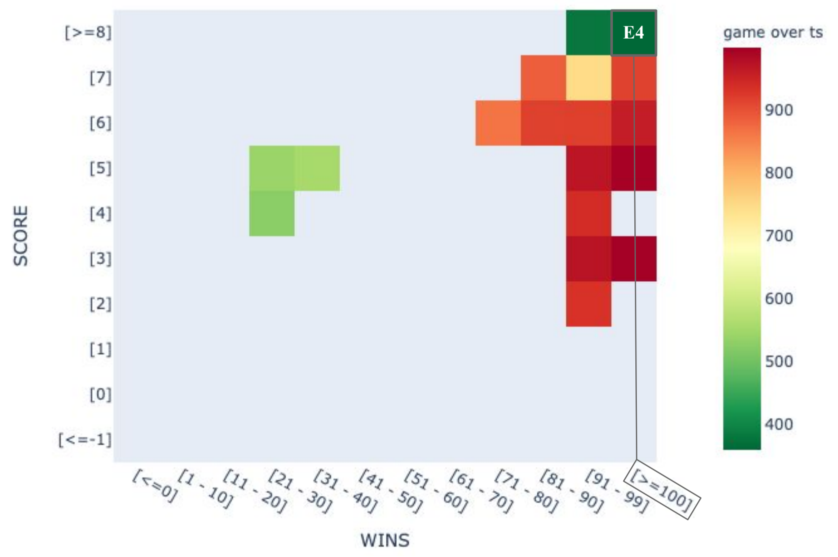


Figure 7.16: *Sheriff behaviour-type* agent identification: *Speed-runner* (E4), with 100% win rate and one of the lowest EoG ticks.

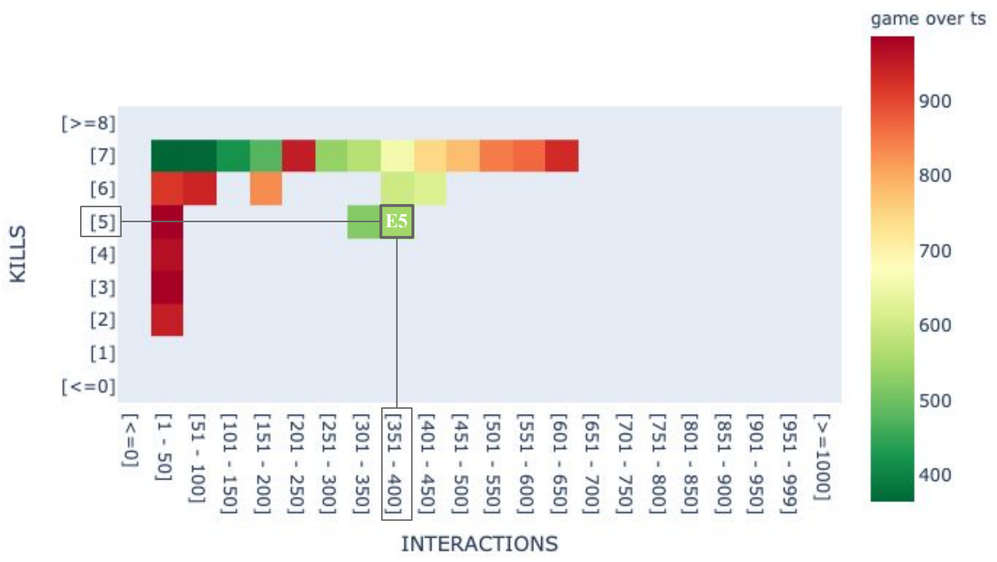


Figure 7.17: *Sheriff behaviour-type* agent identification: *Barrels shooter* (E5), with a high rate of interactions and hits but a mid-range rate of kills, so should target the barrels instead of the bandits.

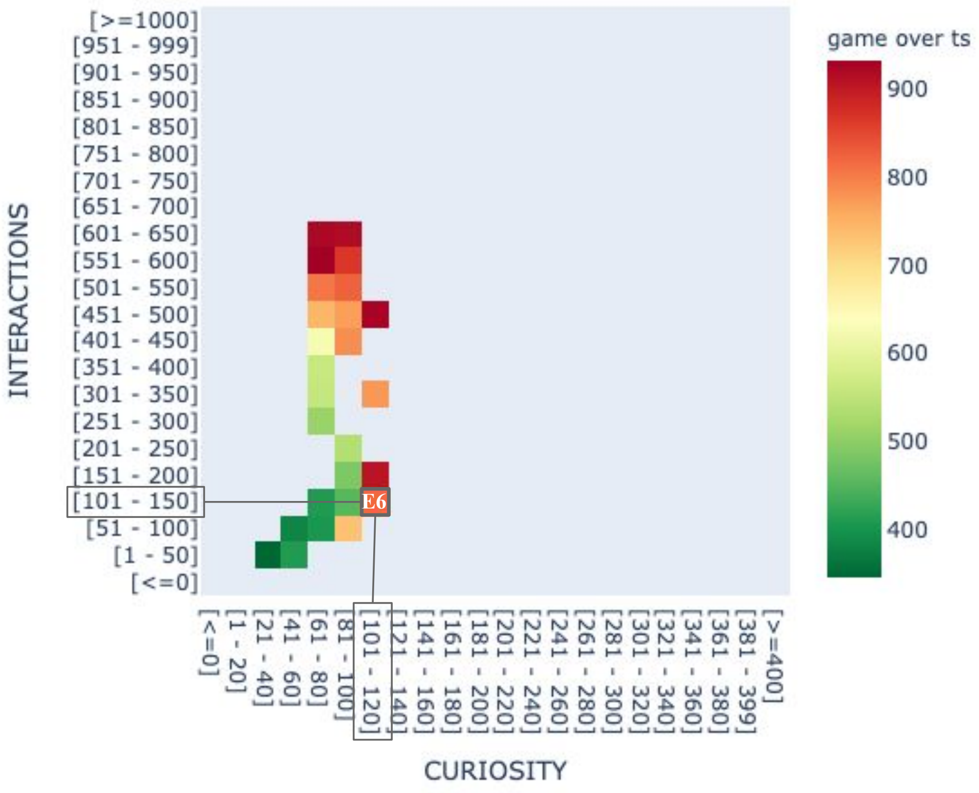


Figure 7.18: *Sheriff behaviour-type* agent identification: *High curiosity and low interactions* (E6), with many interactions but a high curiosity, so when it interacts with elements, it should do so in different positions and move on.

7.3 Experiment: Portability of the Agents

I carry out an experiment in each game to determine the portability of the *behaviour-type* agents identified and detailed in Section 7.2. I propose that if an agent presents similar stats or a similar tendency across different levels, it suggests that their strength is carried between them. Therefore, that agent can be used with similar expectations independently of the level used to generate it.

The experiments I have carried out in each game to determine the portability of the agents identified are covered in separate sections. I present four new levels for each game that have a similar size to the original one. The idea behind creating these new levels is not to modify the game or its rules but to present different characteristics and distribution of the elements to study how each of the agents behaves in them. I run each of the agents a total of 100 times in each of the levels to obtain the gameplay stats and compare them to their corresponding features. I also run the agents in the original level (Fig. 7.2) again to obtain the stats that serve as guidance. The corresponding code can be found in a secondary branch of the Github repository [Guerrero Romero, 2021a] and the config files, executables, and results can be found in an OSF repository [Guerrero Romero, 2021c].

Appendix E includes the tables with all the resulting averages of stats per game, agent, and level. The following sections include a unique table per game that summarises the results relevant to each *behaviour-type* agent. The table also provides pertinent information about each of the levels as a reference. This information helps to put the resulting values into perspective. For each feature I have used to identify agents in each game, a graph detailing the results per level of the 100 play-throughs of the corresponding agent (or agents) is included. The scripts used to generate these graphs can be found in a Github repository [Guerrero Romero, 2021b].

7.3.1 Portability experiment in *Butterflies*

New levels description The four new levels have the same size but a different number of butterflies, cocoons, and trees. They present different distributions of the elements and shapes of the areas. The two main game elements that can affect how it is played are the butterflies and cocoons. The distance between the butterflies, cocoons, and the player is relevant for the gameplay, as it can influence how quickly the game is won (or lost). Therefore, I design these levels by providing different distributions between these elements. Some of the levels are more open than the original one (7.19a and 7.19c), provided by the distribution of the trees (walls), which also affect how the player or NPCs can move around and access the distinct elements. Level 7.19b is horizontally symmetric; the butterflies are in the central area, most of the cocoons on the left side, and the player on the right side. Level 7.19d is designed to have the player and butterflies and cocoons in differentiated areas, with only one access point between them. As a result, the characteristics of this level are very different from the rest of them. While I expect to observe some discrepancies in the stats of this level, the overall tendency between agents should still uphold and the agents should still show traces of their proficiency.

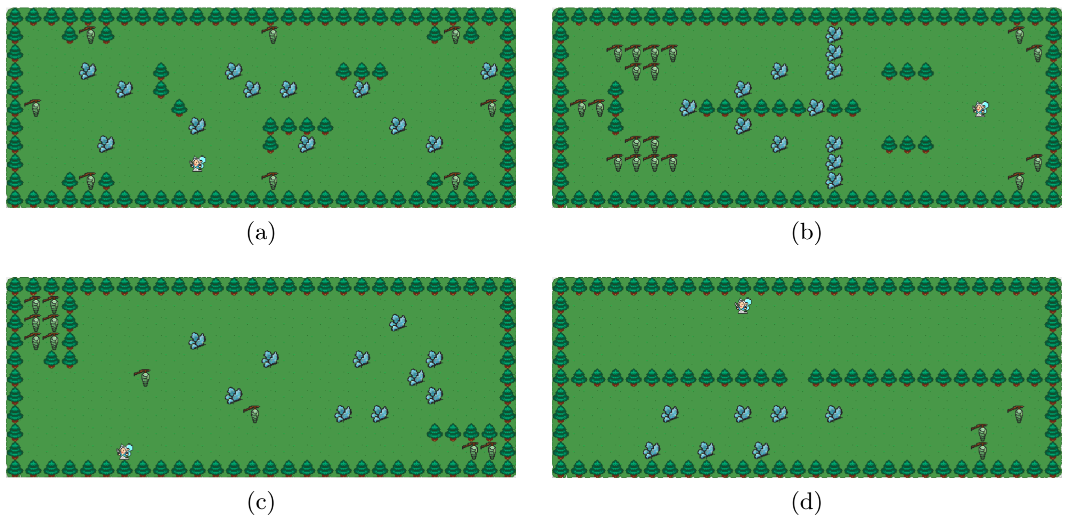


Figure 7.19: New levels used for the *team* portability experiments in *Butterflies*.

Portability results All the resulting average of stats of the play-through of the agents, classified into level and agent, are included in Appendix E.1. The results are summarised in Table 7.6. A graph is included for each of the features relevant to the agents and their selection: *collisions* (Fig. 7.20), *curiosity* (Fig. 7.21), *EoG* (Fig. 7.22), *exploration* (Fig. 7.23), *score* (Fig. 7.24), and *wins*, represented as the total number of *victories* (Fig. 7.25). It shows the results of the 100 play-throughs per level of the corresponding agent (or agents).

By analysing the data, I notice a trend on the stats confirming that most of the agents are portable between levels. This trend is quite clear for the agents related to exploration (E5 and E6), curiosity (E3 and E6) and winning (E4). For the latter, the win rate and exploration drops in level 7.19d compared to the other ones. However, by looking at the overall winning stats across the agents in this level (Table 7.7, Fig 7.26), E4 does indeed achieves the highest rate of wins (89%), compared to the rest: 85% (E1), 9% (E2), 4% (E3), 8% (E5), and 5% (E6). E4 also shows the lowest average of EoG ticks (61.13), vs 78.68, 959.46, 862.94, 793.78, and 860.37, respectively. Similarly, E5 showcases one of the highest average of exploration rates (74.56%) compared to the rest of the agents (Table 7.7, Fig 7.27): 32.25% (E1), 78.05% (E2), 42.82% (E3), 25.60% (E4), and 43.25% (E6). Therefore, the strength of these agents (winning the game quickly and achieving high exploration, respectively) is still transferred to level 7.19d, even when the portability is not noticeable at first sight. Level 7.19d is also the most different one, as there are two differentiated areas with a unique point of access between them. Therefore, the agents need to find that access point to be able to collect the butterflies and win. There are only 3 cocoons, and the butterflies are close to these, so in this level, the game may end before the agent can even reach the area to win the game or explore it.

Regarding the agents related to score, either high (E1) or low (E2), the resulting stats implies that they are not as easily transferable between levels as the other agents (Fig 7.24). In *Butterflies*, the score depends on the disposition of the butterflies (clustered or dispersed) and their distance to the player. I believe it is hard to get a fair comparison between levels when the score is on different scales and depends on factors tight to the game rules and the distribution of the elements.

Butterflies					
Features	**7.2a**	**7.19a**	**7.19b**	**7.19c**	**7.19d**
Max. score	64	38	58	38	18
Total walls	102	93	91	83	99
E1: *Low scorer*					
Score	20.44 (8.70)	29.58 (1.99)	43.56 (10.54)	21.2 (1.42)	12.64 (4.12)
E2: *High scorer*					
Score	47.76 (16.60)	30.78 (1.90)	51.1 (6.21)	22.58 (1.24)	11.86 (3.99)
E3: *High curiosity*					
Curiosity	116.53 (25.03)	98.99 (8.81)	116.38 (18.25)	82.13 (7.66)	76.38 (30.97)
Collisions	1436.82 (500.53)	1754.85 (160.49)	1362.33 (603.94)	1780.46 (133.97)	715.56 (708.63)
E4: *Speed-runner*					
Victories	100/100	100/100	99/100	100/100	89/100
EoG	102.15 (47.83)	103.49 (34.69)	114.38 (42.03)	73.61 (35.66)	61.13 (30.34)
E5: *High explorer*					
Exploration	99.29% (4.78)	99.30% (5.18)	99.33% (5.77)	99.77% (1.87)	74.56% (27.15)
E6: *Walls interaction*					
Exploration	67.43% (16.16)	60.28% (8.26)	68.15% (10.06)	44.60% (5.74)	43.25% (15.00)
Curiosity	114.59 (27.54)	96.53 (11.28)	120.46 (17.41)	79.86 (9.37)	75.16 (28.25)

Table 7.6: Overview of the portability results for *Butterflies*. *Victories* are the total number of wins in the 100 play-throughs. The other dimensions are the average of the resulting gameplay stats and the corresponding *Std. Deviation.* The stats shown per agent are the ones related to their proficiency. EoG is 2000.

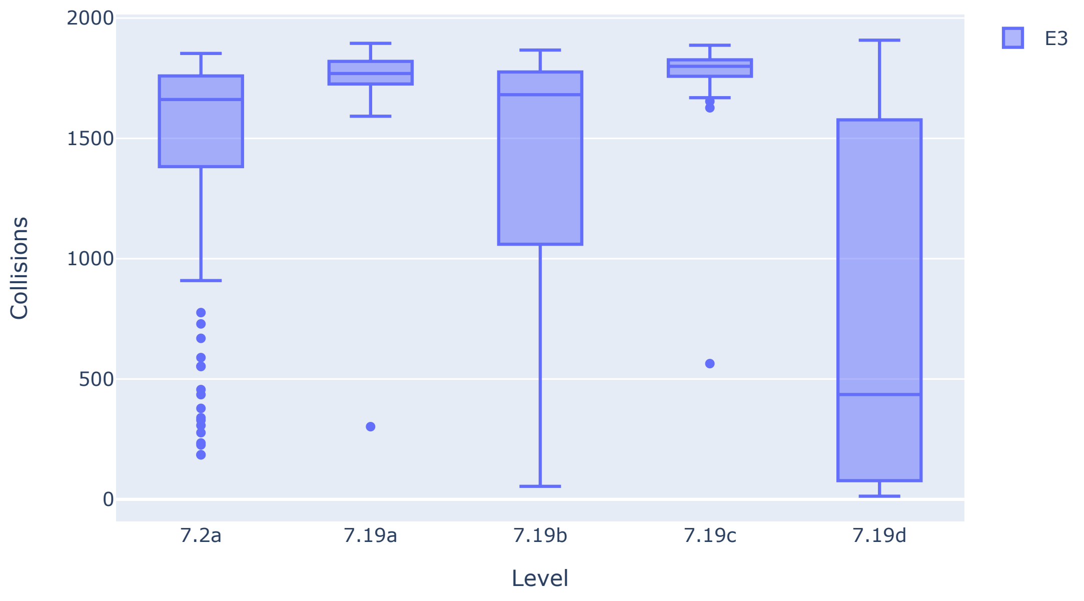


Figure 7.20: *Butterflies*: Resulting *collisions* per level, from 100 gameplays. E3: *High Curiosity*.

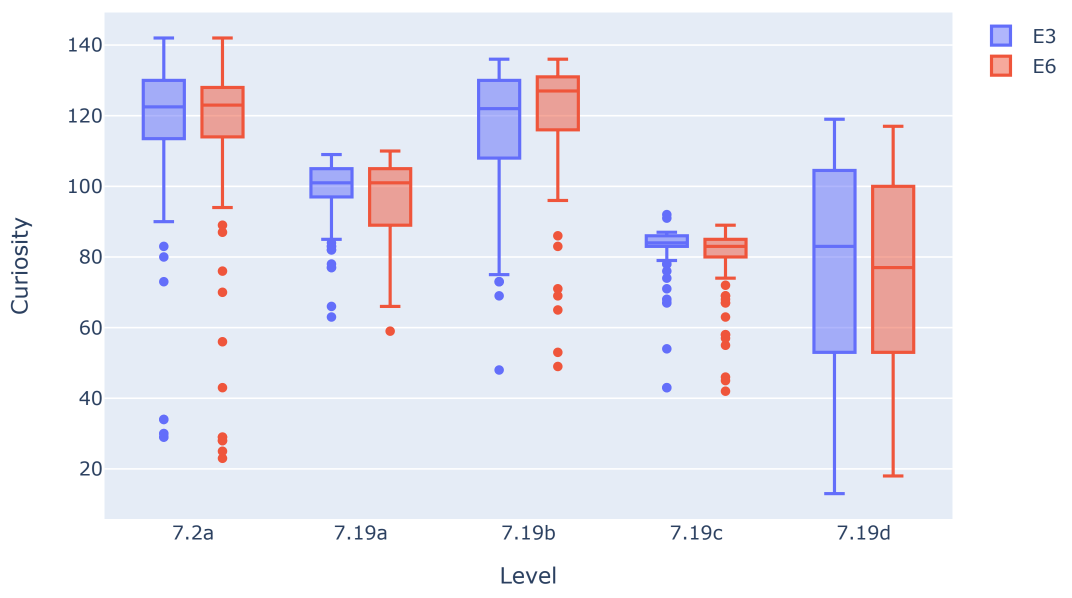


Figure 7.21: *Butterflies*: Resulting *curiosity* per level, from 100 gameplays. E3: *High curiosity*, E6: *Walls interaction*.

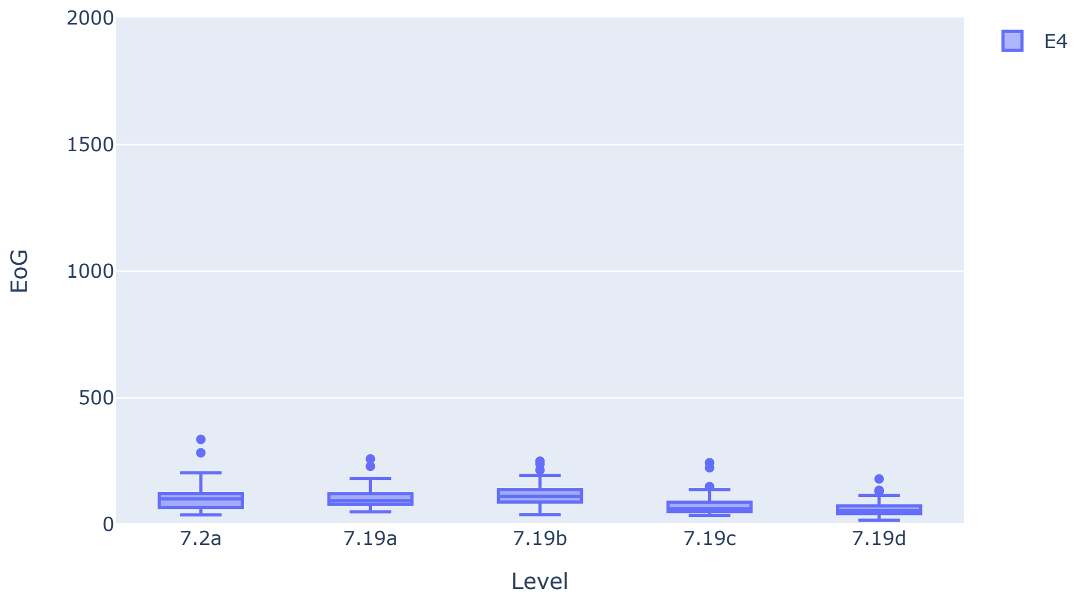


Figure 7.22: *Butterflies*: Resulting *EoG* per level, from 100 gameplays. E4: *Speed-runner*.

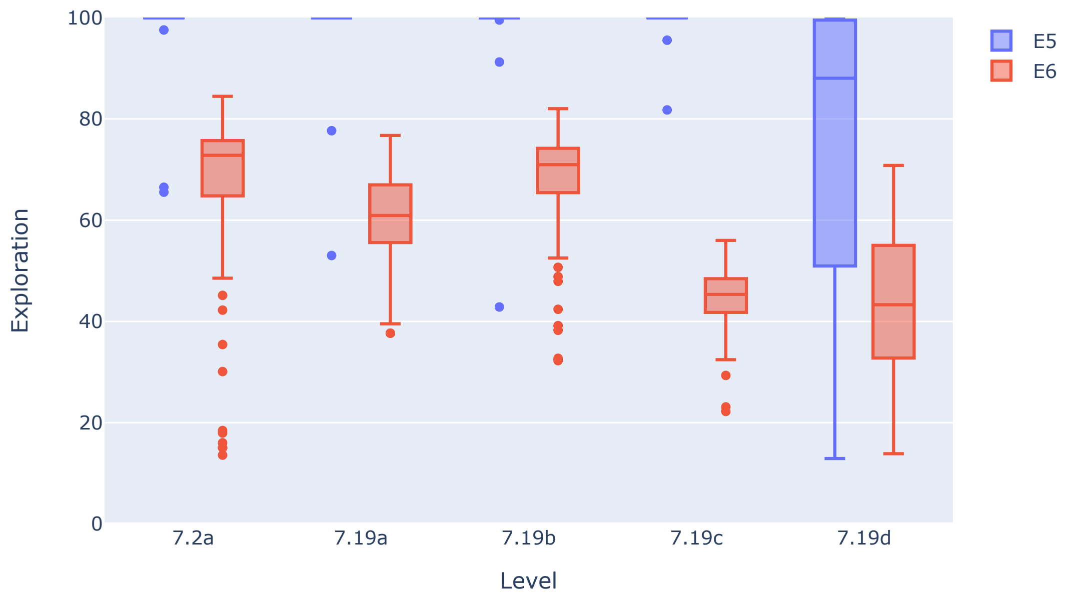


Figure 7.23: *Butterflies*: Resulting *exploration* per level, from 100 gameplays. E5: *High explorer*, E6: *Walls interaction* (with mid-range exploration).

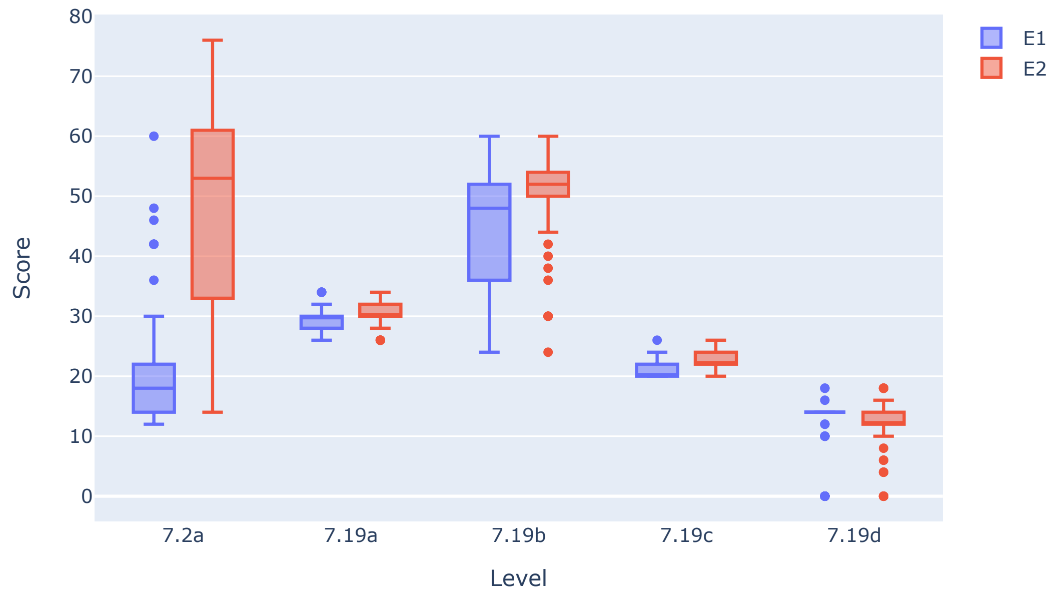


Figure 7.24: *Butterflies*: Resulting *score* per level, from 100 gameplays. E1: *Low scorer*, E2: *High scorer*.

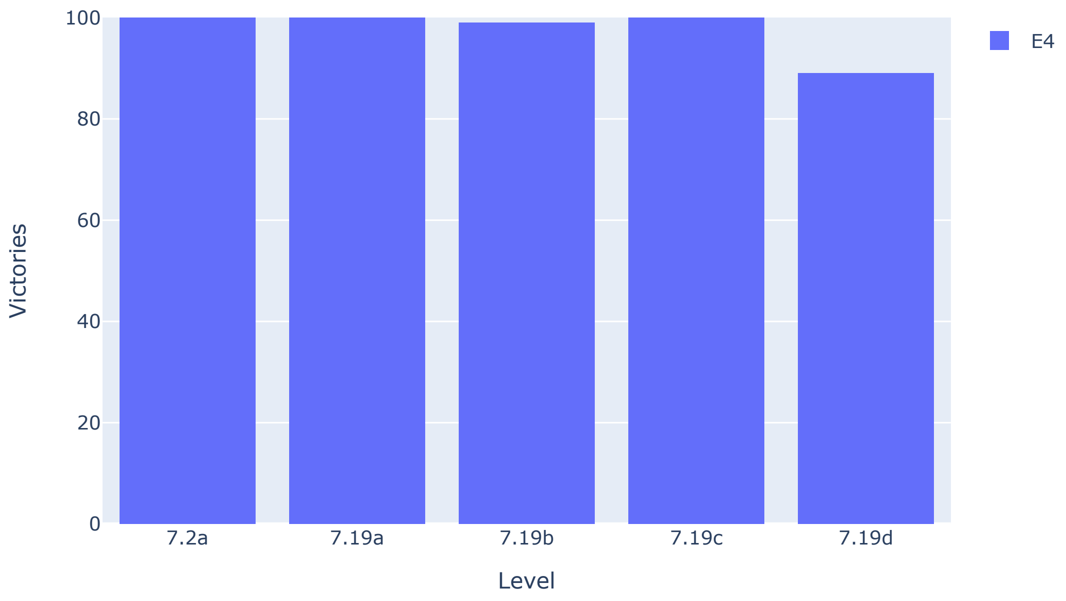


Figure 7.25: *Butterflies*: Resulting total number of *victories* per level, from 100 gameplays. E4: *Speed-runner*.

Level 7.19d							
Info							
Butterflies	7						
Cocoons	3						
Open locations	209						
Walls (trees)	99						
Gameplay Results							
Features	**Max**	**E1**	**E2**	**E3**	**E4**	**E5**	**E6**
EoG	**2000**	78.68	959.46	862.94	61.13	793.78	860.37
Win rate	**-**	85%	9%	4%	89%	8%	5%
Score	**18**	12.64	11.86	11.22	13.28	12.54	11.24
Exploration %	**-**	32.25%	78.05%	42.82%	25.60%	74.56%	43.25%
Unique interactions	**-**	1.48	1.94	1.91	1.8	1.97	1.96
Curiosity	**-**	5.89	74.79	76.38	6.66	73.98	75.16
Collisions	**-**	6.25	254.69	715.56	6.97	258.09	708.53
Interactions	**-**	6.25	254.69	715.56	6.97	258.09	708.53

Table 7.7: *Butterflies*: Per-agent portability results for Level 7.19d. E1: *Low scorer*, E2: *High scorer*, *E3: High curiosity*, E4: *Speed-runner*, E5: *High explorer*, E6: *Walls interaction.*

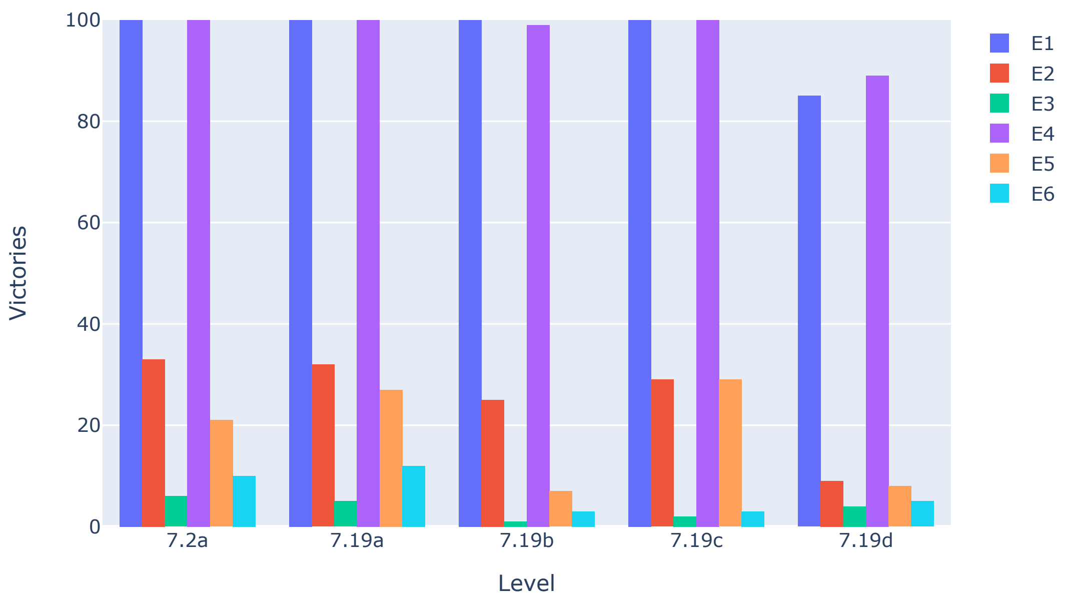


Figure 7.26: *Butterflies*: Resulting total number of *victories* per level, from 100 gameplays. Comparison between all agents. E1: *Low scorer*, E2: *High scorer*, E3: *High curiosity*, **E4: *Speed-runner***, E5: *High explorer*, E6: *Walls interaction.* Although the rate of victories drops in Level 7.19d for E4, it still achieves a significantly higher rate of wins in comparison.

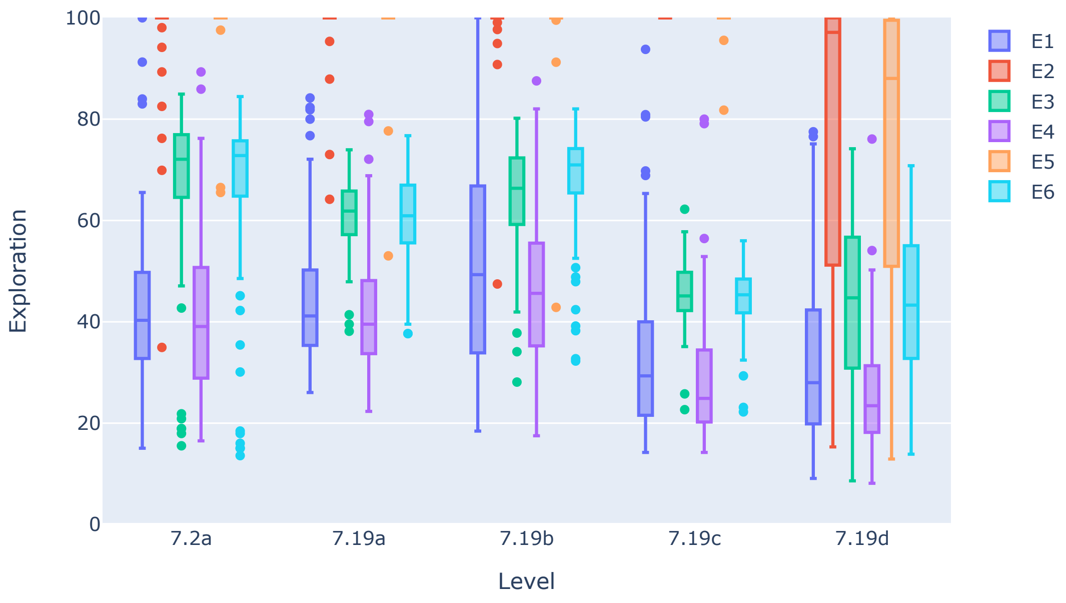


Figure 7.27: *Butterflies*: Resulting *exploration* per level, from 100 gameplays. Comparison between all agents. E1: *Low scorer*, E2: *High scorer*, E3: *High curiosity*, E4: *Speed-runner*, **E5: *High explorer***, E6: *Walls interaction*. Although the exploration drops in Level 7.19d and is more variable for E5, it still achieves a significantly overall higher percentage in comparison.

7.3.2 Portability experiment in *Zelda*

New levels description The four new levels have a similar size to the original one and maintain the number of monsters (except 7.28c, where I include two additional ones). The primary motivation behind the creation of each of these levels is to study the behaviour of the agents when the characteristics of the dungeon and the size of the rooms that comprise it changes. The main difference between these levels comes from their shape given by the distribution of the walls. Levels 7.28a and 7.28c show wide open spaces, the latter representing a unique room with dispersed blocks in the middle and in the area surrounding the key. Levels 7.28b and 7.28d, on the other hand, have a lot of walls. The former has narrow corridors and the latter three big rooms connected by one access point between them. In all the levels, the key is accessible but at different distances from the player and the exit door. It is required for the player to move around the map to finish the game. It is expected that each of the agents, independently of how the rooms are shaped, can play the game and carry out their expected behaviour.

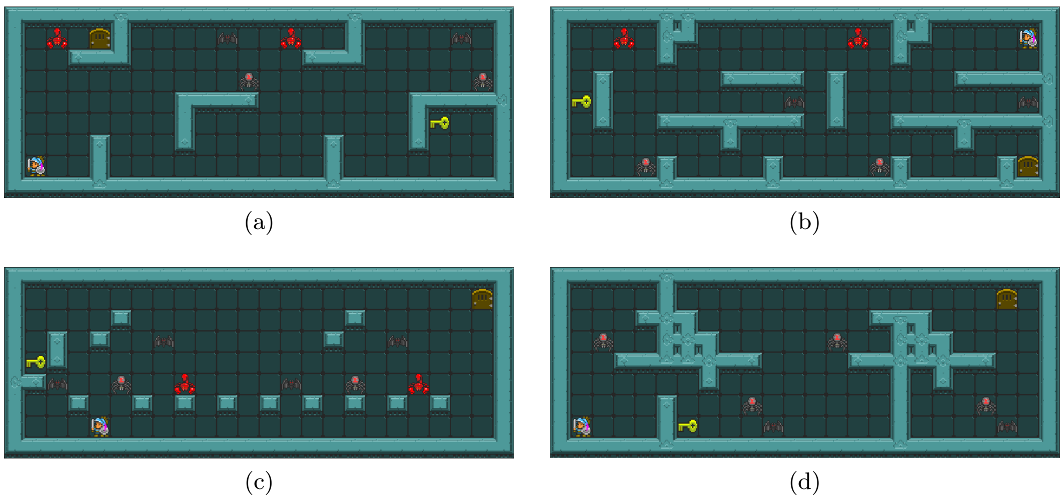

(a) (b) (c) (d)

Figure 7.28: New levels used for the *team* portability experiments in *Zelda*.

Portability results All the resulting average of stats of the play-through of the agents, classified into level and agent, are included in Appendix E.2. The results are summarised in Table 7.8. A graph is included for each of the features relevant to the agents and their selection: *EoG* (Fig. 7.29), *exploration* (Fig. 7.30), *kills* (Fig. 7.31), *score* (Fig. 7.32), and *wins*, represented as the total number of *victories* (Fig. 7.33). It shows the results of the 100 play-throughs per level of the corresponding agent (or agents).

Looking at the results, all the agents selected for this game seem to be transferable between levels. For the *High scorer* (E1), the final score seems to be similar through all levels, increasing accordingly to the additional monsters in level 7.28c. I believe these results come from the characteristics of the game and the fact that I use a similar number of the elements involved in its calculation. The number of kills increase and decrease relative to the characteristics of the agents related to that feature: E5 and E6 achieve a high number of kills across all levels, and E4, on the other hand, obtains a comparably low number. I observe similar results for exploration, where agents expected to obtain a high percentage (E3 and E5) achieve, in average, more than 93.33% in every level. In comparison, the *Low explorer* (E6) does not achieve more than 54.79%.

Taking a look at the *Speed-runner* (E2), it is unclear at first sight if the tendency is maintained throughout the levels. However, when comparing the overall wins and EoG ticks with the rest of the agents, I observe that it actually does. Regarding the number of victories achieved by the agents (Fig.7.34), most of them achieve a rate of wins less than 13%, while E2 achieves >= 65% in all games. Therefore, I can conclude that, even in those levels where it seems to be more challenging to win, the agent is showcasing its proficiency in achieving victory. Similarly, looking at the results for the EoG ticks (Fig. 7.35), E2 obtains significantly lower EoG ticks across levels than the other agents, even when these are objectively high, like in levels 7.28a and 7.28b.

Zelda					
Features	**7.2b**	**7.28a**	**7.28b**	**7.28c**	**7.28d**
Max score	14	14	14	18	14
Total enemies	6	6	6	8	6
E1: *High scorer*					
Score	12.25 (2.64)	12.11 (2.71)	11.7 (3.21)	16.44 (2.73)	11.55 (3.40)
E2: *Speed-runner*					
Victories	90/100	76/100	65/100	89/100	96/100
EoG	617.18 (487.03)	1187.98 (566.51)	1242.26 (652.64)	689.03 (526.05)	450.86 (331.35)
E3: *High explorer*					
Exploration	98.03% (7.80)	95.36% (15.00)	95.25% (14.70)	98.31% (6.93)	93.33% (17.73)
E4: *Low killer*					
Kills	1.62 (1.37)	3.25 (1.44)	2.97 (1.51)	3.53 (2.03)	1.79 (1.34)
E5: *High killer and explorer*					
Kills	5.68 (0.98)	5.66 (0.85)	5.44 (1.18)	7.71 (0.75)	5.41 (1.23)
Exploration	97.19% (10.81)	97.93% (7.04)	94.71% (15.07)	98.61% (5.93)	94.00% (15.27)
E6: *High killer and low explorer*					
Kills	4.78 (1.04)	5.17 (0.80)	4.21 (1.19)	7.26 (1.30)	3.77 (1.33)
Exploration	48.38% (11.54)	50.66% (13.73)	42.25% (15.96)	54.79% (15.34)	35.73% (14.71)

Table 7.8: Portability results for *Zelda*. *Victories* are the total number of wins in the 100 play-throughs. The other dimensions are the average of the resulting gameplay stats and the corresponding *Std. Deviation*. The stats shown per agent are the ones related to their proficiency. EoG is 2000.

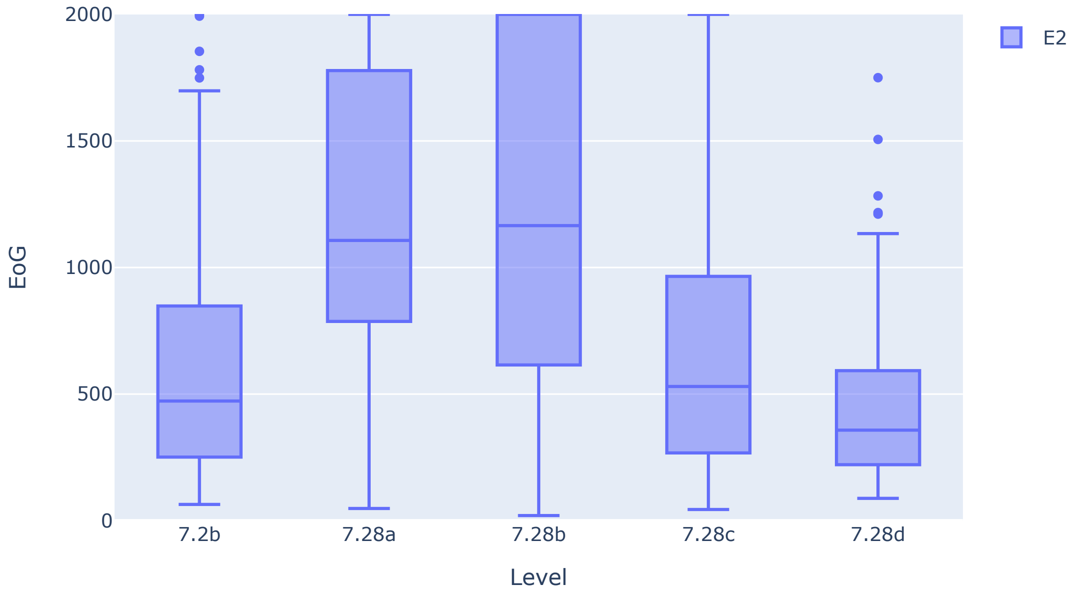


Figure 7.29: *Zelda*: Resulting *EoG* per level, from 100 gameplays. E2: *Speed-runner*.

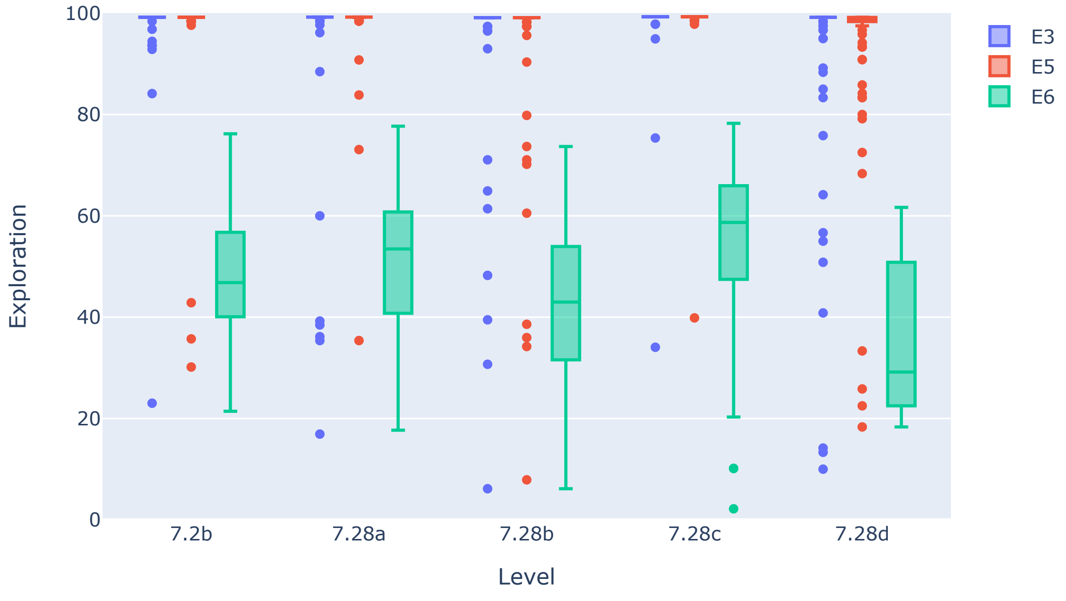


Figure 7.30: *Zelda*: Resulting *exploration* per level, from 100 gameplays. E3: *High explorer*, E5: *High killer and explorer*, E6: *High killer and low explorer*.

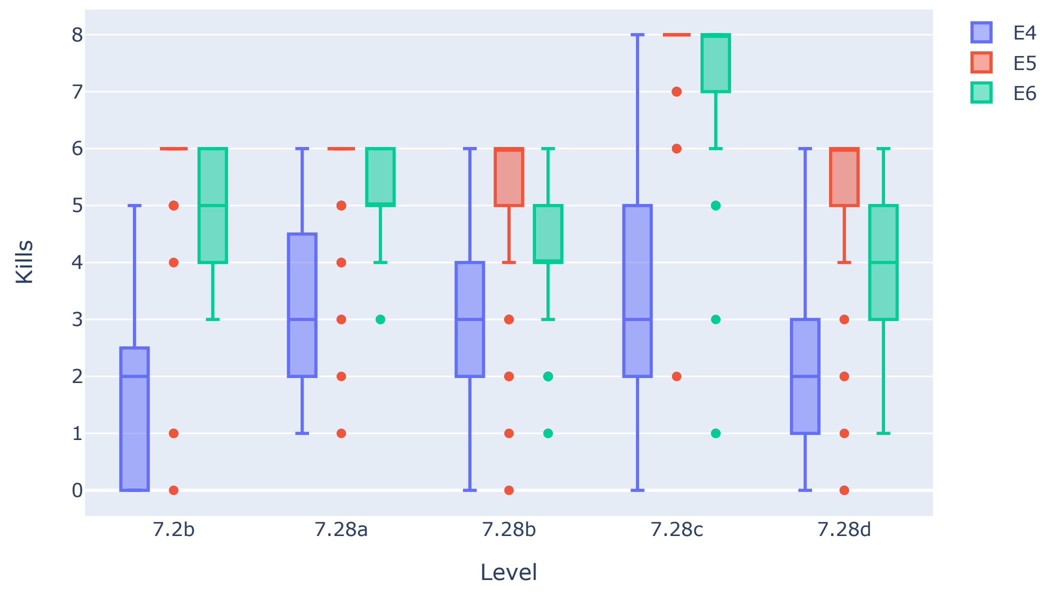


Figure 7.31: *Zelda*: Resulting *kills* per level, from 100 gameplays. E4: *Low killer*, E5: *High killer and explorer*, E6: *High killer and low explorer*.

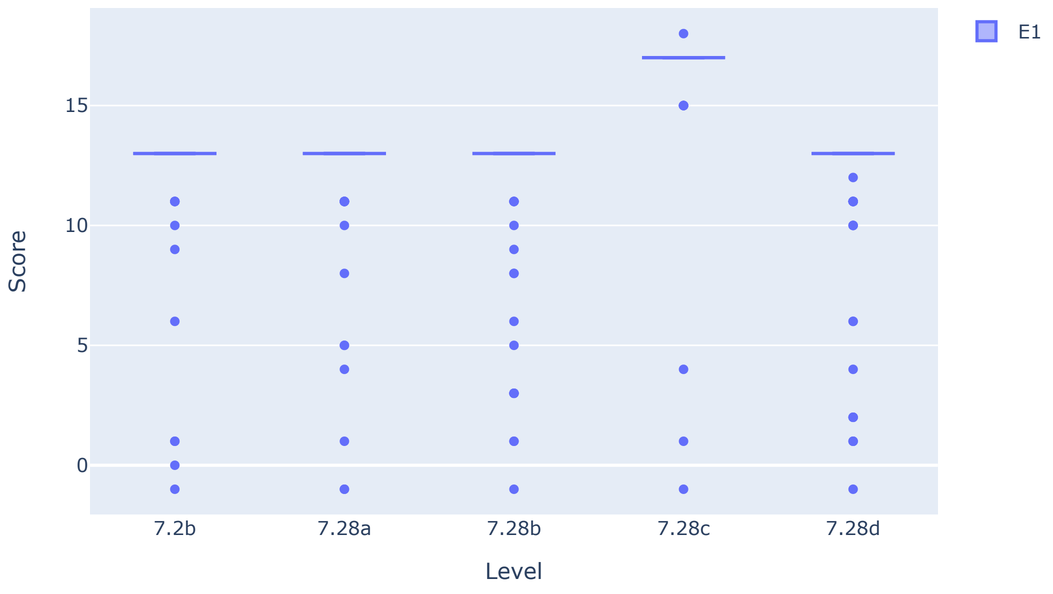


Figure 7.32: *Zelda*: Resulting *score* per level, from 100 gameplays. E1: *High scorer*.

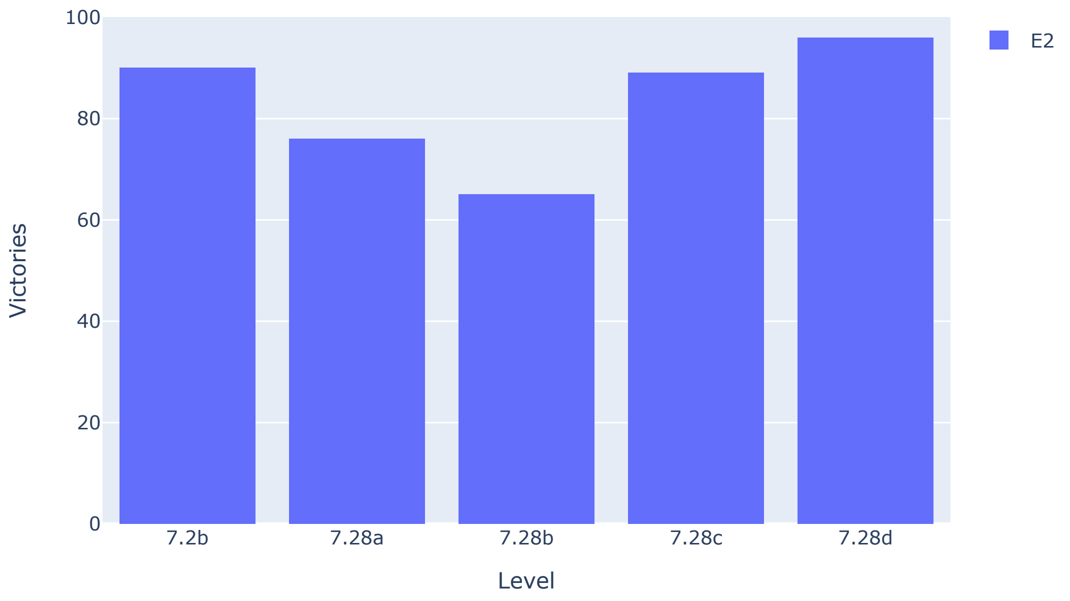


Figure 7.33: *Zelda*: Resulting total number of *victories* per level, from 100 gameplays. E2: *Speed-runner*.

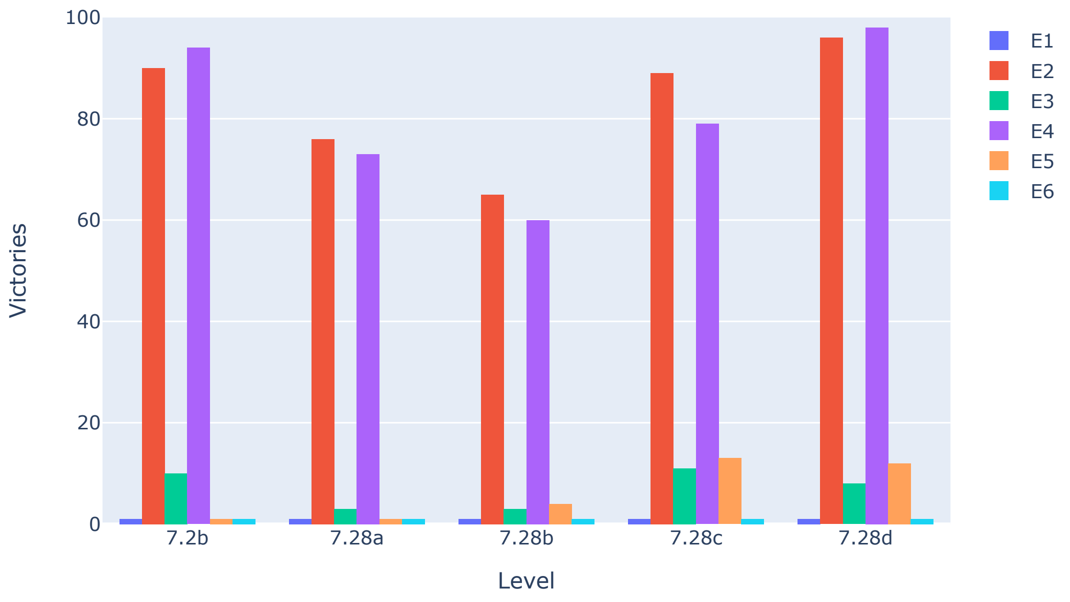


Figure 7.34: *Zelda*: Resulting total number of *victories* per level, from 100 gameplays. Comparison between all agents. E1: *High scorer*, ***E2: Speed-runner***, E3: *High explorer*, E4: *Low killer*, E5: *High killer and explorer*, E6: *High killer and low explorer*. Although the rate of wins drops below 80% in some levels, E2 still achieves a significantly high number of victories in comparison to most of the other agents.

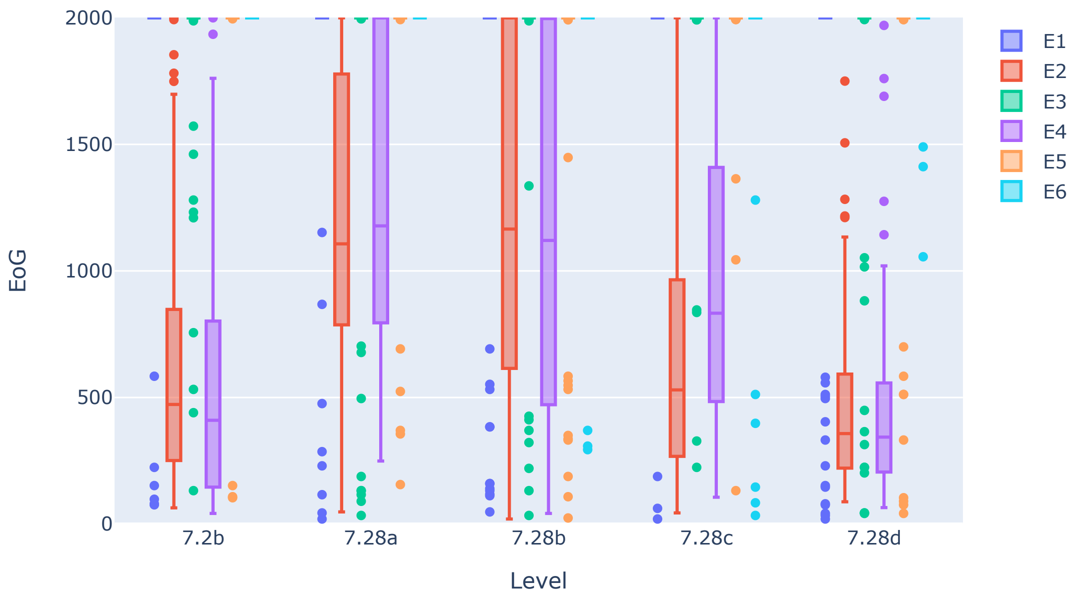


Figure 7.35: *Zelda*: Resulting total number of *EoG* per level, from 100 gameplays. Comparison between all agents. E1: *High scorer*, ***E2: Speed-runner***, E3: *High explorer*, E4: *Low killer*, E5: *High killer and explorer*, E6: *High killer and low explorer*. Although the EoG ticks obtained by E2 is, in average, higher that we would expect for this *behaviour-type* agent, it is still lower when compared to the other agents.

7.3.3 Portability experiment in *Digdug*

New levels description The size of the four new levels is the same, but all the elements that constitute the game provide differences between them. Level 7.36a contains breakable walls covering most of the level and three spawners. However, it does have very few items. Most of the items and monsters are reachable without having to break the walls. Level 7.36b contains a high density of walls but with an open area in the middle that separates them. In this level, the player is *trapped* in the walls so it needs to break them to access the open area and most of the items. Level 7.36c has a heterogeneous distribution of walls and open spaces, with a few monsters and items. Without breaking any walls, the player is only able to kill the monsters. Although there are no many items, it is required to break thin walls to reach them. Level 7.36d is the most different one, as the walls surround the level and separate two big open areas. One of these areas is where the player starts the game, while the other contains a high number of items and monsters. The player does not need to break walls to access the big open area. The motivation behind the characteristics of these new levels was providing a diversity of possibilities that are brought by the rules of the game. At the same time, we expect the agents and their behaviour to still work across them.

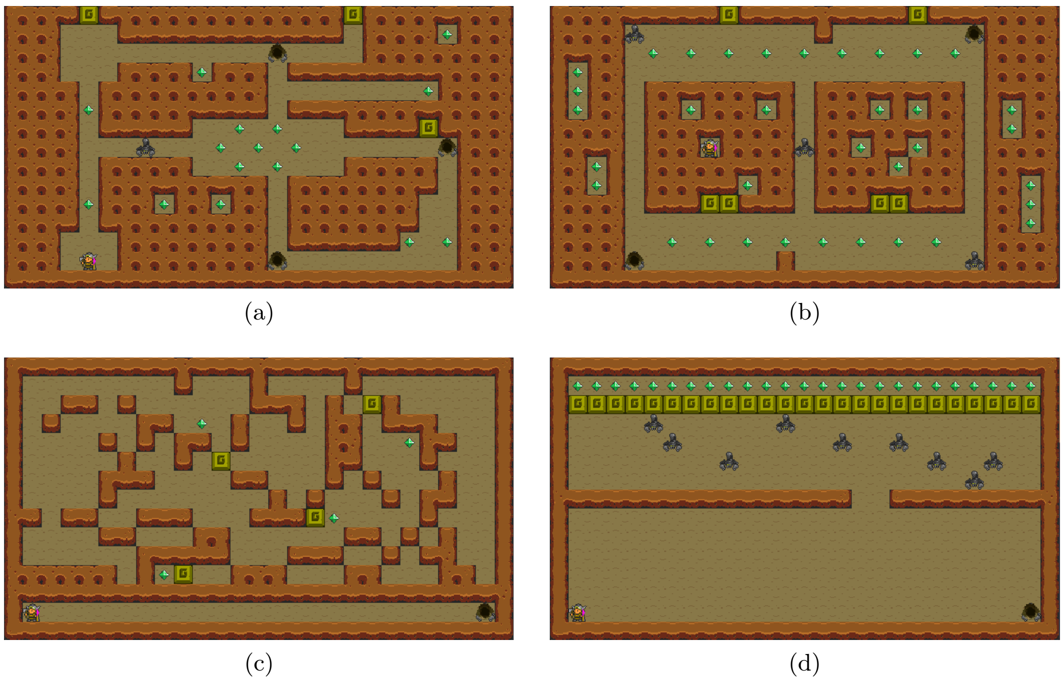


Figure 7.36: New levels used for the *team* portability experiments in *Digdug*.

Portability results All the resulting average of stats of the play-through of the agents, classified into level and agent, are included in Appendix E.3. The results are summarised in Table 7.9. A graph is included for each of the features relevant to the agents and their selection: *curiosity* (Fig. 7.37), *exploration* (Fig. 7.38), *hits* (Fig. 7.39), *items* (Fig. 7.40), *kills* (Fig. 7.41), and *score* (Fig. 7.42). It shows the results of the 100 play-throughs per level of the corresponding agent (or agents).

For this game, all the agents identified seem to be transferable between levels. Those agents expected to collect a high number of items (E1 and E2) manage to do so, achieving an average very close to the maximum number of resources available. On the other hand, the agent selected for being a *Low collector* (E3) gathers at most a third of those resources. The stats of these agents also match with the corresponding number of kills expected of each of them. While E1 and E3 achieve a high number of kills, E2 does not. Therefore, these three agents with different skills in collecting items and killing enemies are transferable, and the trend of their results is maintained between the different levels.

I observe a similar tendency in the agents with features related to exploration: The *Walls breaker* (E4) is expected to achieve a mid-average exploration (<= 70.12%) and high curiosity, and it gets similar resulting stats in the new levels. The *High explorer* (E5) achieves between 98.51% and 100% exploration across the levels, and E6, on the other hand, reaches a maximum of 37.98% while achieving a high score, as expected.

Digdug					
Features	**7.2c**	**7.36a**	**7.36b**	**7.36c**	**7.36d**
Max score	44	48	61	14	53
Total enemies	12	16	13	5	14
Total items	27	19	41	8	50
Breakable walls	267	292	247	196	103
E1: *High collector and killer*					
Kills	10.64 (0.73)	13.74 (1.50)	11.56 (0.86)	3.88 (0.35)	12.68 (0.69)
Items	26.92 (0.30)	18.91 (0.28)	40.76 (0.83)	7.95 (0.41)	49.90 (0.62)
E2: *High collector and low killer*					
Kills	2.81 (1.50)	3.19 (1.70)	3.05 (1.65)	0.60 (0.68)	4.69 (1.59)
Items	27.00 (0.00)	19.00 (0.00)	40.99 (0.10)	8.00 (0.00)	50.00 (0.00)
E3: *Low collector and high killer*					
Kills	9.64 (1.77)	12.62 (3.57)	12.53 (0.77)	4.7 (0.87)	13.31 (0.97)
Items	9.14 (3.10)	5.05 (3.08)	11.36 (3.48)	1.72 (1.08)	9.74 (8.02)
E4: *Walls breaker*					
Exploration	70.12% (11.21)	67.66% (13.90)	67.70% (12.66)	64.73% (06.65)	46.98% (17.62)
Curiosity	464.6 (68.09)	463.77 (87.06)	434.05 (67.64)	358.54 (28.05)	249.87 (77.28)
Hits	269.65 (38.97)	282.32 (51.86)	245.53 (37.10)	197.70 (14.77)	120.67 (36.05)
E5: *High explorer*					
Exploration	99.98% (0.12)	100% (0.00)	99.99% (0.07)	100% (0.00)	98.51% (10.47)
E6: *Low explorer and high scorer*					
Exploration	30.92% (5.29)	26.49% (5.13)	36.32% (4.12)	26.96% (5.96)	37.98% (5.83)
Score	35.49 (5.60)	35.46 (6.04)	51.3 (4.25)	11.02 (2.51)	47.71 (9.44)

Table 7.9: Portability results for *Digdug*. The results are the average of the gameplay stats obtained in the 100 play-throughs. The values in parenthesis represent the corresponding *Std. Deviation*. The stats shown per agent are the ones related to their proficiency. EoG is 2000.

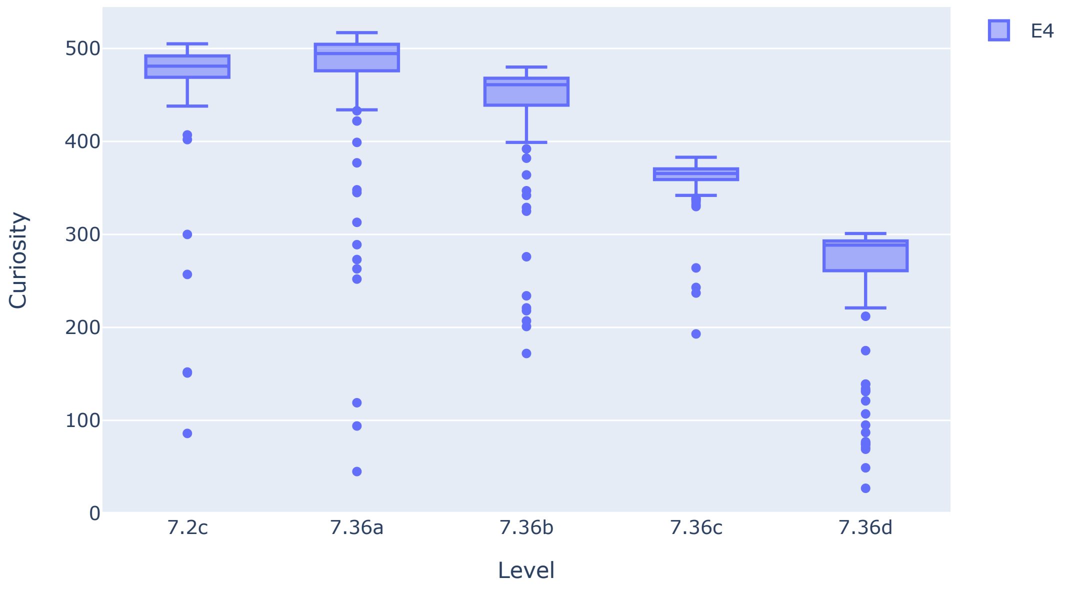


Figure 7.37: *Digdug*: Resulting *curiosity* per level, from 100 gameplays. E4: *Walls breaker*.

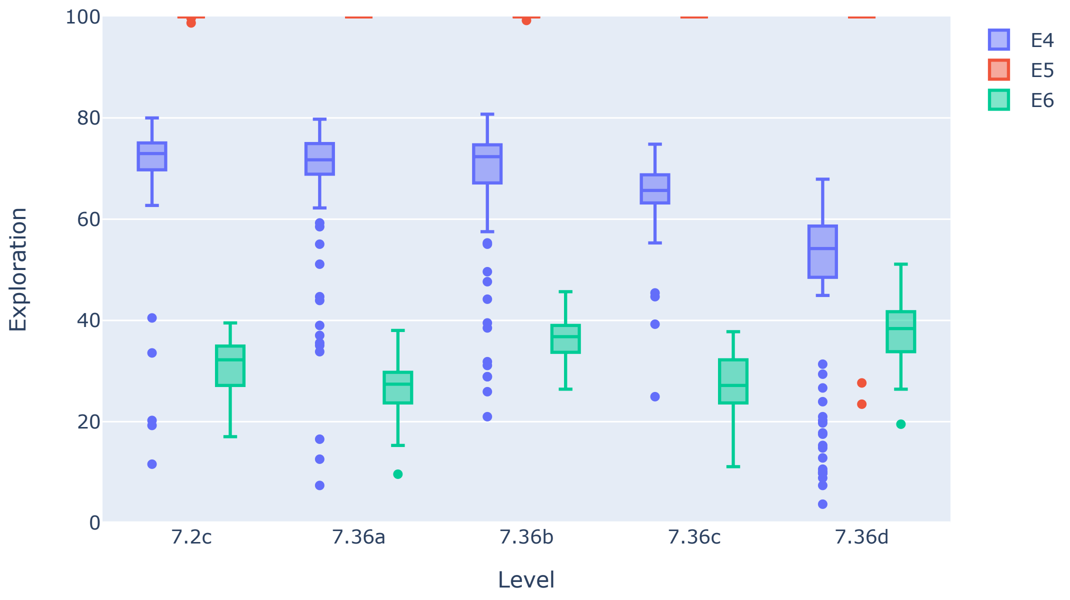


Figure 7.38: *Digdug*: Resulting *exploration* per level, from 100 gameplays. E4: *Walls breaker*, E5: *High explorer*, E6: *Low explorer and high scorer*.

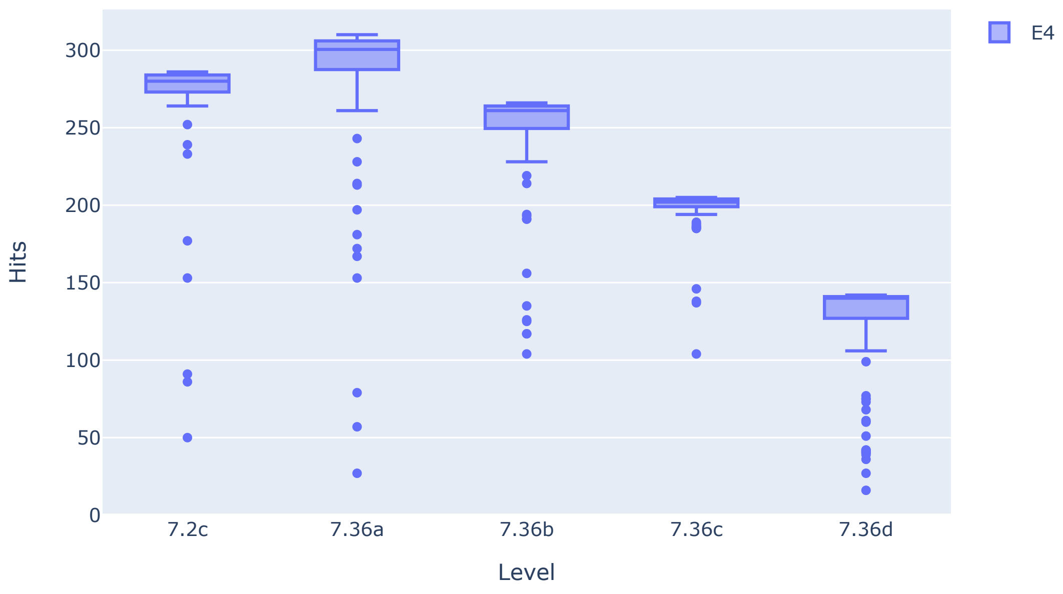


Figure 7.39: *Digdug*: Resulting *hits* per level, from 100 gameplays. E4: *Walls breaker*.

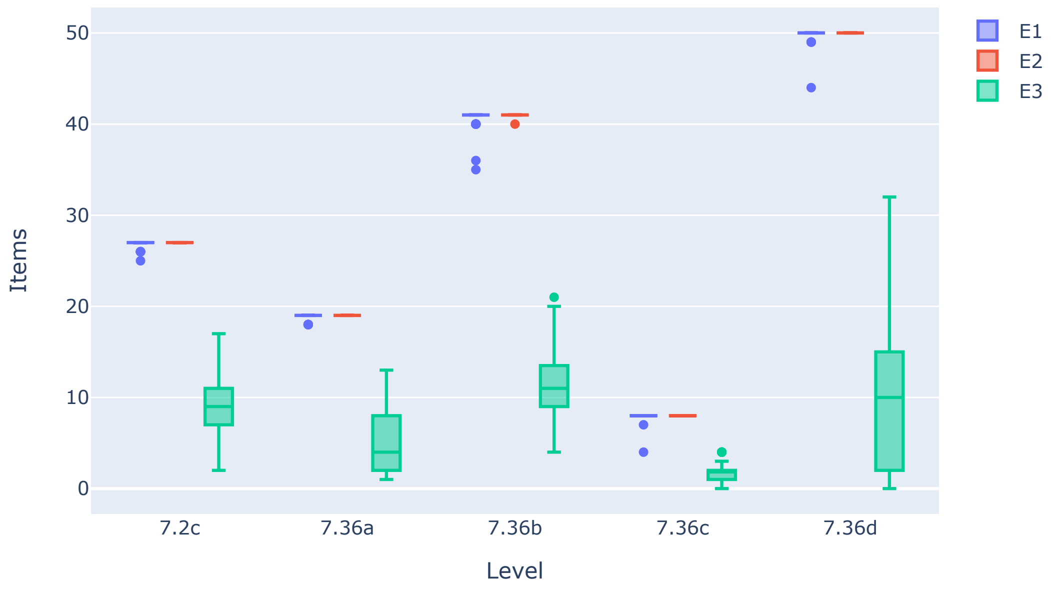


Figure 7.40: *Digdug*: Resulting *items* per level, from 100 gameplays. E1: *High collector and killer*, E2: *High collector and low killer*, E3: *Low collector and high killer*.

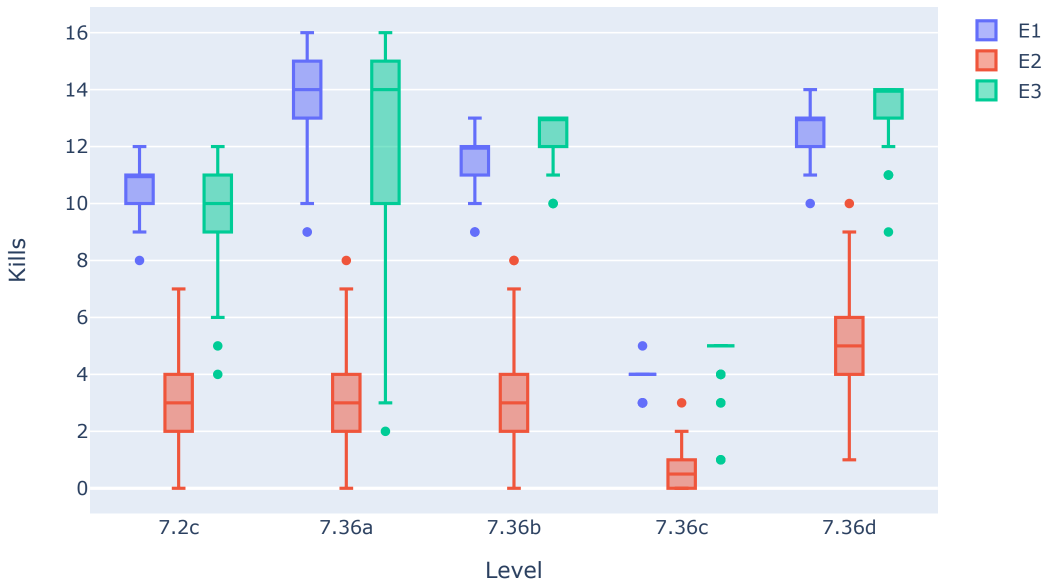


Figure 7.41: *Digdug*: Resulting *kills* per level, from 100 gameplays. E1: *High collector and killer*, E2: *High collector and low killer*, E3: *Low collector and high killer*.

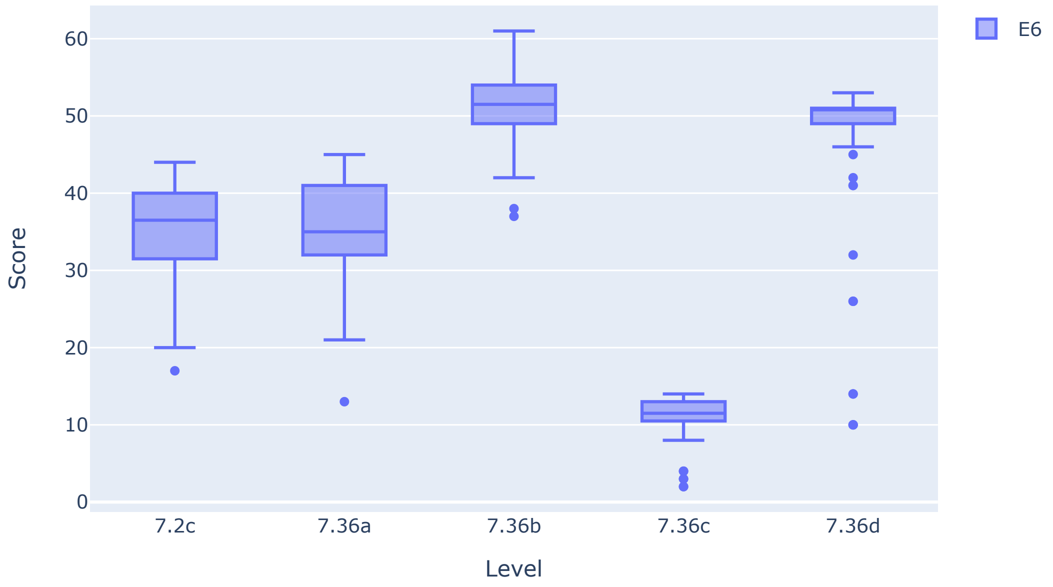


Figure 7.42: *Digdug*: Resulting *score* per level, from 100 gameplays. E6: *Low explorer and high scorer*.

7.3.4 Portability experiment in *Sheriff*

New levels description The shape of the game is similar on the four new levels, and the number of bandits varies between 9 and 11. Given the rules of the game and its mechanics, not many elements can be changed in the new levels. The main difference comes from the number and distribution of the barrels that serve as cover for the player. In levels 7.43a and 7.43b, the barrels are distributed around the borders. The difference between these two levels is the number of overall barrels. While the former has 65 barrels, the latter barely has any (27). In levels 7.43c and 7.43d, the barrels are also distributed in the middle of the map instead of only around the borders. The agents are expected to present their behaviour still, even when the number or distribution of the barrels varies.

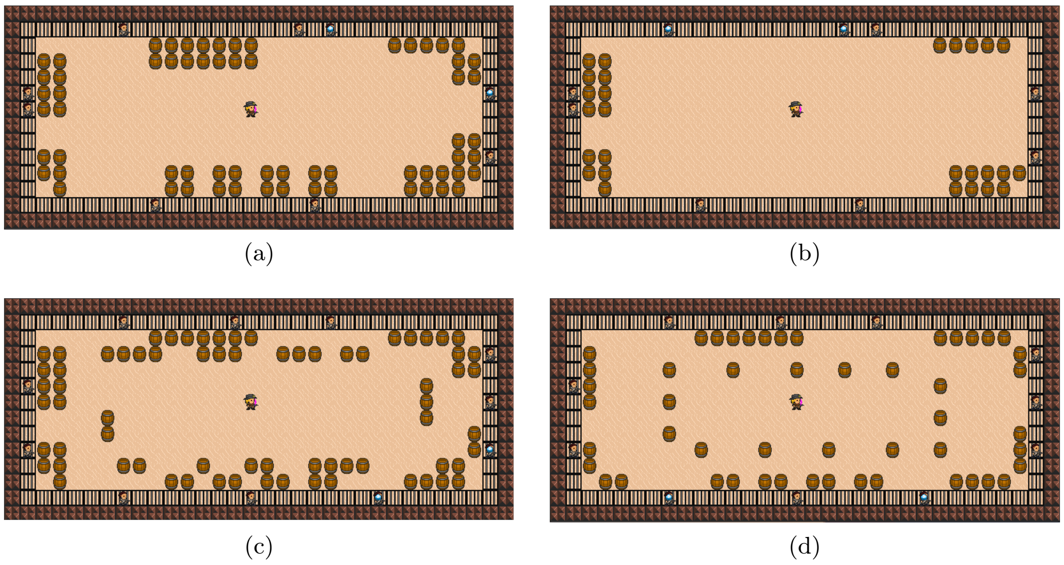

(a) (b) (c) (d)

Figure 7.43: New levels used for the *team* portability experiments in *Sheriff*.

Portability results All the resulting average of stats of the play-through of the agents, classified into level and agent, are included in Appendix E.4. Table 7.10 presents a summary of those results. A graph is included for each of the features relevant to the agents and their selection: *curiosity* (Fig 7.44), *EoG* (Fig 7.45), *exploration* (Fig 7.46), *hits* (Fig 7.47), *interactions* (Fig 7.48), *kills* (Fig 7.49), and *wins*, represented as the total number of *victories* (Fig 7.50). It shows the results of the 100 play-throughs per level of the corresponding agent (or agents).

All the agents seem to display the characteristics they were selected for across the different levels. E1 and E4 are agents related to winning the game but in different ways. E1 is a *Survivor* that wins the game by reaching the end of the game while avoiding killing many enemies. Its behaviour is shown on the overall stats across levels: >= 98% win rate, less than half of the bandits kill on average, and EoG ticks close to the maximum (1,000). E4, on the contrary, achieves a low average of EoG ticks across levels (<= 418.7) while still achieving a high rate of wins (>= 97%). E2 and E3 are both *High killers* and achieve similar killing stats through the levels. However, they have different skills related to exploration: E2 is a *High explorer* and reaches high exploration overall levels (>= 97.89%), while E3's exploration is low (<= 17.81%). Similarly, stats for E5 and E6 follow a similar trend across the levels.

Sheriff					
Features	**7.2d**	**7.43a**	**7.43b**	**7.43c**	**7.43d**
Total enemies	8	9	9	11	11
Total barrels	55	65	27	71	51
E1: *Survivor and low killer*					
Victories	98/100	99/100	100/100	99/100	100/100
Kills	3.28 (1.17)	4.02 (1.39)	3.89 (1.49)	4.78 (1.73)	4.87 (1.64)
EoG	992.58 (67.67)	990.32 (96.80)	1000 (0.00)	991.82 (81.80)	1000 (0.00)
E2: *High killer and explorer*					
Exploration	97.89% (10.65)	98.97% (9.13)	99.01% (8.22)	98.20% (9.86)	99.55% (3.08)
Kills	6.96 (0.31)	7.95 (0.50)	7.96 (0.40)	9.91 (0.71)	10.00 (0.0)
E3: *High killer and low explorer*					
Exploration	15.33% (4.06)	16.36% (5.15)	15.26% (6.03)	17.81% (5.46)	15.96% (4.72)
Kills	6.95 (0.50)	7.93 (0.70)	8.00 (0.00)	9.95 (0.50)	10.00 (0.00)
E4: *Speed-runner*					
Victories	100/100	97/100	98/100	99/100	100/100
EoG	364.75 (134.56)	360.53 (127.35)	363.59 (162.99)	417.71 (141.44)	418.7 (154.96)
E5: *Barrels shooter*					
Interactions	439.42 (343.26)	372.63 (311.24)	344.74 (305.19)	385.54 (314.72)	310.89 (275.01)
Kills	5.83 (1.86)	5.89 (2.47)	6.22 (2.36)	7.46 (3.30)	6.65 (3.22)
Hits	35.96 (10.31)	38.19 (13.63)	17.49 (7.81)	39.86 (12.58)	29.28 (9.06)
E6: *High curiosity and low interactions*					
Curiosity	95.77 (22.26)	101.81 (25.46)	73.46 (18.53)	99.92 (26.57)	89.47 (18.41)
Interactions	122.52 (66.51)	127.46 (68.56)	106.28 (64.33)	125.93 (67.61)	116.98 (64.28)

Table 7.10: Portability results for *Sheriff*. *Victories* are the total number of wins in the 100 play-throughs. The other dimensions are the average of the resulting gameplay stats and the corresponding *Std. Deviation*. The stats shown per agent are the ones related to their proficiency. EoG is 1000.

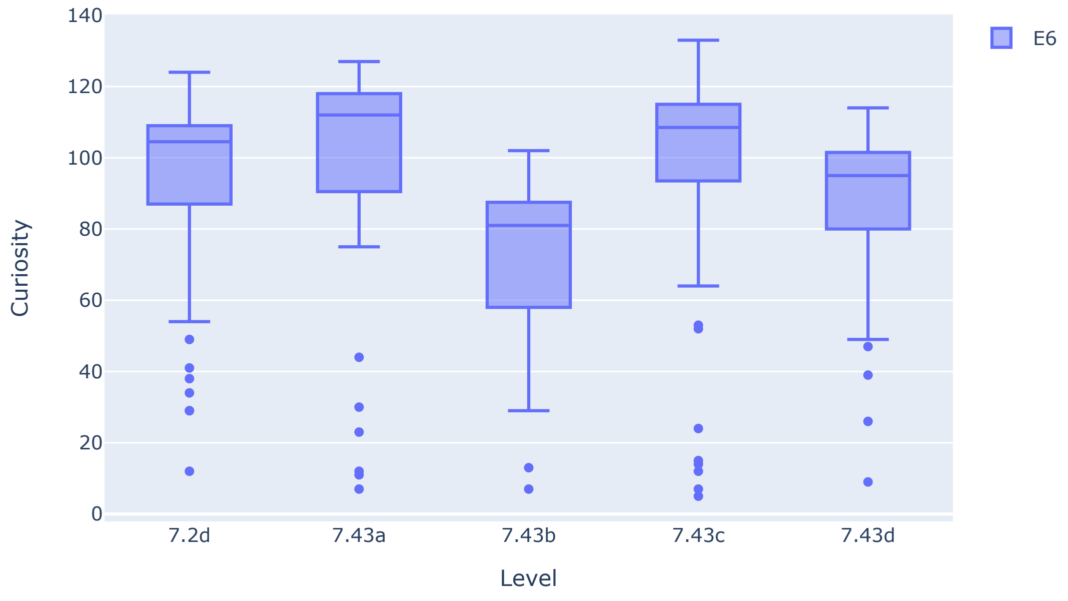


Figure 7.44: *Sheriff*: Resulting *curiosity* per level, from 100 gameplays. E6: *High curiosity and low interactions.*

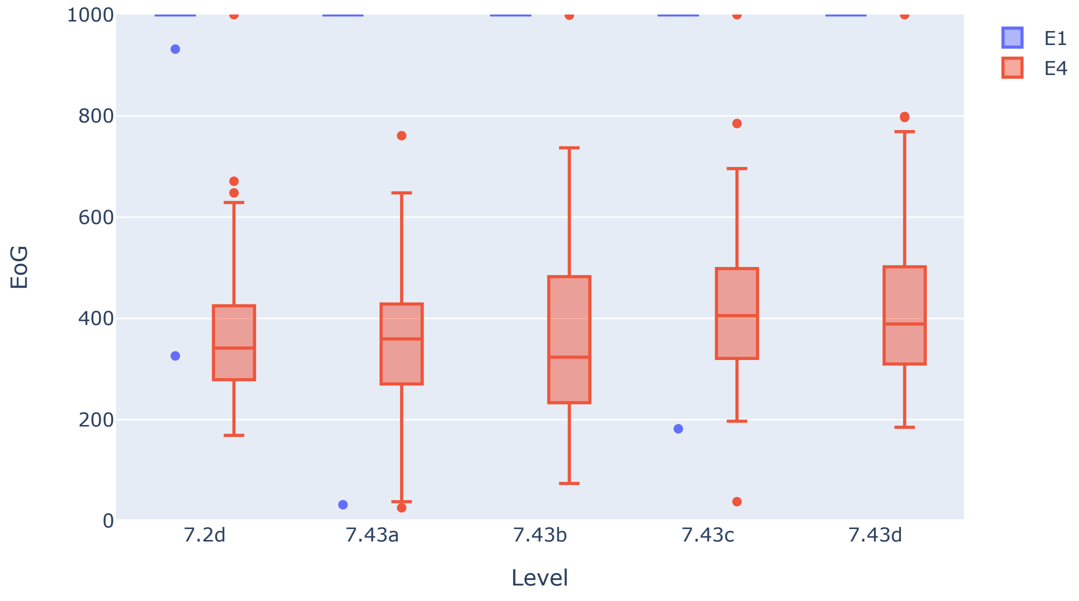


Figure 7.45: *Sheriff*: Resulting *EoG* per level, from 100 gameplays. E1: *Survivor and low killer*, E4: *Speed-runner*.

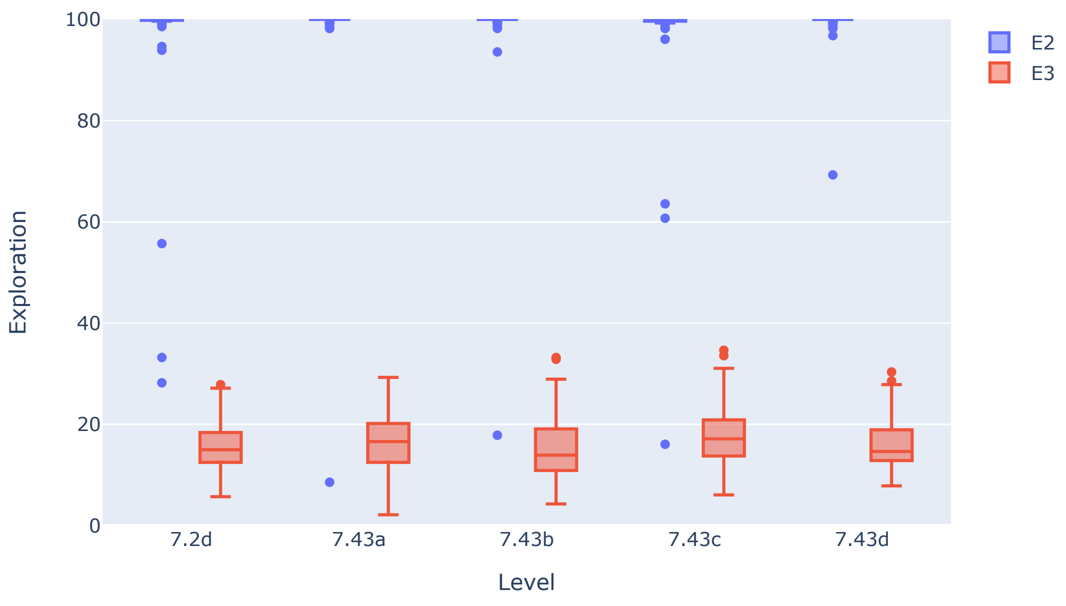


Figure 7.46: *Sheriff*: Resulting *exploration* per level, from 100 gameplays. E2: *High killer and explorer*, E3: *High killer and low explorer*.

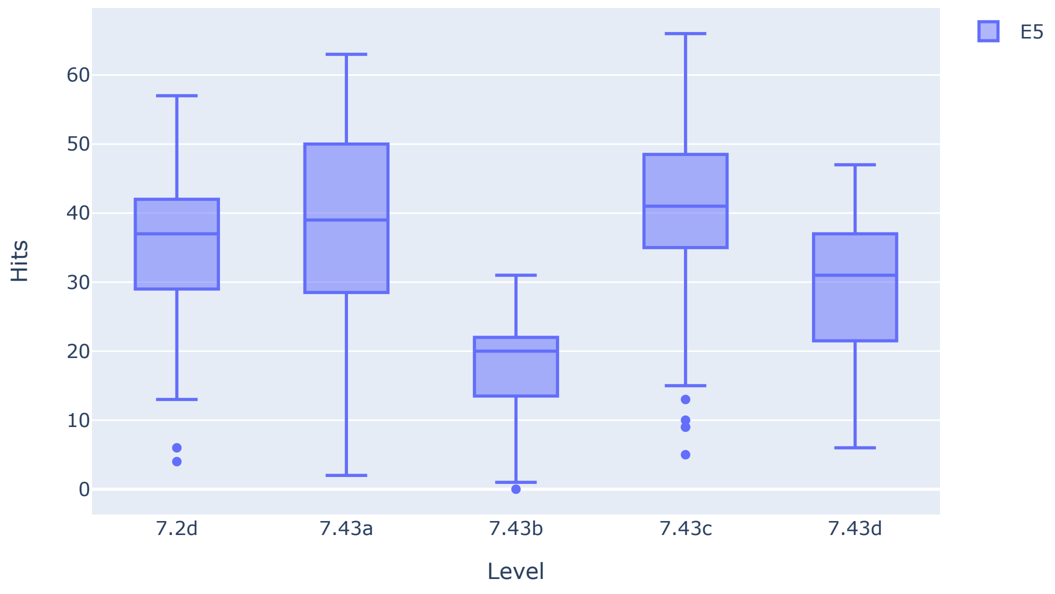


Figure 7.47: *Sheriff*: Resulting *hits* per level, from 100 gameplays. E5: *Barrels shooter*.

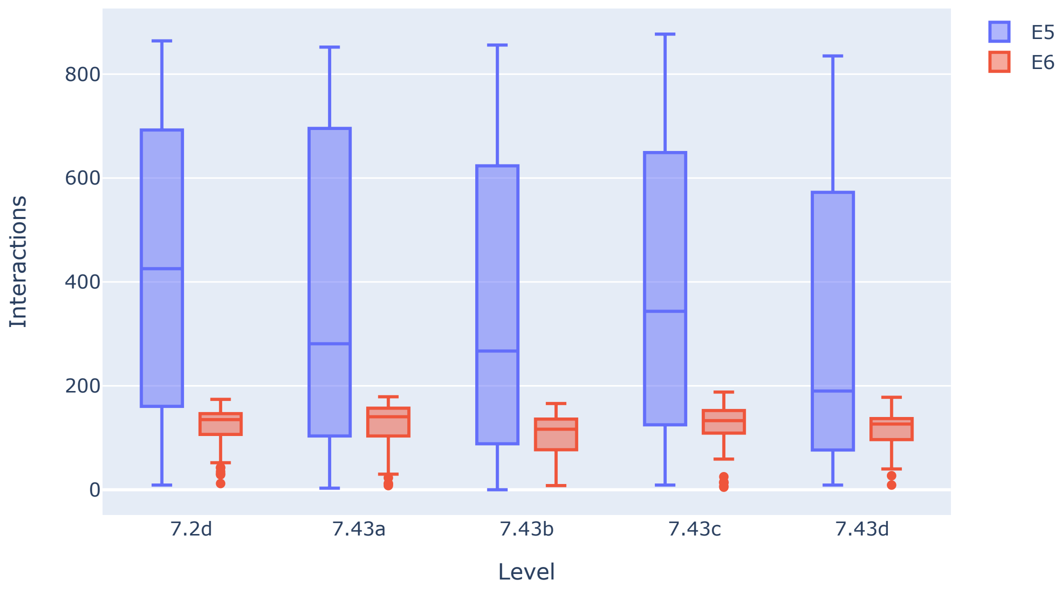


Figure 7.48: *Sheriff*: Resulting *interactions* per level, from 100 gameplays. E5: *Barrels shooter*, E6: *High curiosity and low interactions*.

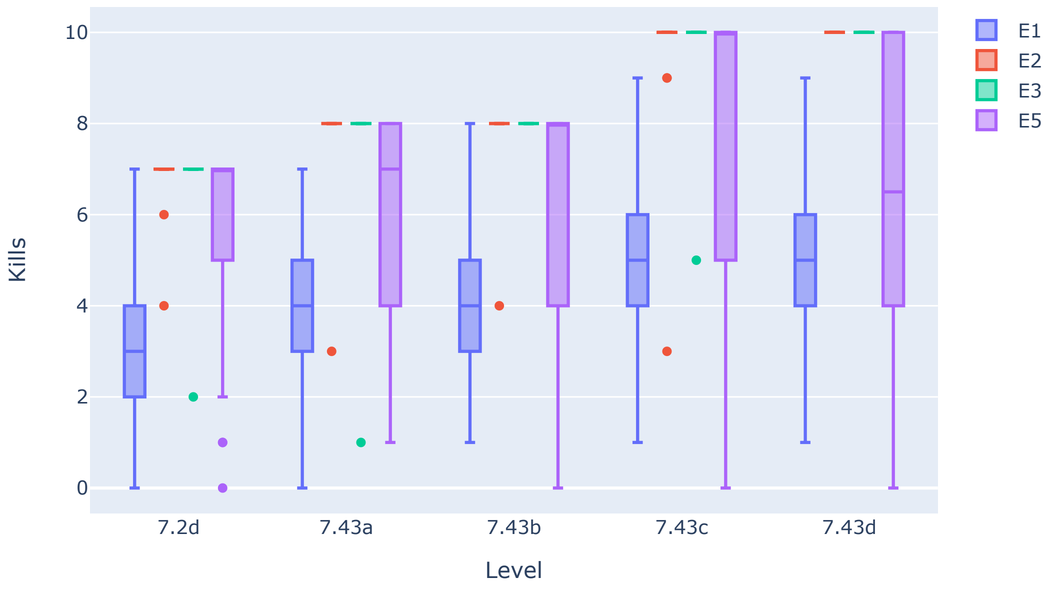


Figure 7.49: *Sheriff*: Resulting *kills* per level, from 100 gameplays. E1: *Survivor and low killer*, E2: *High killer and explorer*, E3: *High killer and low explorer*, E5: *Barrels shooter*.

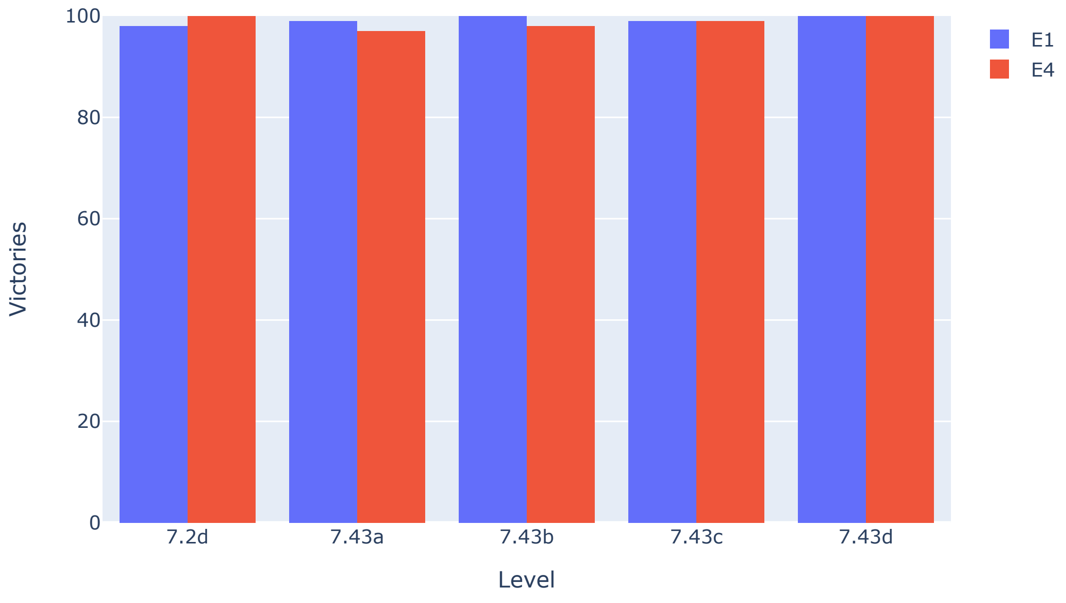


Figure 7.50: *Sheriff*: Resulting number of *victories* per level, from 100 gameplays. E1: *Survivor and low killer*, E4: *Speed-runner*.

7.3.5 Results overview

The generality of the heuristics allows running the *behaviour-type* agents at any level of the game. However, whether the proficiency identified in each of those agents is portable to new levels was an open question that I was looking to answer with this experiment.

The results show that the tendency of the resulting gameplay stats related to their *behaviour-type* is generally maintained between levels. In most cases, it is straightforward to confirm this portability by just looking at the results of the agent. However, there are cases where it is also necessary to consider the global stats between agents at a particular level to reach this conclusion. It is also possible that the characteristics of a game impede a particular *behaviour-type* agent from being portable to new levels. For example, the score in *Butterflies* is very dependent on the distribution of the elements at the start of the game, so the agents related to it, E1 and E2, are not capable of porting their proficiency when they play new ones. Therefore, a limitation of the approach appears when the feature relevant to the agent is very tight to the rules of the game. To mitigate the problem, I recommend running similar portability experiments after identifying the agents to ensure they can generalise to new levels before they are employed for testing.

These results support the theory that virtually all the agents generated are portable between levels. Therefore, they could be used to test new or updated levels to help find potential design flaws. These design flaws could be detected by highlighting those cases where the stats obtained by the agents do not fit the expectations. In the next section, I introduce preliminary work that experiments with this idea.

7.4 Exploratory Work: Using the *Team* for Level Testing

This section showcases work towards using the *team* of *behaviour-type* agents for testing the design and validity of new levels. I modify the original levels of each game used to generate the *team* to 1) be impossible to win, 2) remove a crucial element, or 3) increase its difficulty. Similarly to the MAP-Elites application and the portability experiments described in the previous section, each of the agents identified plays the level 100 times to gather the resulting stats. In each experiment, I describe the characteristics of the 'broken' level used and include a table with the resulting stats from each agent, highlighting those related to its *behaviour-type*.

Similarly to the portability experiments, the code can be found in a secondary branch of the Github repository [Guerrero Romero, 2021a] and the config files, executables, and results can be found in an OSF repository [Guerrero Romero, 2021c]. The scripts used to generate the graphs can be found in Github [Guerrero Romero, 2021b].

7.4.1 Testing a ‘broken‘ level in *Butterflies*

The structure of the new level (Fig. 7.51) is similar to the original one, but all the butterflies are cocoons instead. An isolated butterfly is included so the game does not end immediately. This butterfly does not have access to the area where the player is. Therefore, the player cannot win as they are unable to catch any butterflies, and cacoons cannot be transformed. As a result, the time always runs out.

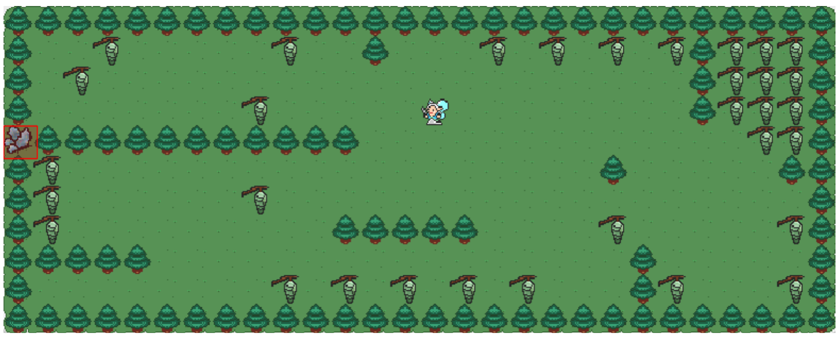


Figure 7.51: New ‘broken‘ level of *Butterflies* used for the exploratory work.

Table 7.11 presents the resulting stats for each *behaviour-type* agent from playing the new level, and Table 7.12 compares the results of the pertinent agents to the ones obtained in the previous levels. I include a graph for each of the features relevant to the agents and their selection: *collisions* (Fig. 7.52), *curiosity* (Fig. 7.53), *EoG* (Fig. 7.54), *exploration* (Fig. 7.55), *score* (Fig. 7.56), and *wins*, represented as the total number of *victories* (Fig. 7.57). It displays the results of the 100 play-throughs of the corresponding agents in the ‘broken‘ level compared to the portability experiment.

The resulting score, victories, and EoG ticks; related to agents E1, E2, and E4 raise a suspicion that something in the level does not work as expected. The average of the score for both high and low *Scorers* (E1 and E2, respectively) is 0, and although I have previously discerned that these agents may not be portable between levels, these results are intriguing. Furthermore, in this level, the *Speed-runner* (E4) never wins the game and averages an EoG ticks similar to the maximum allowed (2,000). Looking at these results, I can speculate that the level is not winnable and that the butterflies are not at a reachable distance to the player, as the score does not increase. These assumptions match with and issues introduced in the level.

On the contrary, the *High curiosity* (E3), *High explorer* (E5), and *Walls interaction* (E6) agents still achieve the expected results and the final average does not raise any areas of concern. The modifications in the level do not impede fulfilling their tasks. However, it is worth mentioning that looking at the graphs for collisions, curiosity, and exploration, the data from the 100 play-throughs seems to be less variable than in the previous levels.

***Butterflies* ‘broken‘ level**						
	E1	**E2**	**E3**	**E4**	**E5**	**E6**
EoG	2000 (0.00)	2000 (0.00)	2000 (0.00)	2000 (0.00)	2000 (0.00)	2000 (0.00)
Victories	0/100	0/100	0/100	0/100	0/100	0/100
Score	0.00 (0.00)	0.00 (0.00)	0.00 (0.00)	0.00 (0.00)	0.00 (0.00)	0.00 (0.00)
Exploration	100.00% (0.00)	100.00% (0.00)	59.49% (5.54)	100.00% (0.00)	100.00% (0.00)	59.40% (5.76)
Curiosity	25.30 (3.22)	111.00 (0.00)	103.74 (9.26)	111.00 (0.00)	111.00 (0.00)	103.72 (9.08)
Collisions	37.65 (4.24)	570.15 (48.72)	1858.56 (17.68)	183.83 (9.98)	810.07 (73.53)	1858.29 (17.11)

Table 7.11: Experimental level testing results overview. *Victories* are the total number of wins in the 100 play-throughs. The other dimensions are the average of the resulting gameplay stats and the corresponding *Std. Deviation.* Shaded cells are related to the *behaviour-type* of the agent. Those highlighted are directly or indirectly affected by the changes in the level. E1: *Low scorer*; E2: *High scorer*; E3: *High curiosity*; E4: *Speed-runner*; E5: *High explorer*; E6: *Walls interaction.* EoG: 2000; Max Score: 64.

Butterflies						
	7.2a	**7.19a**	**7.19b**	**7.19c**	**7.19d**	**‘Broken‘**
Max. score	64	38	58	38	18	64
Total walls	102	93	91	83	99	102
E1: *Low scorer*						
Score	20.44 (8.70)	29.58 (1.99)	43.56 (10.54)	21.2 (1.42)	12.64 (4.12)	**0.00 (0.00)**
E2: *High scorer*						
Score	47.76 (16.60)	30.78 (1.90)	51.1 (6.21)	22.58 (1.24)	11.86 (3.99)	**0.00 (0.00)**
E3: *High curiosity*						
Curiosity	116.53 (25.03)	98.99 (8.81)	116.38 (18.25)	82.13 (7.66)	76.38 (30.97)	103.74 (9.26)
Collisions	1436.82 (500.53)	1754.85 (160.49)	1362.33 (603.94)	1780.46 (133.97)	715.56 (708.63)	1858.56 (17.68)
E4: *Speed-runner*						
Victories	100/100	100/100	99/100	100/100	89/100	**0/100**
EoG	102.15 (47.83)	103.49 (34.69)	114.38 (42.03)	73.61 (35.66)	61.13 (30.34)	**2000.00 (0.00)**
E5: *High explorer*						
Exploration	99.29% (4.78)	99.30% (5.18)	99.33% (5.77)	99.77% (1.87)	74.56% (27.15)	100.00% (0.00)
E6: *Walls interaction*						
Exploration	67.43% (16.16)	60.28% (8.26)	68.15% (10.06)	44.60% (5.74)	43.25% (15.00)	59.40% (5.76)
Curiosity	114.59 (27.54)	96.53 (11.28)	120.46 (17.41)	79.86 (9.37)	75.16 (28.25)	103.72 (9.08)

Table 7.12: *Butterflies*: Portability and experimental level testing results comparison. *Victories* are the total number of wins in the 100 play-throughs. The other dimensions are the average of the resulting gameplay stats and the corresponding *Std. Deviation.* The stats shown per agent are the ones related to their proficiency. The dimensions highlighted are the ones directly or indirectly impacted by the changes in the ‘broken‘ level. EoG is 2000.

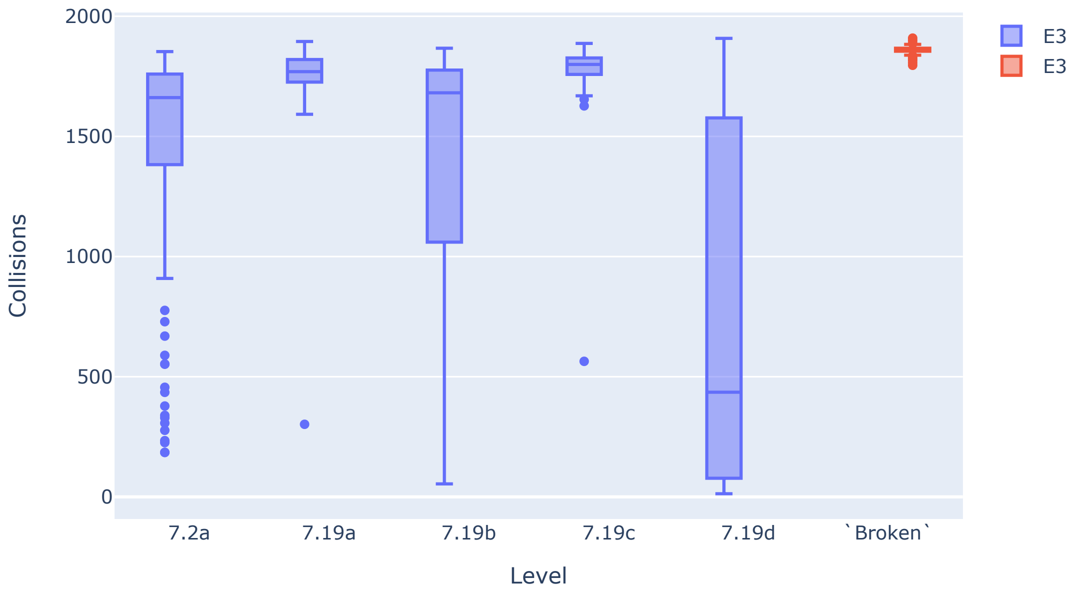


Figure 7.52: Experimental level testing in *Butterflies*: Resulting *collisions* in the 'broken' level compared to the portability results, from 100 gameplays. E3: *High curiosity*.

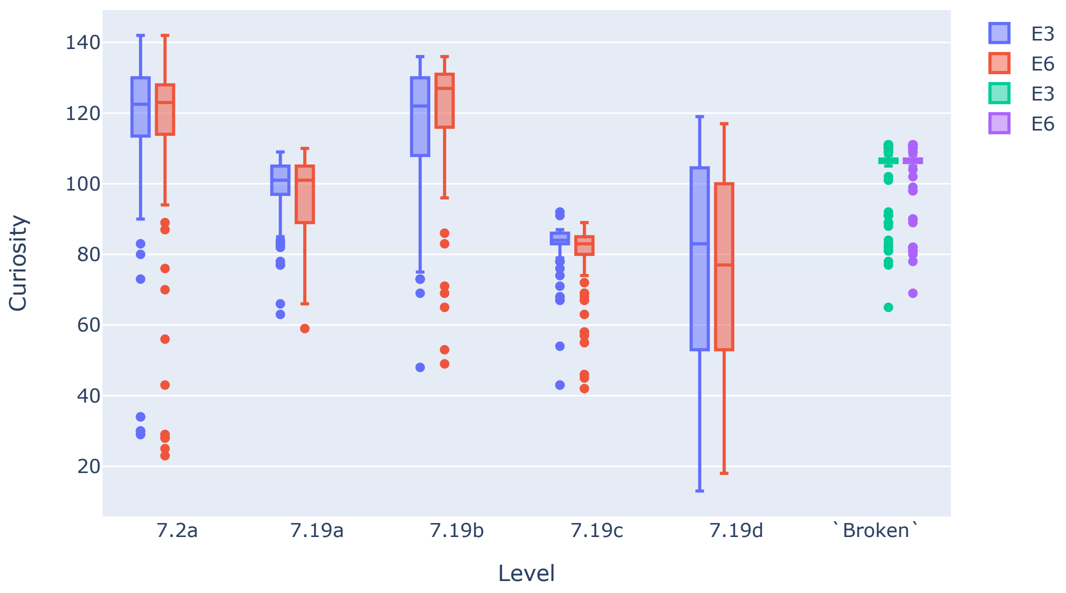


Figure 7.53: Experimental level testing in *Butterflies*: Resulting *curiosity* in the 'broken' level compared to the portability results, from 100 gameplays. E3: *High curiosity*, E6: *Walls interaction*.

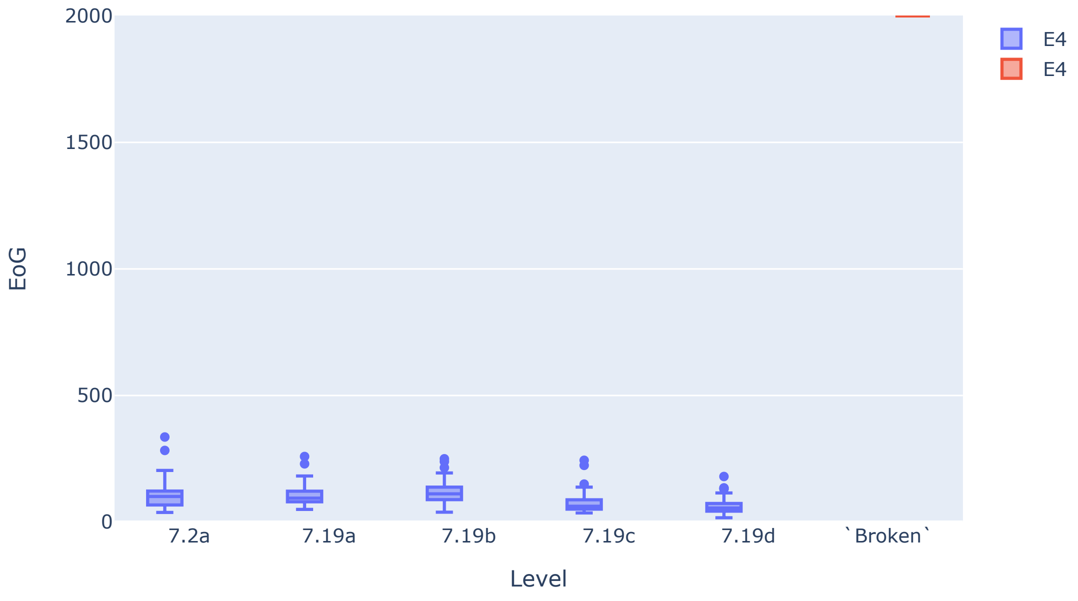


Figure 7.54: Experimental level testing in *Butterflies*: Resulting *EoG* in the 'broken' level compared to the portability results, from 100 gameplays. E4: *Speed-runner*.

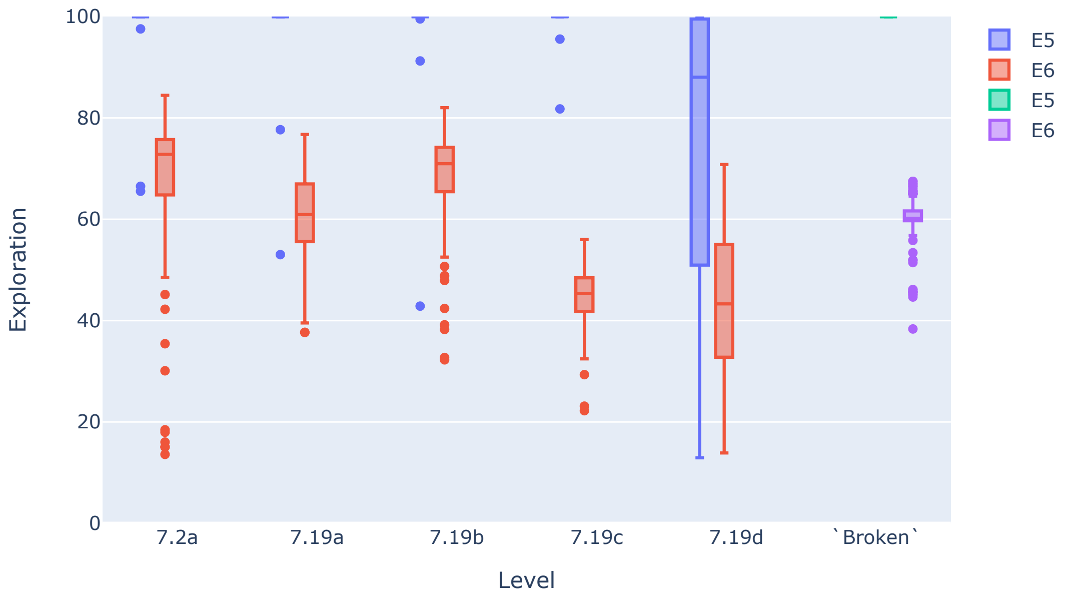


Figure 7.55: Experimental level testing in *Butterflies*: Resulting *exploration* in the 'broken' level compared to the portability results, from 100 gameplays. E5: *High explorer*, E6: *Walls interaction* (with mid-range exploration).

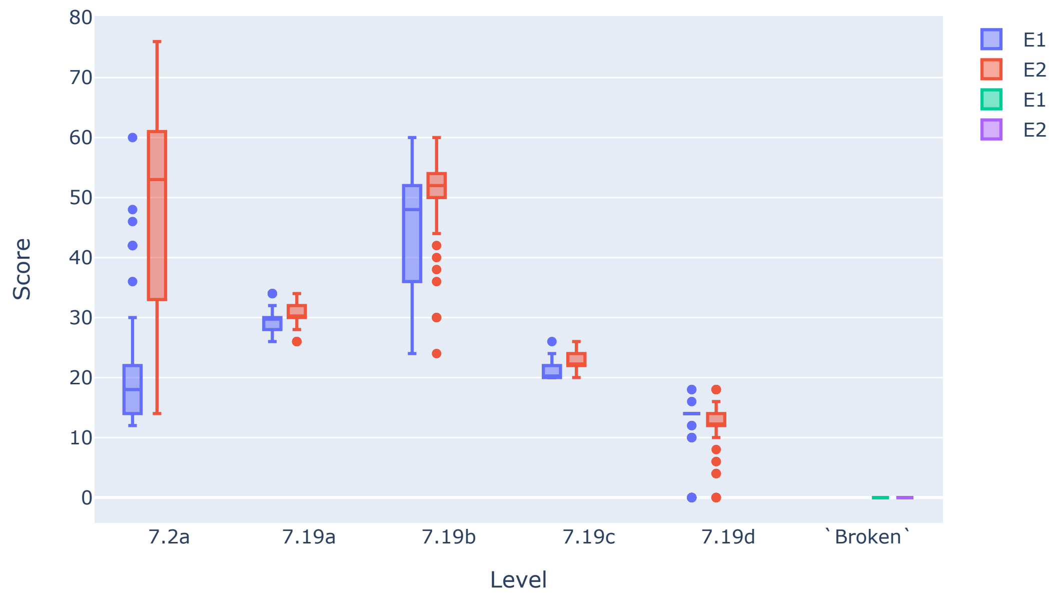


Figure 7.56: Experimental level testing in *Butterflies*: Resulting *score* in the ‘broken‘ level compared to the portability results, from 100 gameplays. E1: *Low scorer*, E2: *High scorer*.

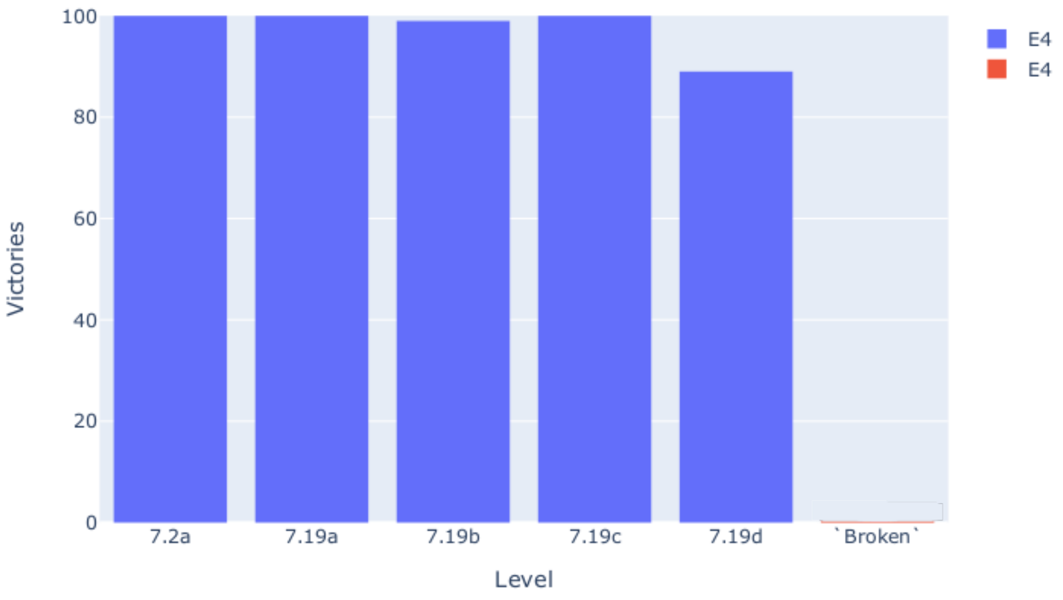


Figure 7.57: Experimental level testing in *Butterflies*: Resulting total number of *victories* in the ‘broken‘ level compared to the portability results, from 100 gameplays. E4: *Speedrunner*.

7.4.2 Testing a ‘broken‘ level in *Zelda*

The new level has the same number of monsters in the same locations as the original level. However, an area of the map is inaccessible and contains one of the monsters and the key. Therefore, this level has the following issues: not all locations are reachable, and the player cannot collect the key or win.

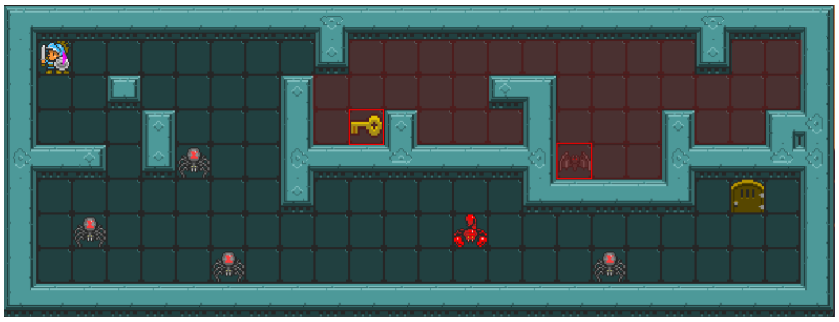


Figure 7.58: New ‘broken‘ level of *Zelda* used for the exploratory work.

Table 7.13 shows the resulting stats for each *behaviour-type* agent from playing this new level, and Table 7.14 compares the results of the pertinent agents to the ones obtained in the previous levels. I include a graph for each of the features relevant to the agents and their selection: *EoG* (Fig. 7.59), *exploration* (Fig. 7.60), *kills* (Fig. 7.61), *score* (Fig. 7.62), and *wins*, represented as the total number of *victories* (Fig. 7.63). It displays the results of the 100 play-throughs of the corresponding agents in the ‘broken‘ level compared to the portability experiment.

The agents whose gameplay results suffer significant changes are E2, E3, and E5. The win rate for the *Speed-runner* (E2) drops to 0%, while the average of EoG ticks increases to 1897.65. This value is higher than any result previously obtained by this agent in the portability experiments (<= 1242.26). In addition, the exploration for the *High explorer* agents (E3 and E5) drops, from an average higher than 90% in the previous levels, to lower than 70% in the ‘broken‘ one. These results hint that part of the level is not accessible. Coincidentally, all these agents relate to either winning or achieving a high exploration, so they are directly impacted by the issues introduced to the level. There is also a difference in the results when the agents are indirectly affected. E4 (*Low killer*) achieves a higher average of kills than in the previous cases (4.06). The number of enemies does not increase, but not being able to finish the game provides more time for this agent to kill enemies, indirectly impacting its resulting stats. Similarly, the final score of E1 drops to 9.72 because it is not possible to get the key, win, or kill the enemy that is out of reach.

Stats related to E6 (*High killer and low explorer*) does not seem to be impacted.

***Zelda* 'broken' level**						
	E1	**E2**	**E3**	**E4**	**E5**	**E6**
EoG	1932.56 (338.70)	1897.65 (386.29)	1972.76 (200.47)	1884.8 (407.23)	1861.88 (462.20)	1983.32 (166.80)
Victories	0/100	0/100	0/100	0/100	0/100	0/100
Score	9.72 (1.46)	8.21 (2.54)	9.56 (1.05)	8.04 (2.48)	9.22 (2.00)	9.01 (1.50)
Exploration	67.61% (6.47)	66.89% (8.57)	68.25% (3.38)	67.23% (7.16)	66.47% (9.18)	44.28% (9.16)
Kills	4.88 (0.64)	4.16 (1.15)	4.76 (0.57)	4.06 (1.17)	4.63 (0.92)	4.48 (0.74)

Table 7.13: Experimental level testing results overview. *Victories* are the total number of wins in the 100 play-throughs. The other dimensions are the average of the resulting gameplay stats and the corresponding *Std. Deviation*. Shaded cells are related to the *behaviour-type* of the agent. Those highlighted are directly or indirectly affected by the changes in the level. E1: *High scorer*; E2: *Speed-runner*; E3: *High explorer*; E4: *Low killer*; E5: *High killer and explorer*; E6: *High killer and low explorer*. EoG: 2000; Max. Score: 14; Number of enemies: 6.

Zelda						
	7.2b	**7.28a**	**7.28b**	**7.28c**	**7.28d**	**'Broken'**
Max. score	14	14	14	18	14	14
Total enemies	6	6	6	8	6	6
E1: *High scorer*						
Score	12.25 (2.64)	12.11 (2.71)	11.7 (3.21)	16.44 (2.73)	11.55 (3.40)	**9.72 (1.46)**
E2: *Speed-runner*						
Victories	90/100	76/100	65/100	89/100	96/100	**0/100**
EoG	617.18 (487.03)	1187.98 (566.51)	1242.26 (652.64)	689.03 (526.05)	450.86 (331.35)	**1897.65 (386.29)**
E3: *High explorer*						
Exploration	98.03% (7.80)	95.36% (15.00)	95.25% (14.70)	98.31% (6.93)	93.33% (17.73)	**68.25% (3.38)**
E4: *Low killer*						
Kills	1.62 (1.37)	3.25 (1.44)	2.97 (1.51)	3.53 (2.03)	1.79 (1.34)	**4.06 (1.17)**
E5: *High killer and explorer*						
Kills	5.68 (0.98)	5.66 (0.85)	5.44 (1.18)	7.71 (0.75)	5.41 (1.23)	4.63 (0.92)
Exploration	97.19% (10.81)	97.93% (7.04)	94.71% (15.07)	98.61% (5.93)	94.00% (15.27)	**66.47% (9.18)**
E6: *High killer and low explorer*						
Kills	4.78 (1.04)	5.17 (0.80)	4.21 (1.19)	7.26 (1.30)	3.77 (1.33)	4.48 (0.74)
Exploration	48.38% (11.54)	50.66% (13.73)	42.25% (15.96)	54.79% (15.34)	35.73% (14.71)	44.28% (9.16)

Table 7.14: *Zelda*: Portability and experimental level testing results comparison. *Victories* are the total number of wins in the 100 play-throughs. The other dimensions are the average of the resulting gameplay stats and the corresponding *Std. Deviation*. The stats shown per agent are the ones related to their proficiency. The dimensions highlighted are the ones directly or indirectly impacted by the changes in the 'broken' level. EoG is 2000.

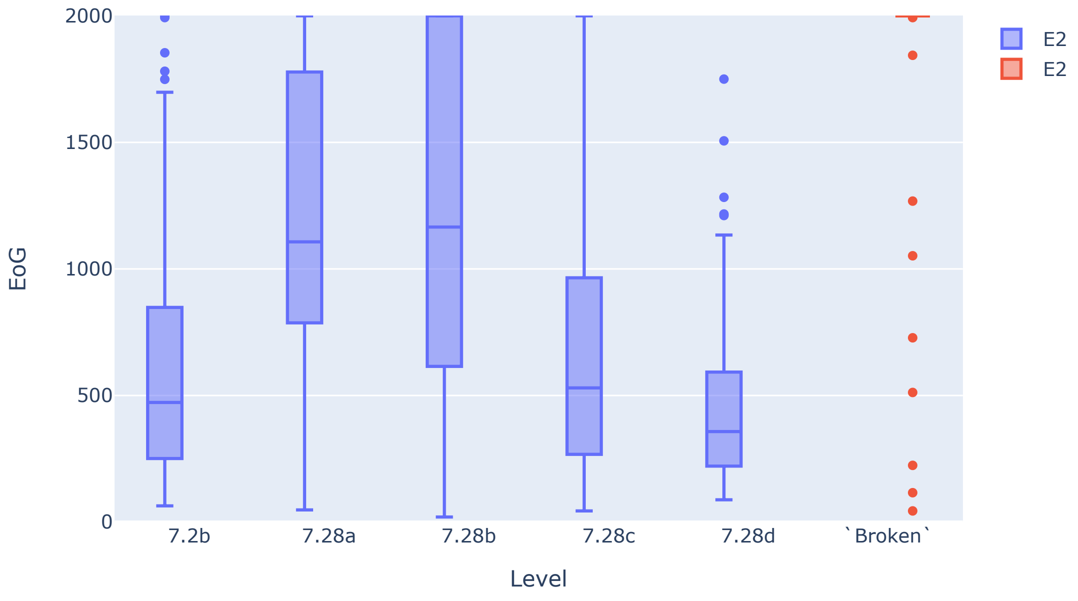


Figure 7.59: Experimental level testing in *Zelda*: Resulting *EoG* in the 'broken' level compared to the portability results, from 100 gameplays. E2: *Speed-runner*.

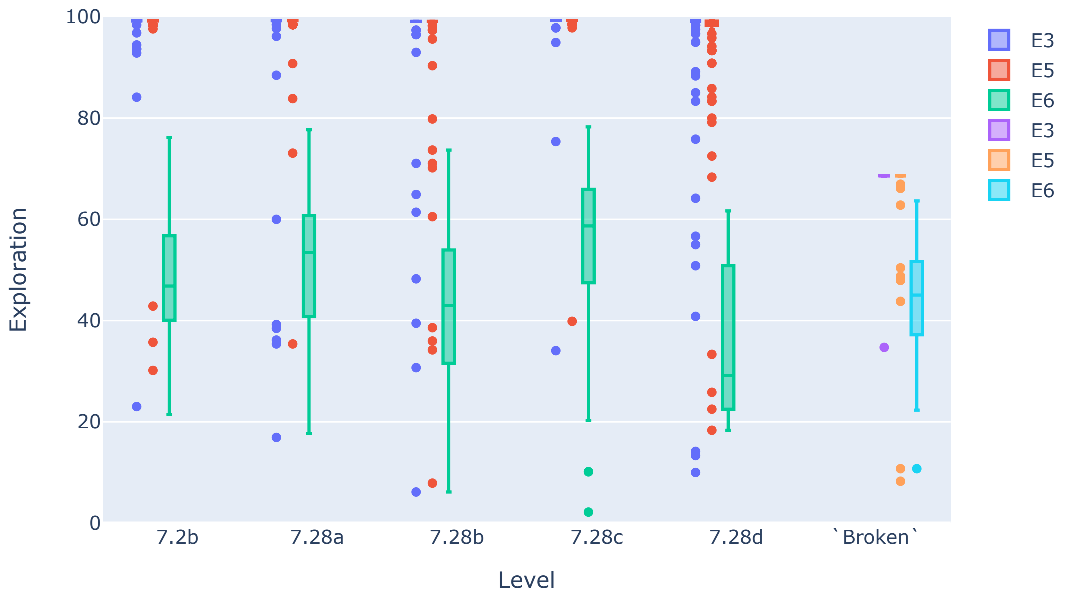


Figure 7.60: Experimental level testing in *Zelda*: Resulting *exploration* in the 'broken' level compared to the portability results, from 100 gameplays. E3: *High explorer*, E5: *High killer and explorer*, E6: *High killer and low explorer*.

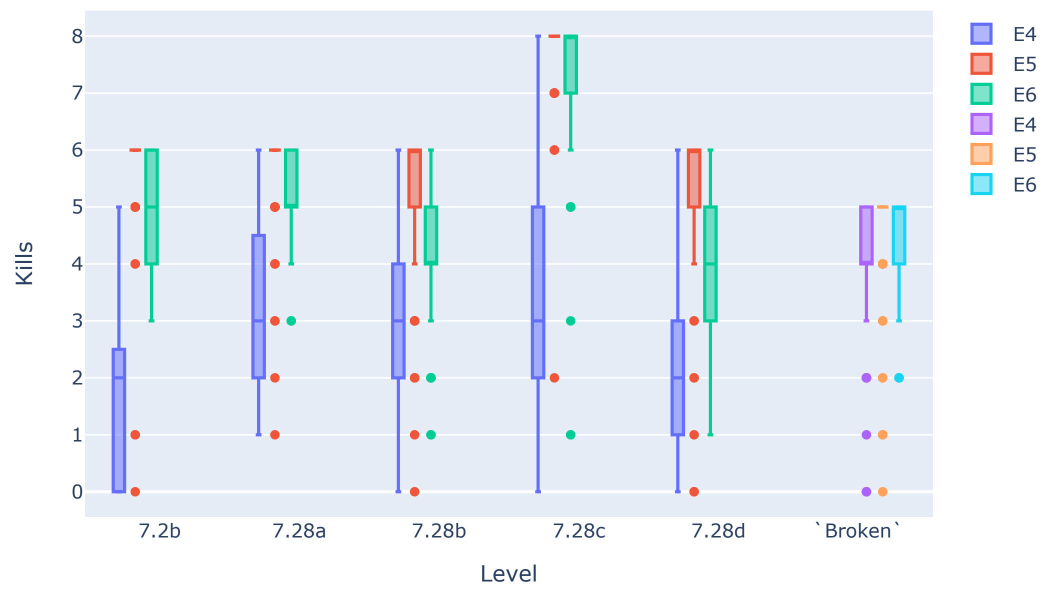


Figure 7.61: Experimental level testing in *Zelda*: Resulting *kills* in the ‘broken‘ level compared to the portability results, from 100 gameplays. E4: *Low killer*, E5: *High killer and explorer*, E6: *High killer and low explorer*.

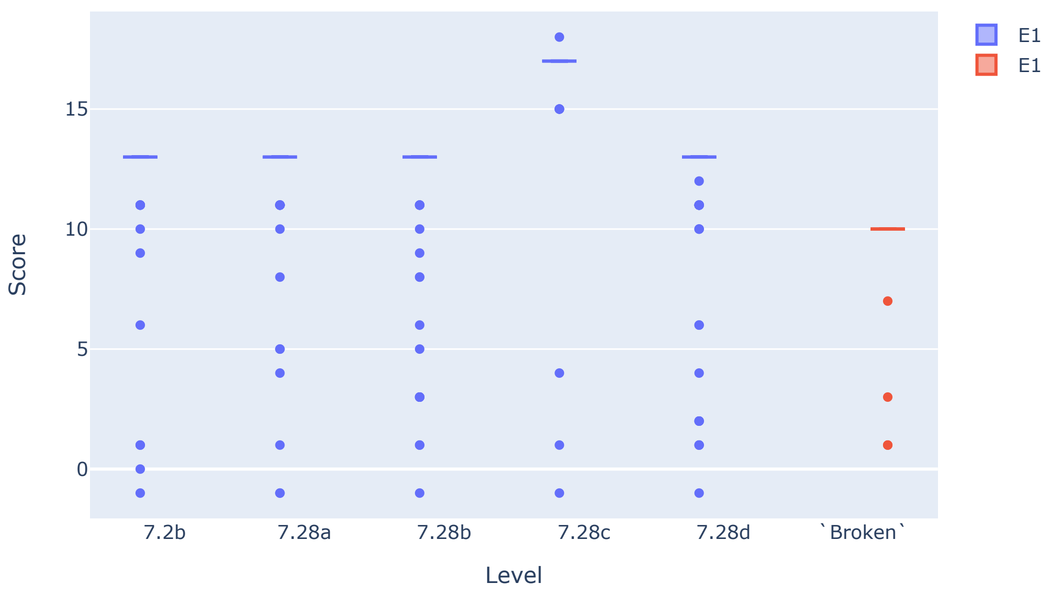


Figure 7.62: Experimental level testing in *Zelda*: Resulting *score* in the ‘broken‘ level compared to the portability results, from 100 gameplays. E1: *High scorer*.

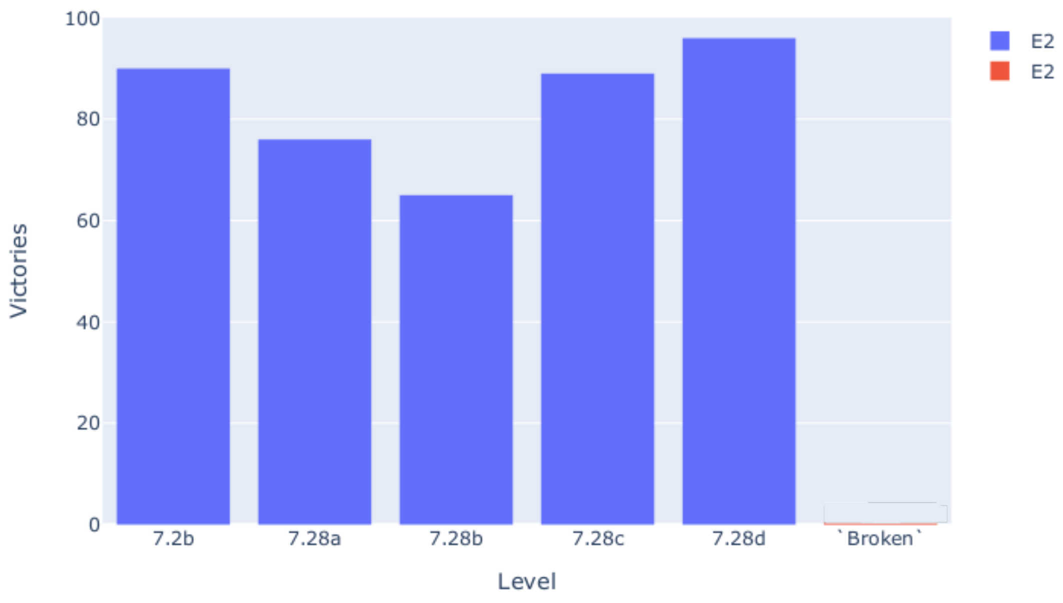


Figure 7.63: Experimental level testing in *Zelda*: Resulting total number of *victories* in the ‘broken‘ level compared to the portability results, from 100 gameplays. E2: *Speedrunner*.

7.4.3 Testing a ‘broken‘ level in *Digdug*

This level is similar to the original in terms of gems, gold, and monsters. The main difference is that all the breakable walls have been removed. These are a crucial element of the game as they can be used as guidance or protection from enemies. As a result, it is expected that removing them affects playability.

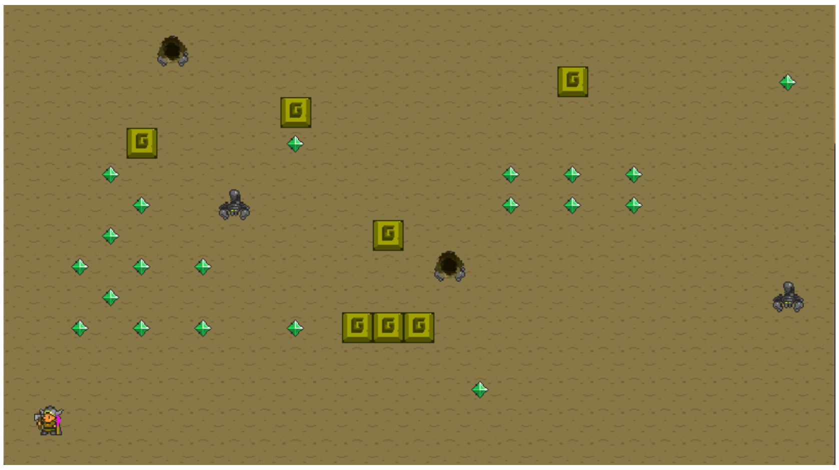


Figure 7.64: New ‘broken‘ level of *Digdug* used for the exploratory work.

Table 7.15 presents the resulting stats for each *behaviour-type* agent when playing this level, and Table 7.16 compares the results of the pertinent agents to the ones obtained in the previous levels. I include a graph for each of the features relevant to the agents and their selection: *curiosity* (Fig. 7.65), *exploration* (Fig. 7.66), *hits* (Fig. 7.67), *items* (Fig. 7.68), *kills* (Fig. 7.69), and *score* (Fig. 7.70). It displays the results of the 100 play-throughs of the corresponding agents in the ‘broken‘ level compared to the portability experiment.

The tweaked level has the walls removed, so the agent expected to break them (E4) is the one directly impacted by the change. This hypothesis is confirmed by looking at its results: the curiosity and hits drop drastically, from an average of 464.6 and 269.65, respectively, on the original level to 47.58 and 18.31 on the new one. The exploration of this agent also is impacted and drops to 45.53% because it is likely that this agent makes use of its ability to break walls to cover the map. On the other hand, the exploration for the *Low explorer* (E6) increases (slightly) to 44.32%, while in the rest of the levels it is between $30 - 37\%$. By analysing this agent on its own, this result could be taken as an outlier. However, when analysed in conjunction with the rest, I can assume that this change could also be linked to the nonexistence of walls. The agent is indirectly impacted because it does not use actions or game-ticks to break the walls, being able to move instead, covering more physical space in the level.

Neither the items, kills, or high exploration features hint that there may be an issue, as these stats maintain the expected trend.

***Digdug* ‘broken‘ level**						
	E1	**E2**	**E3**	**E4**	**E5**	**E6**
Score	40.21 (2.76)	25.30 (2.74)	31.95 (2.7)	41.78 (1.99)	32.41 (3.41)	40.26 (3.13)
Exploration	45.90% (5.18)	100.00% (0.00)	32.61% (6.03)	45.53% (6.06)	100.00% (0.00)	44.32% (5.42)
Kills	10.34 (1.31)	2.61 (1.34)	11.54 (0.74)	11.47 (0.67)	6.25 (1.64)	10.67 (1.48)
Items	26.01 (1.32)	27.00 (0.00)	10.33 (2.82)	25.53 (2.28)	27.00 (0.00)	25.08 (2.62)
Curiosity	42.83 (2.17)	36.57 (1.29)	24.19 (4.63)	47.58 (2.96)	41.32 (1.70)	40.67 (3.37)
Hits	17.07 (1.30)	9.61 (1.34)	14.03 (1.68)	18.31 (0.74)	13.25 (1.64)	17.03 (1.54)

Table 7.15: Experimental level testing results overview. The results are the average of the gameplay stats obtained in the 100 play-throughs. The values in parenthesis represent the corresponding *Std. Deviation*. Shaded cells are related to the *behaviour-type* of the agent. Those highlighted are directly or indirectly affected by the changes in the level. E1: *High collector and killer*; E2: *High collector and low killer*; E3: *Low collector and high killer*; E4: *Walls breaker*; E5: *High explorer*; E6: *Low explorer and high scorer.* Max. Score: 44; Number of enemies: 12; Number of items: 27.

Digdug						
	7.2c	**7.36a**	**7.36b**	**7.36c**	**7.36d**	**‘Broken‘**
Max. score	44	48	61	14	53	44
Total enemies	12	16	13	5	14	12
Total items	27	19	41	8	50	27
Breakable walls	267	292	247	196	103	0
E1: *High collector and killer*						
Kills	10.64 (0.73)	13.74 (1.50)	11.56 (0.86)	3.88 (0.35)	12.68 (0.69)	10.34 (1.31)
Items	26.92 (0.30)	18.91 (0.28)	40.76 (0.83)	7.95 (0.41)	49.90 (0.62)	26.01 (1.32)
E2: *High collector and low killer*						
Kills	2.81 (1.50)	3.19 (1.70)	3.05 (1.65)	0.60 (0.68)	4.69 (1.59)	2.61 (1.34)
Items	27.00 (0.00)	19.00 (0.00)	40.99 (0.10)	8.00 (0.00)	50.00 (0.00)	27.00 (0.00)
E3: *Low collector and high killer*						
Kills	9.64 (1.77)	12.62 (3.57)	12.53 (0.77)	4.7 (0.87)	13.31 (0.97)	11.54 (0.74)
Items	9.14 (3.10)	5.05 (3.08)	11.36 (3.48)	1.72 (1.08)	9.74 (8.02)	10.33 (2.82)
E4: *Walls breaker*						
Exploration	70.12% (11.21)	67.66% (13.90)	67.70% (12.66)	64.73% (06.65)	46.98% (17.62)	**45.53% (6.06)**
Curiosity	464.6 (68.09)	463.77 (87.06)	434.05 (67.64)	358.54 (28.05)	249.87 (77.28)	**47.58 (2.96)**
Hits	269.65 (38.97)	282.32 (51.86)	245.53 (37.10)	197.70 (14.77)	120.67 (36.05)	**18.31 (0.74)**
E5: *High explorer*						
Exploration	99.98% (0.12)	100% (0.00)	99.99% (0.07)	100% (0.00)	98.51% (10.47)	100.00% (0.00)
E6: *Low explorer and high scorer*						
Exploration	30.92% (5.29)	26.49% (5.13)	36.32% (4.12)	26.96% (5.96)	37.98% (5.83)	**44.32% (5.42)**
Score	35.49 (5.60)	35.46 (6.04)	51.3 (4.25)	11.02 (2.51)	47.71 (9.44)	40.26 (3.13)

Table 7.16: *Digdug*: Portability and experimental level testing results comparison. The results are the average of the gameplay stats obtained in the 100 play-throughs. The values in parenthesis represent the corresponding *Std. Deviation*. The stats shown per agent are the ones related to their proficiency. The dimensions highlighted are the ones directly or indirectly impacted by the changes in the ‘broken‘ level. EoG is 2000.

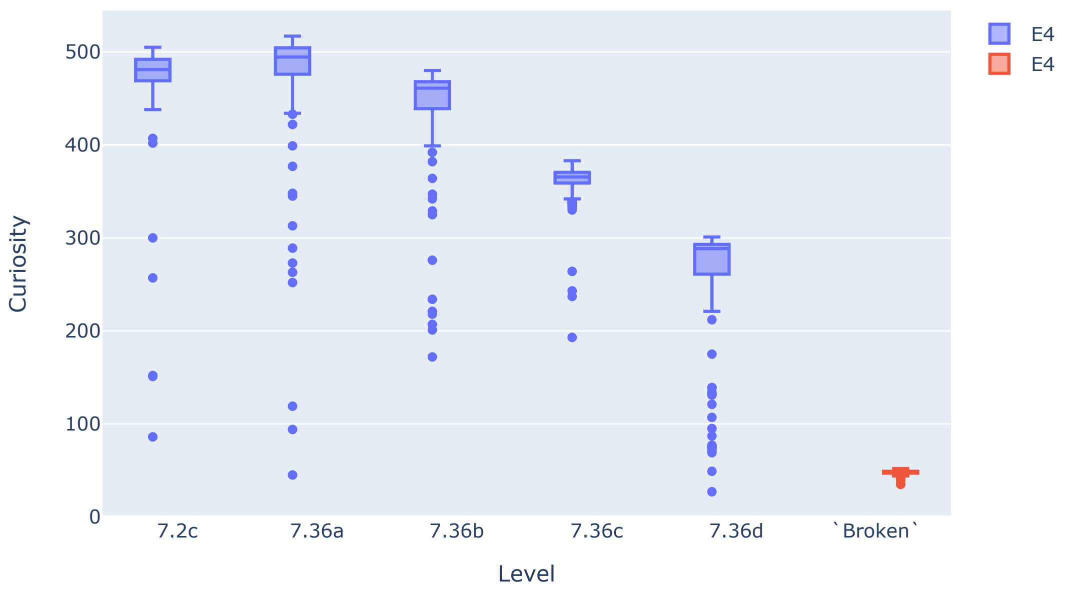


Figure 7.65: Experimental level testing in *Digdug*: Resulting *curiosity* in the 'broken' level compared to the portability results, from 100 gameplays. E4: *Walls breaker*.

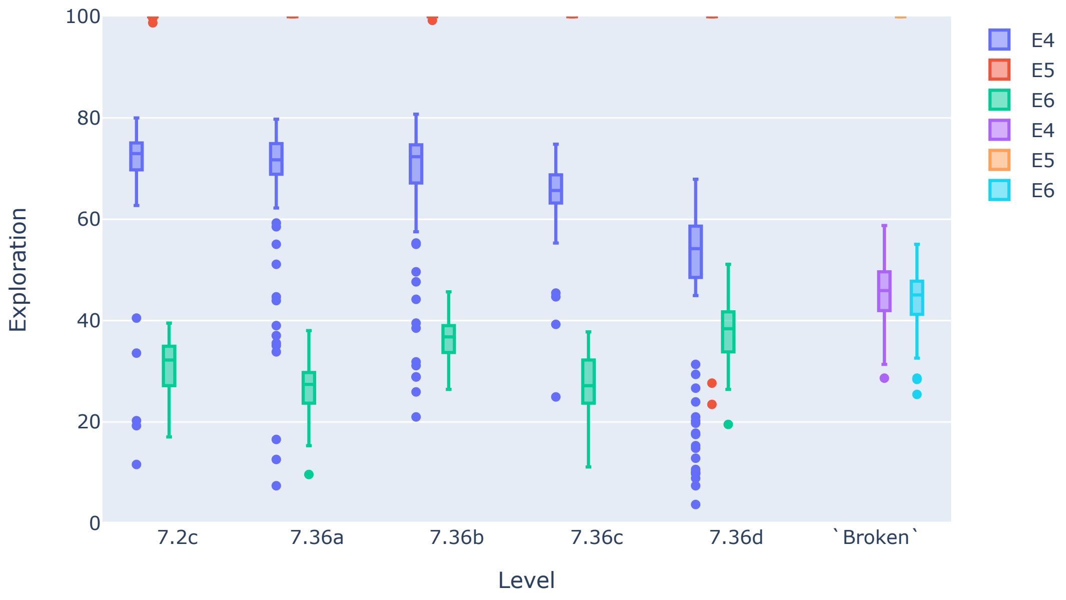


Figure 7.66: Experimental level testing in *Digdug*: Resulting *exploration* in the 'broken' level compared to the portability results, from 100 gameplays. E4: *Walls breaker*, E5: *High explorer*, E6: *Low explorer and high scorer*.

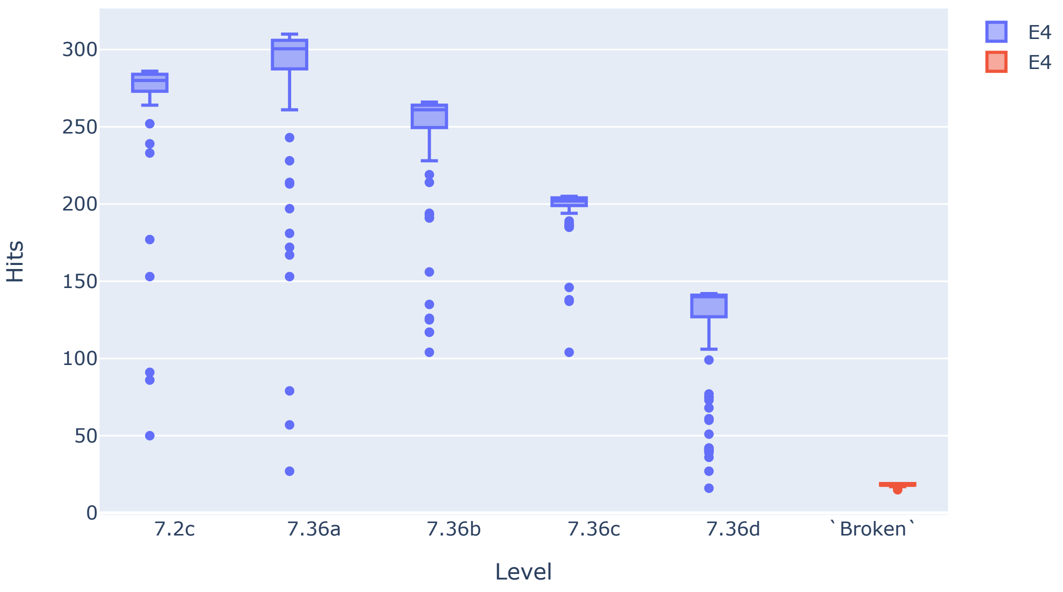


Figure 7.67: Experimental level testing in *Digdug*: Resulting *hits* in the ‘broken‘ level compared to the portability results, from 100 gameplays. E4: *Walls breaker.*

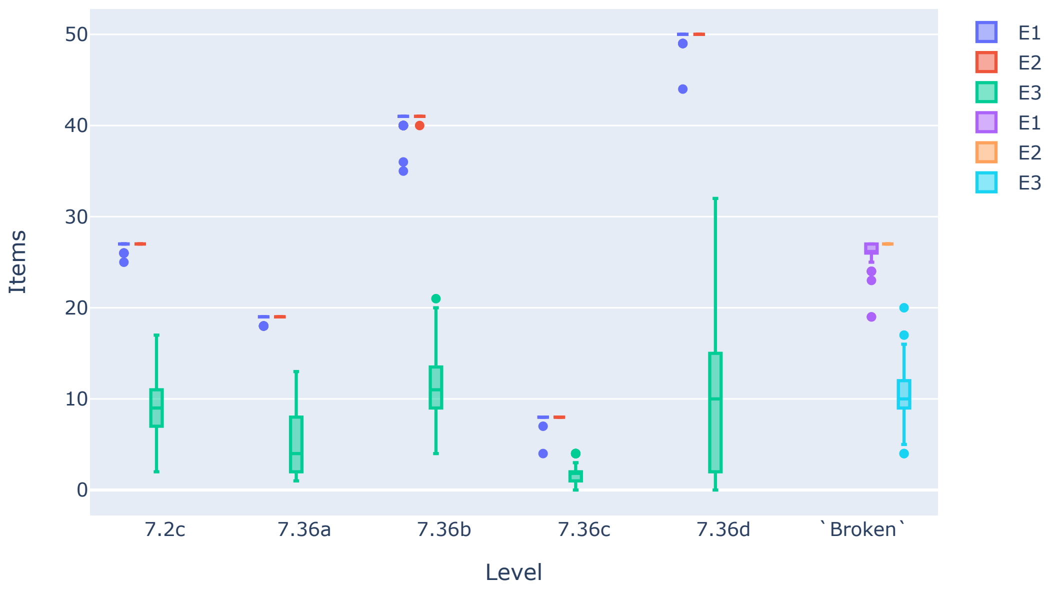


Figure 7.68: Experimental level testing in *Digdug*: Resulting *items* in the ‘broken‘ level compared to the portability results, from 100 gameplays. E1: *High collector and killer*, E2: *High collector and low killer*, E3: *Low collector and high killer.*

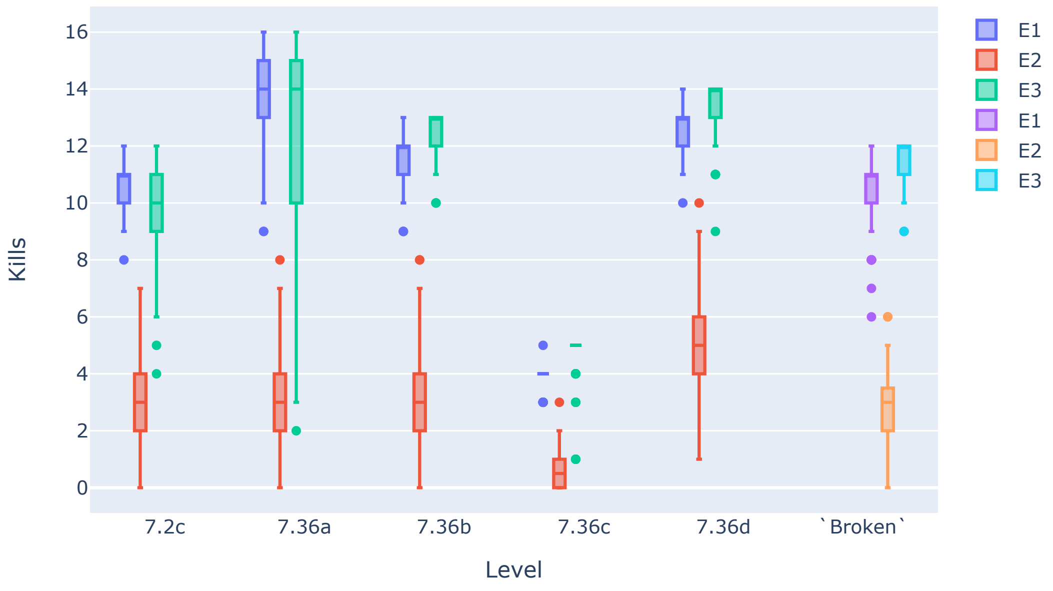


Figure 7.69: Experimental level testing in *Digdug*: Resulting *kills* in the ‘broken‘ level compared to the portability results, from 100 gameplays. E1: *High collector and killer*, E2: *High collector and low killer*, E3: *Low collector and high killer*.

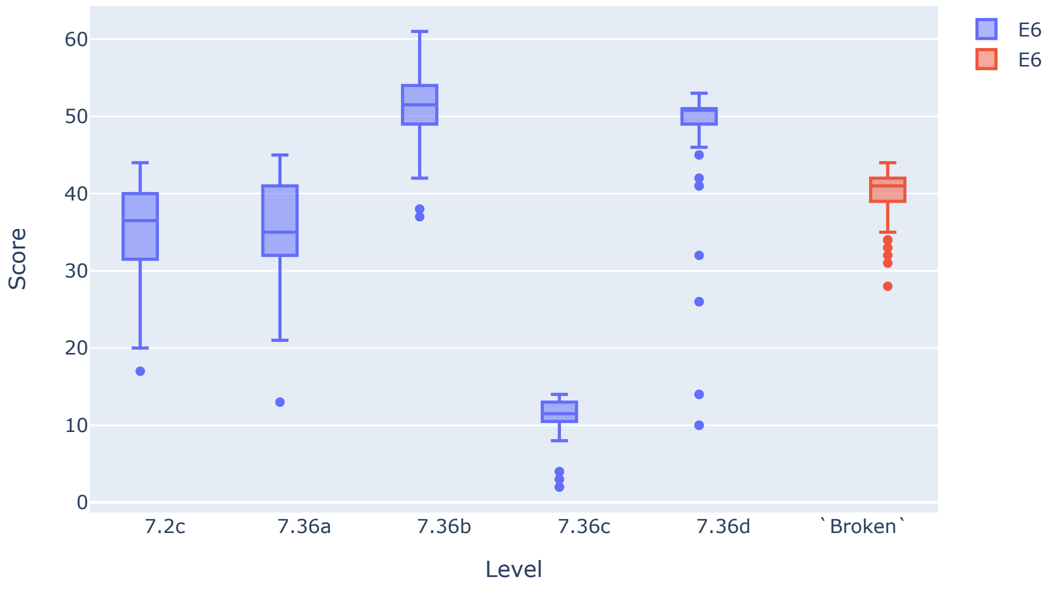


Figure 7.70: Experimental level testing in *Digdug*: Resulting *score* in the ‘broken‘ level compared to the portability results, from 100 gameplays. E6: *Low explorer and high scorer*.

7.4.4 Testing a ‘broken‘ level in *Sheriff*

The structure of the level and distribution of the barrels are similar to the original. However, I have increased the number of bandits from 8 to 35. I hypothesise that by almost quadrupling the number of enemies, the difficulty would increase drastically, being unmanageable for some players.

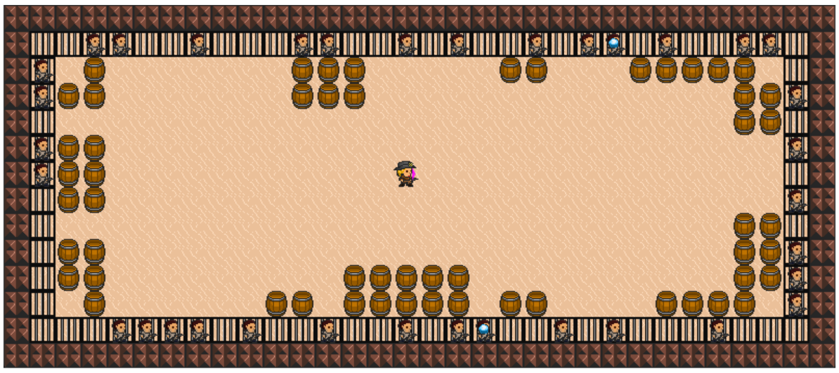


Figure 7.71: New ‘broken‘ level of *Sheriff* used for the exploratory work.

Table 7.17 presents the resulting stats for each *behaviour-type* agent when playing the new level, and Table 7.18 compares the results of the pertinent agents to the ones obtained in the previous levels. I include a graph for each of the features relevant to the agents and their selection: *curiosity* (Fig 7.72), *EoG* (Fig 7.73), *exploration* (Fig 7.74), *hits* (Fig 7.75), *interactions* (Fig 7.76), *kills* (Fig 7.77), and *wins*, represented as the total number of *victories* (Fig 7.78). It displays the results of the 100 play-throughs of the corresponding agents in the ‘broken‘ level compared to the portability experiment.

Quadrupling the number of bandits has less effect on the difficulty of the level than expected, but it still has an influence on the results. The resulting gameplay stats of the *Barrel shooter* (E5) are the ones impacted the most, as the average of interactions drops drastically, from 439.42 to 70.21. This agent dies quickly (149.38 EoG ticks) and often (1% win rate), not having enough time to destroy the barrels or interact with them. I believe that the main cause of this change is the indifference of the agent to killing enemies and, given the high number of these, it is unable to dodge the bullets. The average of EoG ticks doubles for the *Speed-runner* (E4) from 364.75 to 650.2, taking it longer to kill all the enemies and win the game. However, the win rate decreases only slightly (95% vs 100%), so, despite the drastic increase in the number of enemies, this agent can still win the game most of the time.

There is a slight change in the trend of the stats for the rest of the agents, but these are not radically different.

Sheriff 'broken' level						
	E1	**E2**	**E3**	**E4**	**E5**	**E6**
EoG	971.60 (140.36)	938.61 (209.22)	655.09 (207.90)	650.20 (200.11)	149.38 (125.43)	909.34 (240.75)
Victories	96/100	93/100	90/100	95/100	1/100	90/100
Exploration	90.71% (10.75)	94.86% (18.47)	20.61% (6.11)	71.21% (23.55)	6.49% (3.37)	88.47% (22.78)
Kills	16.51 (3.74)	32.73 (5.15)	32.11 (6.36)	32.71 (6.07)	9.18 (6.73)	32.02 (7.10)
Curiosity	32.61 (6.01)	38.68 (6.92)	26.75 (6.36)	42.56 (12.51)	30.96 (13.95)	75.2 (18.59)
Interactions	39.95 (20.85)	49.91 (29.66)	51.86 (29.47)	61.99 (33.81)	70.21 (65.98)	118.84 (64.57)
Hits	23.15 (4.50)	40.05 (6.45)	37.44 (7.05)	39.81 (7.53)	20.86 (9.08)	42.74 (9.29)

Table 7.17: Experimental level testing results overview. *Victories* are the total number of wins in the 100 play-throughs. The other dimensions are the average of the resulting gameplay stats and the corresponding *Std. Deviation.* Shaded cells are related to the *behaviour-type* of the agent. Those highlighted are directly or indirectly affected by the changes in the level. E1: *Survivor and low killer*; E2: *High killer and explorer*; E3: *High killer and low explorer*; E4: *Speed-runner*; E5: *Barrels shooter*; E6: *High curiosity and low interactions.* EoG: 1000; Number of enemies: 35.

Sheriff						
	7.2d	**7.43a**	**7.43b**	**7.43c**	**7.43d**	**'Broken'**
Total enemies	8	9	9	11	11	35
Total barrels	55	65	27	71	51	55
E1: *Survivor and low killer*						
Victories	98/100	99/100	100/100	99/100	100/100	96/100
Kills	3.28 (1.17)	4.02 (1.39)	3.89 (1.49)	4.78 (1.73)	4.87 (1.64)	16.51 (3.74)
EoG	992.58 (67.67)	990.32 (96.80)	1000 (0.00)	991.82 (81.80)	1000 (0.00)	971.60 (140.36)
E2: *High killer and explorer*						
Exploration	97.89% (10.65)	98.97% (9.13)	99.01% (8.22)	98.20% (9.86)	99.55% (3.08)	94.86% (18.47)
Kills	6.96 (0.31)	7.95 (0.50)	7.96 (0.40)	9.91 (0.71)	10.00 (0.0)	32.73 (5.15)
E3: *High killer and low explorer*						
Exploration	15.33% (4.06)	16.36% (5.15)	15.26% (6.03)	17.81% (5.46)	15.96% (4.72)	20.61% (6.11)
Kills	6.95 (0.50)	7.93 (0.70)	8.00 (0.00)	9.95 (0.50)	10.00 (0.00)	32.11 (6.36)
E4: *Speed-runner*						
Victories	100/100	97/100	98/100	99/100	100/100	95/100
EoG	364.75 (134.56)	360.53 (127.35)	363.59 (162.99)	417.71 (141.44)	418.7 (154.96)	**650.20 (200.11)**
E5: *Barrels shooter*						
Interactions	439.42 (343.26)	372.63 (311.24)	344.74 (305.19)	385.54 (314.72)	310.89 (275.01)	**70.21** (65.98)
Kills	5.83 (1.86)	5.89 (2.47)	6.22 (2.36)	7.46 (3.30)	6.65 (3.22)	9.18 (6.73)
Hits	35.96 (10.31)	38.19 (13.63)	17.49 (7.81)	39.86 (12.58)	29.28 (9.06)	20.86 (9.08)
E6: *High curiosity and low interactions*						
Curiosity	95.77 (22.26)	101.81 (25.46)	73.46 (18.53)	99.92 (26.57)	89.47 (18.41)	75.20 (18.59)
Interactions	122.52 (66.51)	127.46 (68.56)	106.28 (64.33)	125.93 (67.61)	116.98 (64.28)	118.84 (64.57)

Table 7.18: *Sheriff*: Portability and experimental level testing results comparison. *Victories* are the total number of wins in the 100 play-throughs. The other dimensions are the average of the resulting gameplay stats and the corresponding *Std. Deviation.* The stats shown per agent are the ones related to their proficiency. The dimensions highlighted are the ones directly or indirectly impacted by the changes in the 'broken' level. EoG is 1000.

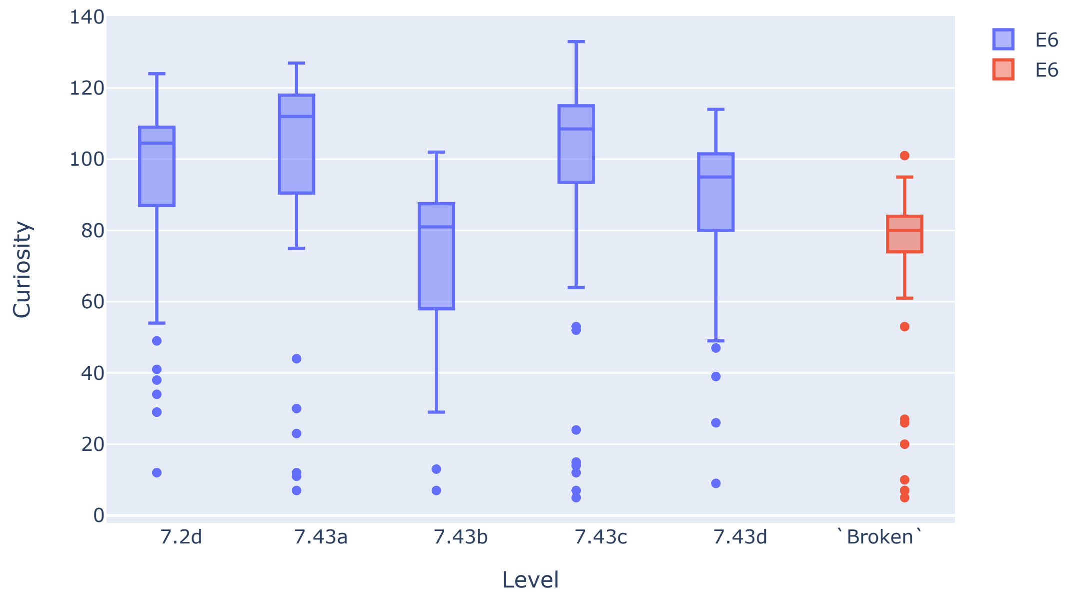


Figure 7.72: Experimental level testing in *Sheriff*: Resulting *curiosity* in the ‘broken‘ level compared to the portability results, from 100 gameplays. E6: *High curiosity and low interactions*.

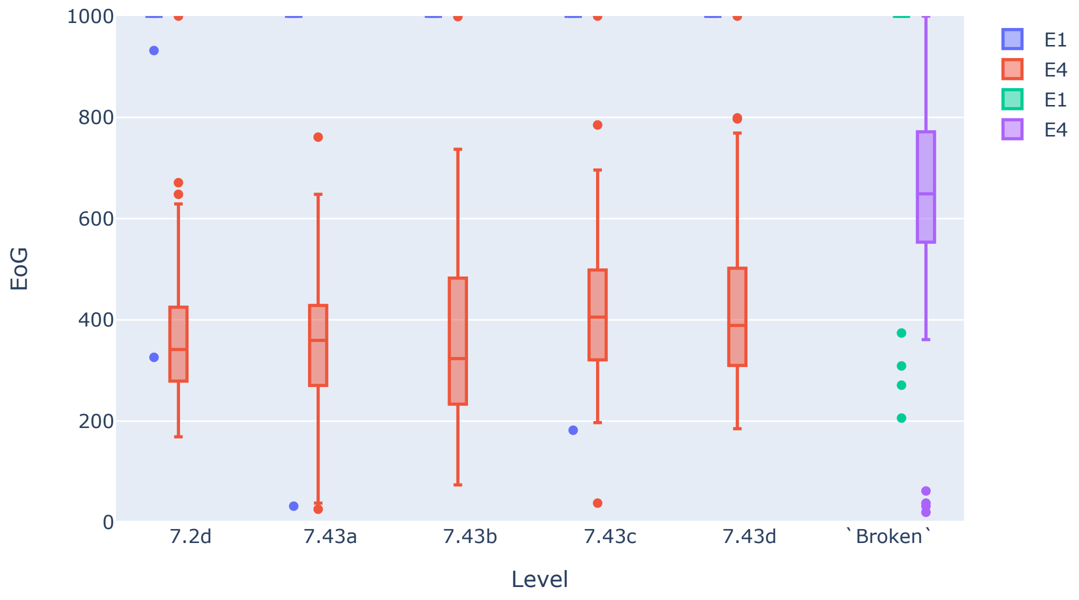


Figure 7.73: Experimental level testing in *Sheriff*: Resulting *EoG* in the ‘broken‘ level compared to the portability results, from 100 gameplays. E1: *Survivor and low killer*, E4: *Speed-runner*.

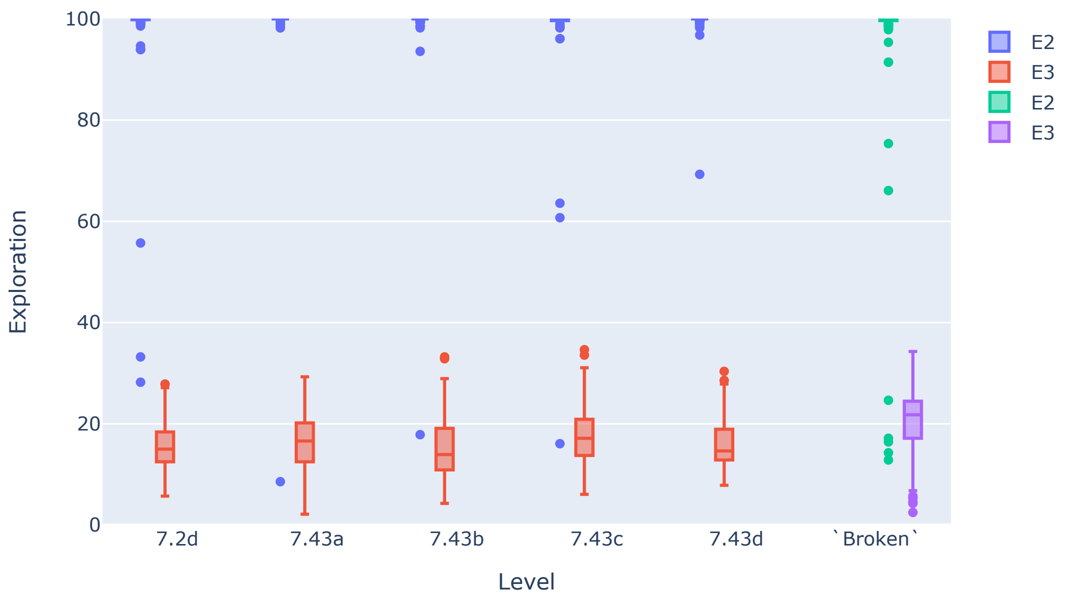


Figure 7.74: Experimental level testing in *Sheriff*: Resulting *exploration* in the ‘broken‘ level compared to the portability results, from 100 gameplays. E2: *High killer and explorer*, E3: *High killer and low explorer*.

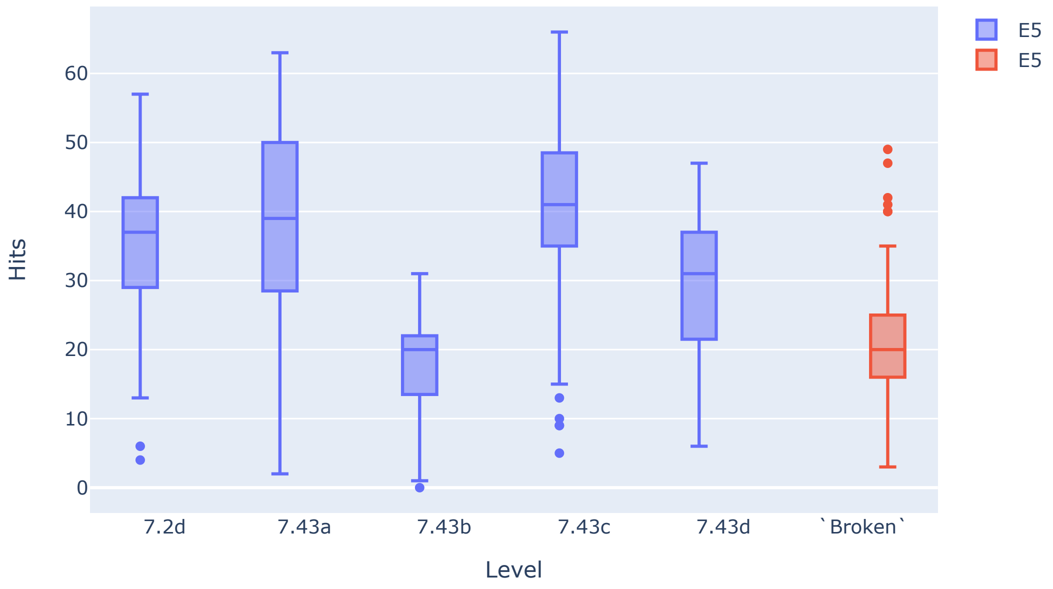


Figure 7.75: Experimental level testing in *Sheriff*: Resulting *hits* in the ‘broken‘ level compared to the portability results, from 100 gameplays. E5: *Barrels shooter*.

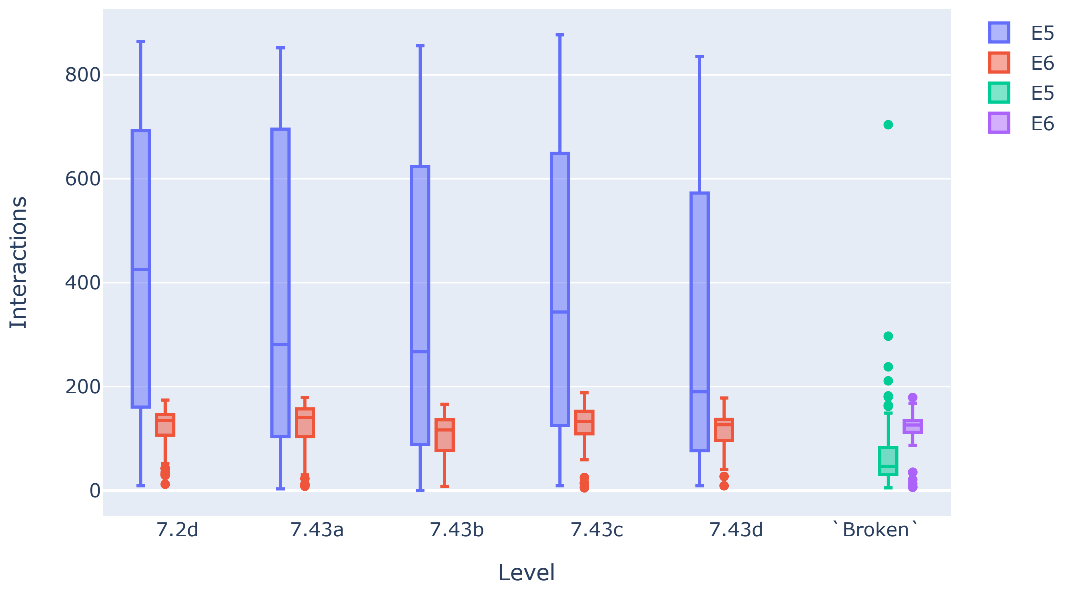


Figure 7.76: Experimental level testing in *Sheriff*: Resulting *interactions* in the 'broken' level compared to the portability results, from 100 gameplays. E5: *Barrels shooter*, E6: *High curiosity and low interactions*.

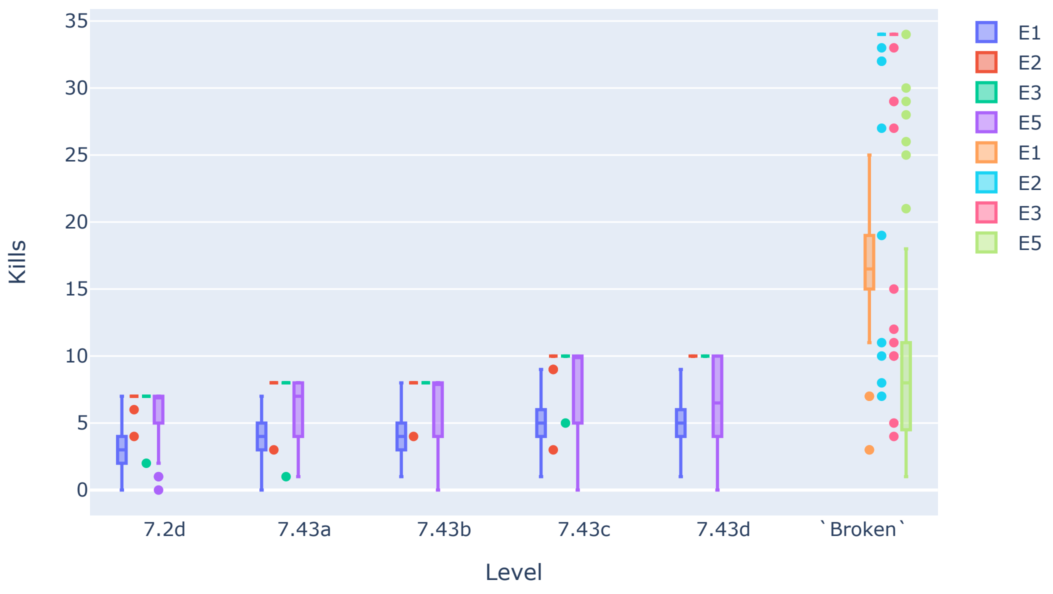


Figure 7.77: Experimental level testing in *Sheriff*: Resulting *kills* in the 'broken' level compared to the portability results, from 100 gameplays. E1: *Survivor and low killer*, E2: *High killer and explorer*, E3: *High killer and low explorer*, E5: *Barrels shooter*.

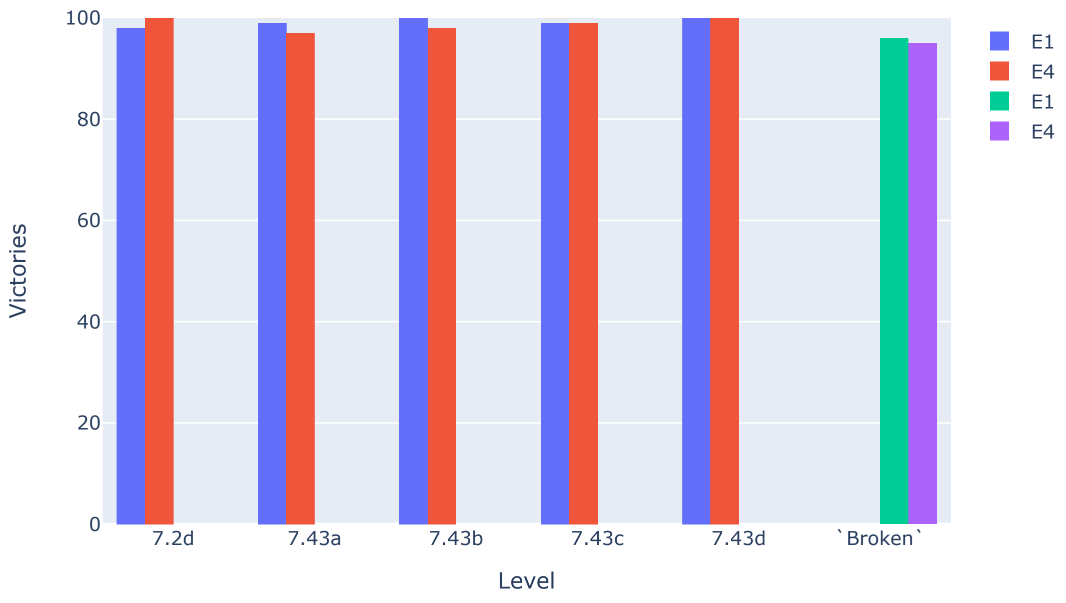


Figure 7.78: Experimental level testing in *Sheriff*: Resulting number of *victories* in the ‘broken‘ level compared to the portability results, from 100 gameplays. E1: *Survivor and low killer*, E4: *Speed-runner*.

7.4.5 Results overview

There is a discrepancy in the resulting stats in agents that have a direct or indirect connection to the features affected by the *problem* introduced in the level. At the same time, unrelated agents still achieve similar results that fit the expectations. Therefore, having available a range of diverse agents (a *team*) can help to identify issues in different aspects of the level.

7.5 Conclusion

This chapter concludes the work carried out to answer RQ2: *How to define, create, and use a team of GVGP agents with distinct behaviours to assist in the development and evaluation of games?*

I previously presented a methodology to use a *team* of GVGP agents to assist in the development and testing of games, and implemented an approach to generate this *team*. In this chapter, I have selected various agents from the available pool created for four different games to verify they can be applied as proposed. This selection is made based on their individual behaviour and achievements, which can be identified by their location in the feature space where the agents are generated. I speculate that these *behaviour-type* agents could be used for automated gameplay to achieve and react to the game in expected ways, find bugs, debug code, obtain game analysis, or check its performance with profiling tools.

I include an experiment to test the portability to different levels that would allow using this approach on newly created levels or after modifications have been made on existing ones. For this experiment, I include 4 new levels for each game with different characteristics. The final stats obtained after playing each of the levels follow a similar tendency for most of the agents. These results support that virtually all the agents identified are portable between levels and carry the strength identified in the original level when applied to new ones. Therefore, the agents from the *team* could ultimately be used to highlight issues in the design of new levels or on modifications applied to them when the resulting stats do not match the expectations. I include a preliminary work exploring this idea and use the *behaviour-type* agents to test a ‘broken‘ level in each of the games. The resulting stats of those agents whose proficiency is directly or indirectly related to the problem in the level are impacted, breaking the tendency found in the portability experiments. At the same time, those agents unrelated to the feature affected still achieve expected stats. Given these results, I infer that not every type of agent could identify issues in the levels. Only those whose behaviour is linked to the features affected by the problem suffer a variation in their stats. Therefore, having a *team* with a range of behaviours and proficiency related to various features in the game allows covering different type of cases and issues.

Consequently, the *team* of agents proposed could ultimately be used for testing. The approach can be enriched by defining and implementing a further selection of heuristics that enable further behaviour combinations for agent selection. A limitation to this approach is that, even when I have shown that is possible for these agents to operate properly when there is changes in the levels, modifications in the core dynamics of the game would require the heuristics to be adjusted. Similarly, if new goals come up as part of the iterative design process, new heuristics will need to be defined. All my experiments have been carried out in the GVGAI Framework, so the proof of concept has been implemented with a search algorithm in a general environment. Not every GVGP approach makes use of a forward model, so integrating the methodology proposed into a system without a forward model would require further investigation. First, it would be needed to define and provide a new diversification of behaviours adapted to its characteristics. Once the *heuristic diversification* is defined for the new scope, a similar approach followed for the generation of the agents using the MAP-Elites could be followed, as it simply requires a vector of weights that encode the behaviour. I believe that following the line of research started by my work to fit different areas and types of GVGP agents would allow adopting the approach in a more diverse range of games and frameworks, ultimately being applicable to the games industry and assisting on game development and testing processes.

The next chapter also looks into applying GVGP agents to games. However, instead of using *behaviour-type* agents to assist in its development process, I present an exploratory experiment that applies these GVGP agents within the game, taking the place of NPCs. To investigate this potential integration, I introduce a case study to analyse the effect that general agents have on the experience of the players when used in a player-vs-AI game. This work proposes a new line of research that I believe has great potential.

Chapter 8

Prospect: Take a Player Experience Perspective

This chapter presents an exploratory experiment that proposes studying the quality of general agents with distinct behaviours from a player experience perspective. The objective is to ultimately integrate GVGP agents as NPCs in video games. The material presented has been published in collaboration with Shringi Kumari in the paper [Guerrero-Romero et al., 2020].

8.1 Introduction and Motivation

Chapters 5, 6, and 7 looked at using General Video Game Playing (GVGP) agents as part of the game development process to ultimately be used to test games and levels automatically. I first proposed a methodology that applies general agents with different behaviours for this objective. Then, I presented the design and implementation of an approach to generate those agents and behaviours to form a *team*. Finally, I assessed the portability of the agents and presented a proof of concept that could ultimately use *behaviour-type* agents from the *team* to test levels. The agents and heuristics used are general within the GVGAI Framework to easily validate the approach in different games, but I envision its usage in other systems and games. This chapter also extends the research in GVGP by looking for applications for these agents in games. However, this work introduces using general agents within a game as Non-Player Characters (NPCs). It proposes making this integration possible by studying the quality of the agents from a Player Experience (PX) perspective instead of their performance. Inspired by the results from Chapter 4, it looks at answering the following research question (RQ3): *Can GVGP agents with distinct behaviours potentially be integrated into commercial video games as an alternative AI approach when these agents are studied from a Player Experience (PX) point of view?*

As pointed out extensively in this thesis, there is an active body of research creating and exploring the improvement of General Video Game Playing (GVGP) agents. These agents have been proven to play a wide range of video games at competitive strength without specific knowledge about them. However, their performance evaluation is usually limited to winning rates and scores. Put simply: research is concerned with how strongly agents play, assuming that higher play strength is better. This leaves little knowledge about how different agents uniquely impact Player Experience (PX): is playing against them actually enjoyable? Often, game designers want to elicit a specific player experience or emotion that has nothing to do with the strength of play. For example, the game *Journey* [Thatgamecompany, 2012] was designed around the player experience arc that the designers wanted [Chen, 2013]. I include this chapter as an addition to the rest of my research to investigate how GVGP agents can be designed keeping Player Experience in mind. I consider this an important initial step towards making GVGP agents viable for commercial use, as research towards such integration is practically non-existent. I present a brief exploratory case study that we carried out to encourage the research community to start looking into this unexplored line of research.

The connection between AI and PX is not new, but previous approaches link these areas from different perspectives than the one described in this chapter. A considerable part of the research in this area is related to assessing and improving the quality of Procedural Content Generation (PCG) techniques in games, called Experience-Driven PCG (EDPCG) [Yannakakis and Togelius, 2011]. The idea is modelling player experience to support its evaluation and dynamically influence the design of the generated content. Additionally, Guckelsberger et al. [2017] propose using computational models of intrinsic motivation to predict PX. Our preliminary work looks at the relationship between AI and PX from the opposite direction. Instead of using AI to influence or evaluate PX, it proposes assessing the GVGP agents from a PX point of view, for the particular case of using them as NPCs.

As introduced in Section 2.5.3, Non-Player Characters (NPCs) are usually hand-designed to give specific steps and goals based on the details of the game's design and purpose of the NPC. And, even when more advanced techniques are applied, as the use of MCTS in *Total War: Rome II* [Rabin, 2015, Chapter 25], their heuristics are still heavily tailored towards the game [Thompson, 2018]. Despite the research interest, the use of GVGP agents with heuristics that do not have specific information about the game has not yet found its way into commercial applications. As a first attempt to address this gap, we integrate two GVGP agents with different general goals as NPCs to create two versions of a player-vs-NPC competitive game and compare their effect on the experience of the player.

8.2 Definitions

The concepts used in this chapter are defined as follows::

Commercial-like games Games eventually meant to face players, opposite to games primarily used for research.

Capture the flag Type of game where the goal is to take control of an area, element or group of elements for a prolonged period of time than the opposing player or team. The symbol of what is being controlled is called *the flag*.

Player Experience (PX) Research field that focuses on studying the perception and responses of the people when playing a game [Bernhaupt, 2015]. It is based on a series of constructs that look at different experiences and feelings of the player, such as enjoyment, engagement, fun, presence, flow, and immersion, among others [Mekler et al., 2014, Caroux et al., 2015].

Intrinsic Motivation Psychological processes that energise and direct behaviour towards an activity and are generated by the activity itself [Ryan and Deci, 2017]. In terms of games, it refers to the belief that people play games for the sake of the experience that playing them provides.

Self-Determination Theory (SDT) It is the most well-established theory for intrinsic motivation. It states that people have innate psychological needs for certain experiences (competence, autonomy, and relatedness) that are satisfied during gameplay [Ryan and Deci, 2017].

Tension (pressure) Player Experience construct that measures the feeling of suspense when playing a game.

Perceived competence Player Experience construct that measures how *capable* the user feels while playing a game. The considerations to measure competence depends on the survey.

Enjoyment Player Experience construct that measures the interest and gratification while playing a game.

8.3 Exploratory Case Study Overview

This chapter presents an exploratory case study for a novel approach to study the relationship between General Video Game Playing (GVGP) agents and Player Experience (PX). The objective is to present a plausible line of research to achieve a successful integration of GVGP agents, with general goals, in commercial games. This is a complex problem to solve, as games come in myriad genres and designs that offer very different experiences for the players. With our case study, we take a first step in this direction by choosing a popular game type: *capture the flag*, which is representative of a large number of existing game features across genres.

We design an implement a new game called *Skulls and Tombstones*. The elements of the game, rules, and screenshot are detailed in Section 3.4.3. This game is implemented in the GVGAI Framework (Section 3.1) because it allows a quick set up of game-playing agents. We create two versions of the game, both assigning a general agent to take the place of *Player1* as a Non-Player Character (NPC), resulting in a player-versus-NPC competitive game. The goal of each agent is different: In *version A*, the NPC agent is driven by the need of winning and maximising its score; while in *version B*, the agent is also driven by the need of physically exploring the map, covering as many locations as possible. Details about each agent are given in Section 8.4. From casual play-tests of the two versions of the game, we discovered that playing against the exploring-encouraged NPC made the players feel more *under pressure* while still *enjoying* the game. On certain instances, the players actually preferred the tenser game experience. To formalise and confirm these relationships, we measure *tension/pressure* and *enjoyment* as Player Experience constructs, and hypothesise that 1) tension should be significantly higher for the version with the exploring-encouraged NPC; 2) there should not be significant difference in enjoyment between the two. Since the two agents present different behaviors when playing the game, we also include an open examination on their impact on player's *perceived competence*, with no specific hypothesis regarding this construct.

We conducted a preliminary study with 38 participants formulating the same hypothesis with promising results: *pressure/tension* was significantly higher when players played the *version B* of the game, and players felt significantly more *competent* when playing *version A*. However, the experimental settings were not good enough and the demographic data was lost, so we have decided to conduct the study again with a more rigorous approach, as I present in this chapter.

8.4 The General Agents (NPCs)

The GVGAI Framework provides the tools to set an agent as one of the players (*Player 1* in this case), so the players face the agent as a Non-Player Character (NPC). *Skulls and Tombstones* is a two-player game, so we can benefit from the search algorithms created as a result of the 2-player GVGAI Competition [Gaina et al., 2016]. Specifically, we use the *sampleMCTS*, a vanilla implementation of Monte-Carlo Tree Search (Section 3.2.2), as it shows a good performance in the competition. The two versions of the controller that lead to two versions of the game are:

8.4.1 Version A: *Std.* NPC

In version A, the *sampleMCTS* is assigned as NPC without modifications, as provided by the GVGAI Framework. Its heuristic is general and drives the agent intending to win by maximising the score. The pseudo-code of the value function is shown in Algorithm 15.

8.4.2 Version B: *Exp.* NPC

In version B, the *sampleMCTS* is modified following the *heuristic diversification* idea presented in Chapter 4. The value function of the agent is isolated and provided exter-

Algorithm 15 Value function for the *Std. NPC*, corresponding to *sampleMCTS* provided by the GVGAI Framework.
Nomenclature: $S' \leftarrow$ simulated game state; $H \leftarrow$ arbitrary high value.

```
function StdValue($S'$)
    $rawScore \leftarrow getScore(S', player1)$
    if $isGameOver(S')$ and $isLoser(S')$ then
        $rawScore$ += $H^-$
    if $isGameOver(S')$ and $isWinner(S')$ then
        $rawScore$ += $H^+$
    return $rawScore$
```

nally. Although the agent is for a 2-player game, the steps followed are similar, so the core of the algorithm and design parameters have not been modified. The main difference between *Std.* and *Exp.* comes from the goal (heuristic), which allows the agent to get rewards by visiting new positions on the map. The *Exp.* agent is also rewarded by the conventional rewards (winning and score), but it is encouraged to visit those tiles of the map where the agent has been fewer number of times. The reward takes the score as the difference between the current score of the game and the one obtained in the state reached by the forward model, while in *Std.* is just considered the latter. For balancing the *Exp.* heuristic, the final tuning is reached by trial and error analysing the trace of the rewards. We give high priority to exploring the map and visiting new positions while allowing the agent to score or win if it gets the chance. Algorithm 16 shows the pseudocode of the value function implemented. $C = 10$ is a constant that scales the heuristic value.

Algorithm 16 Value function for the *Exp. NPC*.
Nomenclature: $S' \leftarrow$ simulated game state; $lastScore \leftarrow$ score of the game in the previous state; $H \leftarrow$ arbitrary high value; $C \leftarrow$ constant to scale the heuristic value.

```
function ExpValue($S'$)
    $rawScore \leftarrow getScore(S', player1)$
    $scoreDifference = rawScore - lastScore$
    if $isGameOver(S')$ and $isLoser(S')$ then
        return $H^-$
    if $isGameOver(S')$ and $isWinner(S')$ then
        return $H^+$
    $position \leftarrow getAvatarPosition(S', player1)$
    if $isOutOfBounds(position)$ then
        return $H^-$
    if $nTimesVisited(position) > 0$ then
        $reward = -C * nTimesVisited(position)$
    else
        $reward = C$
    return $reward$ += $(scoreDifference * C)$
```

8.4.3 NPC's behaviour comparison

Increasing the score in *Skulls and Tombstones* requires performing two actions: 1) get a skull (which does not affect the score); 2) carry the skull to the tombs. *Std.* exclusively focuses on winning and maximising score. When the forward model is not executed enough times, the agent cannot discover the potential score rewards, facing a large, flat reward landscape. As a result, the NPC's decision-making becomes highly arbitrary, resulting in static behaviour or circling randomly around the map.

On the other hand, although *Exp.* takes winning and maximising the score into consideration, this agent is also driven by an exploratory heuristic. This heuristic causes the agent to move all over the map and visit new locations (tiles) or those that it has previously visited the least number of times. Therefore, the agent is likely to collect the skulls scattered around the map and see the reward given by the score change. As a result, it displays a more dynamic and competitive behaviour than *Std.*.

8.5 Participants

We recruited 50 participants in person by advertisement in the University and snowball sampling. We had to exclude 15, ending up with a total of 35 participants - 15 female and 20 male. All participants were 18 or above years old (20 between 18 to 24; 13 between 25 to 34; 1 between 35 to 44; and one older than 45). Players were picked from a diverse pool and were not limited to just students or any one category. They were spread across demographics and gaming abilities. None of the participants had played *Skulls and Tombstones*, but all of them had played games before and were familiar with the studied game genre and controls. 12 participants used to play digital games at least once per day; 13 at least once per week; 8 at least once per month; 1 at least once per year; and one less often. 34 had played platformers before, and 27 had played shooters before. We provided all the players with the game information sheet before they consented to take part in the study.

8.6 Material

The *jar* executable corresponding to each version of the game and the original VGDL code are available in an OSF repository [Guerrero Romero and Kumari, 2020]. Appendix F includes the consent form, information sheet, and list of questions provided to the participants.

Games The two versions of *Skulls and Tombstones*, featuring each of the NPCs described, are exported as an executable standalone to be played on an iMac using the keyboard. The game has no music or sound effects and is played with the four arrows where only corresponding directional actions are allowed ($\leftarrow$, $\rightarrow$, $\uparrow$, and $\downarrow$). The aim of the game is simple and visual enough for players to have clarity about the goals and rewards. Game sessions quit after 30 seconds each. Players play the entire game for 3 rounds, 90 seconds in total.

Questionnaires We use the Intrinsic Motivation Inventory (IMI) [Deci and Ryan, 2003] and Player Experience of Need Satisfaction (PENS) [Ryan et al., 2006] (7 point Likert) scales. These are frequently used and validated questionnaires for Player Experience and are most directly related to the motivational model we use. IMI is based on the Self-Determination Theory (SDT) and comprises of seven sub-scales: *interest/enjoyment*, *perceived competence*, *effort*, *value/usefulness*, *pressure/tension*, and *perceived choice*. The *interest/enjoyment* sub-scale is considered the self-report measure of intrinsic motivation, while the *pressure/tension* sub-scale captures outer pressures to perform an activity. PENS is designed to capture SDT components in players and has sub-scales for the three basic needs (*competence*, *autonomy*, and *relatedness*), as well as presence and intuitive controls. PENS is specifically designed for video games, while IMI is general. We have decided to utilise both because the former does not have all the components we are interested in, like *enjoyment* and *tension*. We choose particular sub-scales from each of these scales to assess specific Player Experience constructs to suit the design of our study. We adopt three IMI sub-scales: *pressure/tension* (5 questions), *perceived/intrinsic competence* (6 questions), and *enjoyment* (7 questions); in addition to the PENS' *competence* (3 questions) sub-scale. Consequently, our questionnaire is comprised by a total of 21 questions to assess the experience of the players in each version of *Skulls and Tombstones*. These questions are included in Table F.1.

Logs Gameplay data is logged from each session containing: name of the AI integrated into the game, human and NPC score per game tick, number of total game ticks (400), final scores achieved by each, and winner (*human*, *AI* or *draw*).

8.7 Experimental Set-Up

We examine whether the difference in the general NPCs' behaviour impacts the following Player Experience constructs: *pressure/tension*, *perceived competence*, and *enjoyment*. Our primary hypothesis is that the *Exp.* NPC, whose behaviour fits the exploratory characteristics of the game created, would feel harder to beat, leading to players experiencing a higher *pressure/tension* when playing against it. We further hypothesise that *Exp.* would not be less enjoyable than *Std.*, even if players feel higher *tension* against this NPC. Lastly, we want to explore in which version of the game players perceive themselves as more competent.

The study carried out is a between participant set-up [Shaughnessy et al., 2000], where two different groups of participants play the two different versions of the game (with *Std.* NPC and with *Exp.* NPC). The participants recruited alternate between the conditions and the researchers are present throughout the process. Participants are given an information sheet along with instructions about how to play the game. They are also asked for their consent and demographic details (age, gender, and gaming experience). Players are in a quiet zone for the duration of the study, play the game three times, and answer some questions. These steps are explained in the information sheet provided at the start of the experiment. Their gameplay is logged and stored in real-time, while the chosen sub-scales of the IMI and PENS questionnaires are filled out after finishing the game, one after the other. Players are debriefed at the end.

We carried out play-tests during the development of the game, and we learned that the game and its controls were simple enough to understand in one play-through. Therefore, playing three times gives players enough gameplay experience to answer PX related questions.

8.8 Results

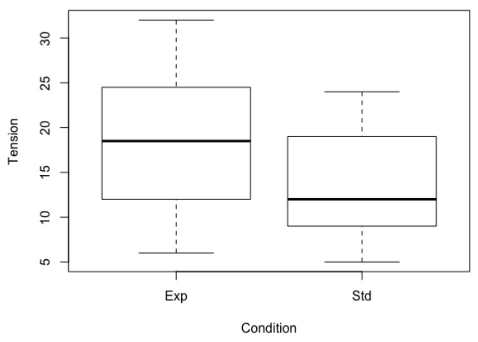
This section presents the results obtained for each Player Experience construct (*tension*, *enjoyment*, and *perceived competence*) in each version of the game (*Std.* and *Exp.*).

Tension The players feel more tense while competing against the *Exp.* NPC (Fig. 8.1a). This is demonstrated by a nearly significant two tailed t-test (t=2.02, df=33, p=0.051) in support of the hypothesis with an effect size of Cohen's d=0.68. The effect size lying between a medium (0.5) and large (0.8) based on Cohen's suggestions makes us consider the results in the direction of the hypothesis. The higher tension is expected for players to feel more nervous while competing against the NPC that can explore the game map better. This result follows a tendency similar to the one we observed in the pre-study. However, in that case, we found the result to be significant.

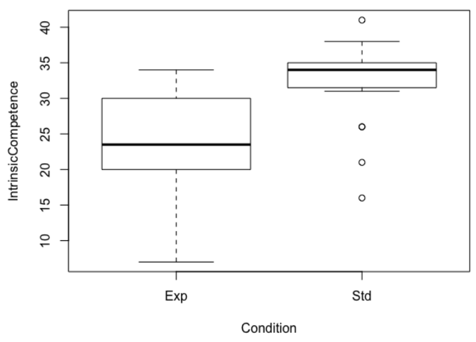
Enjoyment There is no significant difference between how enjoyable players find the two versions of the game (t=0.99, df=33, p=0.329). We expected that the version with the *Exp.* NPC would not be less enjoyable than the one featuring the *Std.* NPC.

Competence We had no particular hypothesis regarding *competence*, yet we wanted to explore this component as well. As measured by IMI's *perceived competence*, players feel significantly more competent when competing against the *Std.* NPC (Fig. 8.1b), demonstrated by a two tailed t-test (t=−3.93, df=33, p< 0.001) with a large effect size of Cohen's d=−1.33. Interestingly, *competence* as measured by the PENS sub-scale does not show a significant difference between the two conditions (t=−1.7, df=33, p=0.099).

We have used a quantitative method to evaluate Player Experience. Our sample size (35) could have been larger. There are non-aligning results between the two measures of *competence* provided by IMI and PENS, and the pre-study results suggest that a larger-scale replication of the study would make this measure clearer. *Tension* from IMI is not designed to capture game-like pleasurable tenseness, but the item has faced validity for capturing data to test our hypothesis. Given these initial results, we advise



(a) *IMI: Tension*



(b) *IMI: Perceived Competence*

Figure 8.1: Most relevant results for *Skulls and Tombstones* with *Exp.* and *Std.* NPCs.

using other research methods to complement the quantitative analysis for more diverse investigations; like qualitative methods for an in-depth analysis, and eye-tracking if the researchers or designers are interested in real-time reactions.

8.9 Discussion

The experimental study described in this chapter has not been conducted to show if one agent (NPC) was *better* than the other. The objective is to study how two general agents with different behaviour affect specific components of Player Experience (PX) when integrated as NPCs in a game. We found that playing against the *Exp.* NPC creates more tension in comparison to the *Std.* NPC. While IMI considers *tension* as a negative indicator of intrinsic motivation, it still can constitute an important experiential factor in games [Seif El-Nasr et al., 2007]. Some games, for example, horror games or arcade-style games like *Super Meat Boy* [Team Meat, 2010], are designed to make the players feel tension. Moreover, tension does not necessarily equal a better or worse experience, supported by the results obtained in *enjoyment*, which do not show any significant difference between the two conditions. A specific degree of experienced tension is often an important design goal for game designers. From the IMI *perceived competence* results, we found that the players felt more competent when playing against the *Std.* NPC. However, according to PENS, we found no significant difference in perceived competence of the players whether they played against the *Std.* or *Exp.* NPC. The *competence* sub-scale of PENS entails items that assess whether the game's challenge was perceived to *match* the player's skills. The underlying rationale here is that a better match should result in higher perceived competence, as players perceive successes as 'well-earned' and failure as 'near-miss'. In contrast, the IMI items of *perceived competence* sub-scale focus on how well a person can perform the given task regardless of the difficulty *match.* Based on PENS results, the two versions do not differ significantly in perceived *competence* in terms of difficulty-skill *match.* As a result, designers and researchers need to choose what they are interested in, as there is a large range in which challenges can be presented to players. For some games, an interaction-difficulty match could be an important aspect, while it could be inconsequential for games that do not pose interaction challenges but focus on emotional ones.

The approach proposed allows inspecting general agents as NPCs in terms of the experience that the designers want their players to have. For instance, if a designer wanted the game to have more *tension* they could use the *Exp.* NPC, while if they wanted the players to feel more *competent* (in IMI sense) they could use the *Std.* NPC instead. It is possible to reduce the skill level of the AI by reducing the number of the iterations of the algorithm, as this is directly proportional to the playing skill up to a certain degree [Nelson, 2016]. This kind of tweaking becomes possible if the designers know what experience they want the players to have. They could then evaluate the agent for that experience or adjust the heuristics to fine-tune the player's journey. Furthermore, if AI researchers also adopt this approach of evaluating agents using player reactions, the field could be broaden in terms of diversely behaved general AI. I believe that this extension suits the currently growing landscape of games.

8.10 Conclusions

The preliminary work presented in this chapter attempts to build a bridge between General Video Game Playing (GVGP) agents and Player Experience (PX). Our experimental case study aspires to make it possible for game designers to look at PX to integrate GVGP agents in their games. To this end, we evaluate GVGP agents in how they impact player motivation rather than their 'raw' performance measured in scores or winning rates. This work presents a novel approach where we apply PX measures to games with GVGP agents. As a first step, we use the GVGAI framework, propose the use of general agents as NPCs, and focus on a particular search algorithm: MCTS. The GVGAI Framework was originally created to compare general agents, not as a video game creation tool. Hence, the game controllers and interface are not ideal and could have an impact on the experience. It should be possible to replicate the approach with more polished game prototypes built in other frameworks more suitable for game development. The work demonstrated can be extended to use other search algorithms or even learning algorithms. As a next step, a similar study could be conducted using *Unity*, which provides the *ML-agents* toolkit that can be used to train agents with learning techniques [Juliani et al., 2018]. The general heuristics showcased in this chapter are quite favourable to the type of game under consideration. I would like to encourage researchers to create more nuanced and diverse general heuristics that can be evaluated for different experiences and emotions when they are accommodated to the needs of the type of game under consideration. Some examples of such general heuristics have been proposed in Chapter 5.

Existing work is primarily looking at GVGP agents from the perspective of the game and how well they perform in it, excluding the player out of the equation. However, assessing the behaviour of the AI from the perspective of the player is likely to succeed. We have demonstrated how in a short time, it is possible to make general agents with varying heuristics, integrate them in small games, and then test how each of them affects the experience of the player. Using GVGAI has allowed us to make a simple game, but visual feedback for players is limited. We iterated over the game design and ran casual play-tests until it was considered self-explanatory and engaging. We assigned colours on the tombs to inform about changes in score to overcome the User Interface (UI) limits of the framework. We have chosen specific PX measures like *tension* because the *Exp.* NPC is expected to behave in a fashion that would be perceived as more purposeful and make the gameplay tenser. We expected that, in this case, such suspenseful tension would not make the game less enjoyable than playing against the *Std.* NPC, measured with *enjoyment*. Researchers could pick PX constructs based on the kind of game they choose and the experiences they are interested in studying. Using PX questionnaires is only one of the methods available; it should be possible to carry out more open-ended investigations with qualitative research. I believe that our proposed line of research would eventually extend the option for game designers to use general agents, particularly as NPCs, for successfully eliciting desired Player Experience (PX) in commercial games.

Part III

Conclusions and Future Work

Chapter 9

Conclusions

This thesis is driven by the ultimate goal of finding applications to existent General Video Game Playing (GVGP) agents instead of creating new solutions or improving current ones. It is inspired by the existence of a variety of behaviours based on the motivations of the player and changes in the way they act and react to the game. It looks at applying this diversity to GVGP by providing the agents with goals that go beyond winning the game without changing the core of the algorithms. The objective is to elicit different behaviours and, ultimately, integrate GVGP agents in games by assisting in their development and testing or by replacing existing AI techniques used within them. An example explored of the latter is using agents with general heuristics to take the place of Non-Player Characters (NPCs). The experiments, focused on planning algorithms and carried out in the GVGAI Framework, are structured in three differentiated blocks, each of them looking at answering a research question and conclude as describe below:

RQ1 *Which general heuristics can be defined and implemented beyond the goal of winning the game, and how does each of these affect the performance and behaviour of existing GVGP agents when it is the only variation in the algorithm?*

Both the performance and the behaviour of GVGP agents vary when algorithms are provided with differentiated goals by *heuristic diversification*. This concept, introduced in Chapter 4, refers to the isolation of the value function in search algorithms to provide the heuristics externally without having to modify their foundation. I define and implement four general heuristics that elicit different goals: *winning*, *exploration*, *knowledge discovery*, and *knowledge estimation* and provide them to GVGP agents to compare their performance. I use a group of controllers of different characteristics: OSLA, OLMCTS, OLETS, RHEA, and RS. When measured by criteria related to the goals provided to the controllers (e.g. final exploration of the level), the performance of the group of agents changes depending on the heuristic. Therefore, the strength of an algorithm depends not only on its characteristics but also on its heuristic and motivations. Therefore, these should be considered when developing GVGP solutions. This conclusion opens a new line of research to apply the GVGP agents in tasks beyond just playing the game and ultimately assist in their development. I investigate this idea further in the thesis.

RQ2 *How to define, create, and use a team of GVGP agents with distinct behaviours to assist in the development and evaluation of games?*

The methodology introduced in Chapter 5 proposes the use of use General Video Game Playing (GVGP) agents for assisting in the game development and testing processes. It presents a series of goals to be applied to the general agents to play the game in different ways. It is argued that having agents focusing on targets that go beyond simply winning the game leads to specialists with distinct gameplay styles that can be used to extract diverse information from the game. Each automated play-through of the agents can be leveraged to generate reports and a logging system to check if the game under evaluation fulfills the expectations. The information retrieved can assist to detect bugs, balance the game, or tweak its parameters. Because of the independence of the rules given by the generality of the agents, the approach allows an early integration in a game under development without requiring major modifications when new levels are included or when existent ones are extended or modified. Using general algorithms for automated game testing provides portability and flexibility non-existent in the current approaches.

I present and implement a method in Chapter 6 that allows creating a *team* of GVGP agents, as the one proposed above, and is general. This generality enables the technique to be applicable and extended outside the scope of my work. By using *heuristic diversification*, I define the characteristics of a GVGP agent and a heuristic that allows assigning a list of goals externally with a description that is easy to define and evolve. The approach presented applies the MAP-Elites algorithm to generate diverse behaviours for the agents by generating and locating their description defined in a feature space constituted by the resulting stats from the play-through of the agents. I define and use five goals (heuristics) that can be applied (or not) to various games, from which I derive a list of features to use to form these spaces. However, the flexibility of the procedure allows providing different goals and features to adapt to the characteristics of the game and apply in different circumstances. The approach has been implemented in the GVGAI Framework to generate a *team* of GVGP agents to four games with different characteristics. This *team* is assembled by the collection of resulting maps, and it is possible to identify *behaviour-type* agents based on their location on the maps and the resulting range of features. This solution, in contrast to previous ones, proposes to make the selection of the agents based on the behaviours needed after the execution of the algorithm. It allows having at the disposal of the user a diverse pool of agents at all times, in case the requirements change.

In Chapter 7, I illustrate guidelines that can be followed to identify different *behaviour-type* agents, independently of the map generated or the game used. When the GVGP agent plays the game using the selected description, it is expected to achieve specific objectives related to its location in the feature space. Fulfilling these tasks cause to elicit a particular behaviour in the agent that is reflected in the resulting stats from its play-through. The generality of the GVGP agents allows them to play levels different to the one they were generated at. I confirm that these agents also carry their strengths and characteristics when used in new ones. The demonstration is executed in the four games I generated the *team* for, by using the agents in new levels with different properties and distribution of the elements. The resulting stats related to the features of the behavioural space of the agent follow the same trend across the levels. Therefore, it would be expected that the agents can ultimately detect issues and design flaws in new levels integrated into the game or after modifications in the existent ones when the resulting stats do not meet the expectations. This theory is also investigated in Chapter 7, where I design a faulty level in each of the games based on the original one used to generate the pool of agents. The stats of the agents whose identified behaviour is directly or indirectly related to the issue in the level are affected, while those unrelated still fit the expectations. Therefore, these results support the argument that a *team* of agents can be used to assist in the development and testing of video games. The approach proposes looking at the resulting stats of identified *behaviour-type* agents to highlight flaws in the design or problems resulting from the modification of the game or levels. Having a range of behaviours related to various features allows covering different types of instances and potential issues. The extension and portability of the approach to other frameworks and games are possible given its generality and flexibility. I conclude that GVGP agents can be employed to assist in the game development process by following this approach. Ultimately, I believe that these agents can be used by running automated gameplay to find bugs, flaws in the design, debug code, or even check the performance by triggering profiling tools. I include details about these expectations in Section 9.1.2.

RQ3 *Can GVGP agents with distinct behaviours potentially be integrated into commercial video games as an alternative AI approach when these agents are studied from a Player Experience (PX) point of view?*

The novelty work explored in Chapter 8 encourages starting looking at General Video Game Playing (GVGP) agents from the perspective of the player instead of the performance of the algorithms. The exploratory experiment carried out in the GVGAI Framework is considered a first step into tackling this line of research to ultimately being able to integrate GVGP agents within games as an alternative AI approach. We use the IMI and PENS questionnaires to measure the Player Experience (PX) constructs of *tension*, *competence*, and *enjoyment* in a player-vs-NPC capture the flag type of game created for the study. Two versions of the game are developed, and their variation comes from the GVGP agent assigned as NPC, with a difference in heuristic between the two: *winning and score* (*Std.* NPC) vs *exploring-encouraged* (*Exp.* NPC). We analyse the PX constructs in both versions. These PX constructs are related to the players' perception when playing against each of these general NPCs. The results observed are related to the characteristics of the game and the behaviours elicit on the agents by the general heuristics. We focus on those particular PX constructs because they are related to our hypothesis. However, a similar procedure can be used with further or alternative research methods to suit the needs and expectations of the game. The work demonstrates that, in a short period of time, it is possible to integrate agents with general and diverse heuristics into a game and study how each of them affects the experience of the player. Given the results for the two versions of the game implemented, we can identify two distinct scenarios. If a designer expects the game to have more *tension*, they can use the *Exp.* NPC, while if they want the players to feel more *competent* from an IMI point of view, they can use the *Std.* NPC instead. By making a collaboration of GVGP and PX part of the development process it may be possible to, for example, see general agents take the role of Non-Player Characters (NPCs) in games. However, further investigation is still required, as it is currently very lacking.

In summary, this thesis demonstrates that it is possible to extend the research and application of General Video Game Playing (GVGP) agents when the existent solutions are driven by goals that go beyond just winning the game. The generality of the GVGP agents allows them to be independent of the rules and parameters of the game. This generality provides the agents with flexibility and adaptation to changes, being transparent to the addition of new levels, elements, or modifications on existent ones. Therefore, GVGP agents can be incorporated in the early stages of the game cycle to assist in its development and testing processes or integrated into the game itself. The possibilities are endless, and I encourage the research community to extend and continue this work and the different lines of research opened by it. I discuss future work in the next section.

9.1 Future Work

To conclude the thesis, I discuss future work related to each of the areas investigated and invite the continuation of the various lines of research emerging from my work. I have used the GVGAI Framework because it provides the support to run general agents and includes a variety of sample search controllers that can be easily modified. It facilitates executing the experiments and validating the approaches in several games with different characteristics. However, I believe the strategies defined should apply to GVGP outside the limitation of this framework. It should be possible to transfer the methods defined to other systems and game engines, and I encourage doing so.

9.1.1 *Heuristic diversification* in GVGP

The heterogeneous results in Chapter 4 are promising and open different lines of research, from improving existing General Video Game Playing (GVGP) by applying *heuristic diversification*, to extending the understanding and study of alternative heuristics and behaviours of general agents. The thesis focuses on the latter by investigating the behaviour driven by the different heuristics and looking for applications for the agents to integrate them in the game development process.

The *heuristic diversification* approach presented applies to search algorithms, corresponding to a specific part of GVGP. A similar concept should be defined to other related areas, but this extension was out of the scope of the thesis. Interestingly, a comparable technique has been recently introduced in Reinforcement Learning (RL) [de Woillemont et al., 2021]. The procedure provides a diverse range of play styles to a configurable agent by making the rewards external and agnostic to the algorithm.

Heuristic diversification can be applied to enlarge GVGP techniques by providing heuristics different from merely winning and maximising the score to improve the overall performance of the general algorithms. These heuristics can, for example, drive the agent to cover the level or gain practical knowledge about the dynamics of the game. It is sensible to think that all the heuristics presented in Chapter 4 may come in handy at different stages of game-playing. An option would be to design a high-level meta-heuristic algorithm that could combine or choose between different agents and heuristics in-game. Having an available set of general algorithms with different objectives (provided by the pertinent heuristic) would allow to accommodate to different situations that emerge during gameplay, and to switch behaviour in response to the environment. For instance, at the first stages of the gameplay, agents could use KDH and KEH to achieve a better understanding of the game. These heuristics penalise losing the game, so it is a safe search until certain conditions have been met during the play-through. At this time, the agents could use WMH or EMH; or even a combination of both in a multi-objective setting [Perez-Liebana et al., 2016]. Following this approach, the GVGP agent could achieve victory and maximise the score, while being influenced by the discoveries found and encouraged by exploring the level. In terms of using the right controller, it would be reasonable to run experiments as the one presented in Chapter 4. The user could choose the agent based on the resulting performances: when using just one heuristic, it could pick the controller with the best results for that particular heuristic. On the contrary, if several heuristics are brought together, the reasonable choice would be to select the agent with steady results for the heuristics involved. For example, in this particular case and based on the results discussed in Section 4.6, if I were looking to combine WMH and EMH, the choice for the controller would be between using RS or OLETS. However, if knowledge were also included, it would be preferable to use RS or OLMCTS. Future work could combine the idea of exploring and exploiting the level, making the most from the knowledge acquired to improve the performance in terms of winning.

Above, I described a potential line of research that looks at the application of *heuristic diversification* to improve the strength of GVGP agents from a winning point of view. Some work related to this proposal has been carried out in collaboration with Anderson et al. [2019]. We applied an Ensemble Decision System (EDS) for General Video Game Playing (GVGP) by comparing different variations constituted by particular combinations of the controllers and heuristics employed in the experiments of Chapter 4. The heuristics described in Section 4.3 were adapted to the experiment and modified to, among other changes, reward the winning EoG states. Two components constitute an EDS: the *voices*, which are the different algorithms used to play the game and involved in suggesting the following action, and the *arbitrator*, which ultimately decides the action to take based on a policy. Six different EDS variations were defined and implemented, one of them including all the heuristics as voices (OLETS with WMH, RS with EMH, RS with KDH, and OLETS with KEH), and the others with different combinations of selection policy, controllers, and WMH or EMH as heuristics. Each of these solutions was compared between them and to the individual controllers (OLMCTS, OLETS, RHEA, and RS) by playing a selection of games from the GVGAI Framework. One of the variations of the EDS was able to reach a win rate of 53.59%, very close to OLETS' 55.69% (first in the final ranking), while winning 2 more games than the rest of the solutions. Although the EDS solution was not able to outperform the individual algorithm, we concluded that it maintained its strength. The use of different general behaviours provided that particular EDS with the flexibility to succeed at more games. The results were promising, but it was concluded that further experimentation was required to reach the potential on flexibility and strength of the EDS solutions. Future work could look at developing further variations of the system, including selection policies or types of arbitration, and compare EDS to portfolio approaches. Since the work was carried out, new potential general heuristics have been identified and listed (Section 5.3), resulting in the implementation of some of them (Section 6.5). These behaviours extend or improve the ones applied to the EDS (WMH, EMH, KDH, and KEH), so future work could also explore the usage of alternative general heuristics. This extension could provide a richer diversity of behaviours and reactions to the game.

Last but not least, although there is still room for improvement, combining the agents provided with different heuristics could gain a relatively good understanding of the game. As discussed above, this information could help improve the agents, but it could also assist in the PCG of levels and games. Similarly to general game-playing agents being used to validate the rules generated for board games [Browne and Maire, 2010], an approach could incorporate GVGP agents to evaluate the strength of the generators in the domain of video games.

9.1.2 Using a *Team* of GVGP agents: extensions and applications

Part of the methodology envisioned in Chapter 5 has been defined and implemented in Chapters 6 and 7. However, there is still considerable open work that I encourage the research community to continue. I believe this line of research can potentially benefit the video games industry and facilitate the game development process.

In this thesis, I present a solution that allows defining and creating a *team* constituted by the assemble of agents distributed in several behavioural spaces. I describe a procedure to identify the skills and expectations required from each of them. In sports, a *team* refers to the group of people that compete in the same club, regardless of the activity involving collaboration or being carried out individually or in parallel. The concept of *team* in my case is related to each member independently serving the same purpose: ultimately assisting the game development and testing processes. However, it should be possible to extend the use and start considering the *team* as a collaboration between the agents as well. In this sense, agents with differentiated behaviours and achievements could gather information of their play-through and share it with the other members of the *team*. It should be possible to have several agents playing the game simultaneously and communicating their findings in real-time to consider them for their actions. These plausible extensions of the methodology were out of the scope of my work, so the experiments only focused on having the agents play independently.

I have focused on particular areas of the methodology envisioned, to define a solution to generate the *team* of agents that has been implemented and integrated with the GVGAI Framework. This tool is practical for research and to validate the approach in several games, but it is not intended for actual game development. The next step would be to transfer the idea to a framework or engine designed with that purpose. The MAP-Elites algorithm is quite simple, so its implementation in a new system should be straightforward. Although the approach, agents, and heuristics are general within the GVGAI framework, no implementation is actually independent and directly transferable between systems. The idea is valid and the code can serve as a reference, but it is still required to revise the existent implementation to adapt it to work on a new system. The definition and integration of the GVGP agents and heuristics depend on several factors. First, the availability of a forward model is crucial to facilitate the use of the same algorithm (OLMCTS) and *heuristic diversification*. Both the agent and heuristics would still need to be adapted to fit the characteristics of the platform and its interface so it can communicate with the games, execute the forward model, and get the information required for the heuristics to work: state information, score, NPCs description, resources, etc. The strength of the approach defined comes from its generality and flexibility. Once a successful integration into a system, the process of generating agents would be available at every moment for any of the games supported. Therefore, a *team* could be at the disposal of game developers to run executions of the game automatically whenever they require it. As discussed in the corresponding chapters of the thesis, the automated gameplay of different *behaviour-type* agents can provide information about the games and levels to identify issues and design flaws on them. I argue that their purpose does not stop there. The solution provides means to play a game without a human player while achieving diverse gameplay. Automated gameplay also allows running the external tools that are only triggered during the execution of the game and retrieving information relevant to its development without having to play the game manually. An example is the execution of profiling tools that measure the game performance by tracing its execution. They allow quantifying the load of the CPU, memory, detecting bugs, and obtaining the frame rate of the game, among others. Therefore, the reports I propose as outputs of the methodology do not necessarily need to be generated by the game or created from scratch. They could also include the information resulting from external tools triggered during the automated gameplay of the agents, as presented in Fig. 9.1. Future work should facilitate the processing and analysis of the results derived from the agents' play-through and look into the generation of the reports proposed.

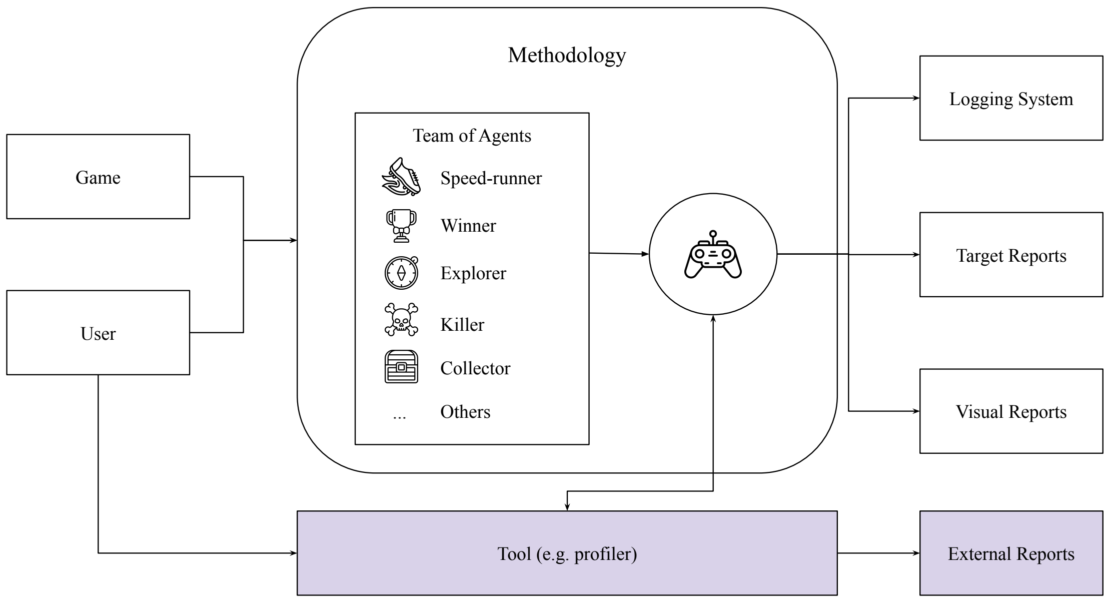


Figure 9.1: . Methodology extension. In addition to the inputs and outputs presented in Chapter 5, automated gameplay could trigger external tools like profilers that assist in the development of the game. These tools create their own reports.

Not every GVGP approach makes use of a forward model. However, the *heuristic diversification* and solution defined and presented for the methodology defined in this thesis do, as they refer to search algorithms. If a forward model is not available in the framework used to develop the game, it would be required to define and implement new GVGP agents that actually fit into the characteristics of the system. Therefore, integrating the methodology proposed into a system without a forward model would require further investigation. First, it would be needed to define and provide a new diversification of behaviours adapted to its characteristics. Once the *heuristic diversification* is defined for the new scope, a similar approach followed for the generation of the agents using the MAP-Elites could be followed. Therefore, the idea presented and developed in this thesis for search algorithms could be translated to Reinforcement Learning by following similar steps. The work mentioned in the previous section carried out by de Woillemont et al. [2021] can be considered the first step, as their approach resembles the *heuristic diversification* in search controllers but applied to Reinforcement Learning algorithms. I believe that implementing the line of research defined in this thesis to fit different areas and GVGP agents would allow adopting the approach in a more diverse range of games and frameworks.

Lastly, a game engine may be focused on the development of a unique game. The approach could also be specialised to use non-general heuristics and accommodate a specific game. In this case, the diversification of heuristics would not apply to GVGP agents, and the solution would only work on this particular game. However, it would still allow executing automated gameplay of different levels and triggering external tools automatically.

In summary, the approach defined to generate the *team* of agents has been implemented with a search algorithm in a general environment. I believe it could be easily extended to use multi-dimensional feature spaces. It could also be applied to more complex games (3D), to other types of games not based on avatars (e.g. strategy games), and to specific ones, with heuristics well-tuned to them. Future work could transfer the idea to other frameworks and solutions, like Reinforcement Learning. It could even look further than the manual design of levels of a game to integrate the approach with Procedural Content Generation (PCG) techniques to assist in their automated generation.

9.1.3 Integrating GVGP agents in games

The exploratory work presented in Chapter 8 is merely a first step looking at the potential use of GVGP agents as an alternative AI technique in games. It analyses the GVGP agents from a Player Experience (PX) point of view and studies the impact their behaviour have on the players. The objective is to ultimately make the integration of GVGP agents within the game possible. However, it is still required further investigation and case studies to see a successful collaboration between these very differentiated areas.

First, focusing on possible extensions of our case study, it could include a third version of the game featuring an NPC with a hand-crafted heuristic. The agent could be programmed to collect the skulls and taking them to the tomb while interfering with the player's map control. This new version could serve as a new baseline about the experience of the players in the game and as a point of comparison to the results found when general agents and heuristics are used instead. However, the limitation of the GVGAI Framework in terms of not being designed as a game development tool with end-users in mind would still be true when merely making this extension to the existing work. An alternative solution would be conducting a similar case study in a framework or game engine directed to the creation of games instead of research. This new study could include the additional version of the game as well. The simplicity of the rules of *Skulls and Tombstones* should allow easy replication of the game in a new framework that, ideally, would provide a forward model so the controllers could also be ported. Another option would be creating new GVGP agents based on similar behaviours used in our experiment but built from scratch, so they are suitable for the new setting. A possible solution would be implementing the game in *Unity*, which is used to develop 2D and 3D games and supports *ML-agents* [Juliani et al., 2018]. This toolkit allows to train agents with reinforcement learning techniques and integrate them into the Unity environment.

We used a quantitative method to evaluate particular Player Experience (PX) constructs related to the game we created. When applying this kind of study, the constructs should be updated to fit the expectations and hypothesis of the game under evaluation. Similarly, the objectives of the general heuristics could change based on the characteristics of the game and the expected behaviours of the agents. The list included in Section 5.3 can serve as inspiration for alternative heuristics to provide to the NPCs. Given the initial results obtained in the case study, it is suggested to use other research methods in addition to quantitative analysis to have a more diverse investigation into the relationship between the behaviour of the agents and the experience of the players. Qualitative methods could provide in-depth analysis and eye tracking could be used when the interest falls back in real-time reactions of the players. Similar AI could also be tested in different games to demonstrate if the PX patterns carry across them.

In general, research looking at integrating GVGP agents in the game as an alternative to existent AI techniques is non-existent. I believe this integration should be possible by studying the effects the agents have on the player's experience. The methods used in game development related to Player Experience are very diverse and depend on the study's objectives and expected outcomes. As a result, building an effective relationship between those methods and GVGP agents is expected to take time. However, I believe it is a line of research with grand potential.

9.2 Concluding Remarks

The research presented in this thesis is the first step in a quite complex area. There is still much open work so that the long-term vision of having General Video Game Playing agents integrated into games and their development process can be a reality. However, I am hopeful. The research in AI has evolved and improved quickly in the past years. What a few years back was simply an idea: machines being capable of playing games like *Chess*; it is now the reality. There are still a lot of exciting challenges emerging every day, and a very active research community is coming up with compelling solutions. I hope this research community sees the potential line of research proposed by my work and gets inspired to investigate further the possibilities brought by merely extending the goals of the general agents beyond winning. I believe that following this path, in the following years, having GVGP agents as part of the tools available for game developers can also be normalised and not a mere dream.

Part IV

Appendices

Appendix A

GVGAI Framework Games: Details and Screenshots

This appendix contains details about the games from the GVGAI Framework used in the experiments without modifications, listed in Section 3.4.1. These details include the player's movement, rules, elements, and a screenshot of the level used.

A.1 *Aliens*

The game (Fig. A.1) is formed by the avatar, which is a **ship** located at the bottom of the screen that can move right or left and shoot **missiles**; **aliens**, NPCs that spawn from the top and drop **bombs**; and immovable **meteorites** that disappear when hit by a missile or a bomb. The objective is to kill all the aliens before they reach the bottom of the screen. Hitting a meteorite with a missile increases the score by 1, killing an alien by 2, and being killed by a bomb ends the game and decreases the score by 1.

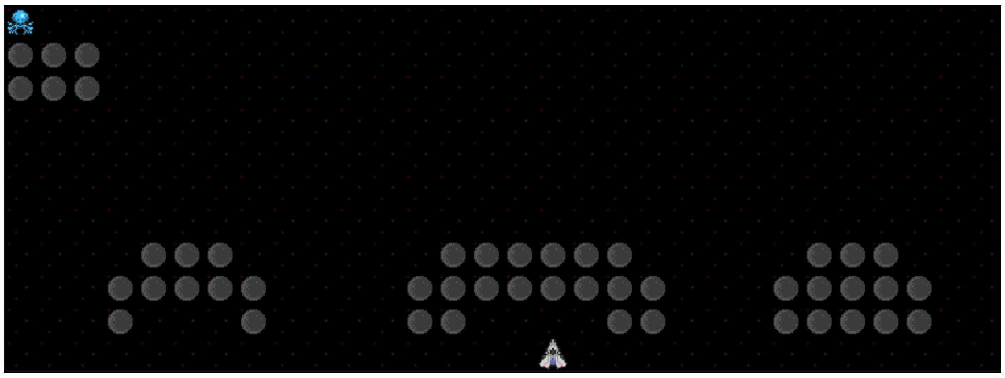


Figure A.1: Screenshot of *Aliens* at $t = 0$: 1 initial alien, 47 meteorites, 30 locations.

A.2 *Bait*

It is a puzzle game, and its goal is to reach the **door** having collected the **key** first. The avatar can move up, down, right, and left; and push the **boxes** scattered around the map. The map is shaped by **walls**. Reaching the goal with the key increases the score by 5. Some levels have **holes** and **mushrooms**, but the level used in the experiments (Fig. A.2) does not include any of these.

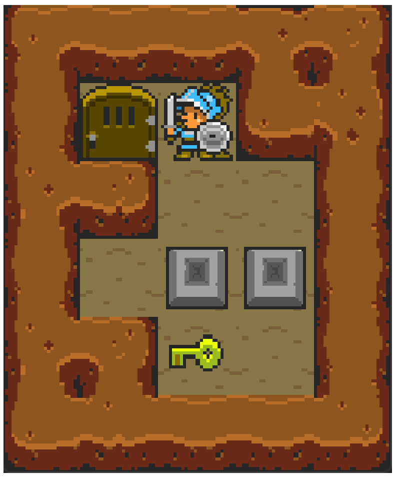


Figure A.2: Screenshot of *Bait* at $t = 0$: 1 door, 1 key, 2 boxes, 21 walls, 9 locations.

A.3 *Butterflies*

The game (Fig. A.3) is formed by an **avatar** that can move up, down, right, and left; **butterflies**, that are random NPCs; **cocoons** that are immutable elements the player can't interact with; and **trees** shaping the limits of the map. The goal is to capture all butterflies before the time runs out or the cocoons have disappeared. Butterflies move randomly, and when they collide with the cocoons scattered around the map, these cocoons vanish and become new butterflies. The player captures the butterflies by colliding with them, an action that increases the score by 2.

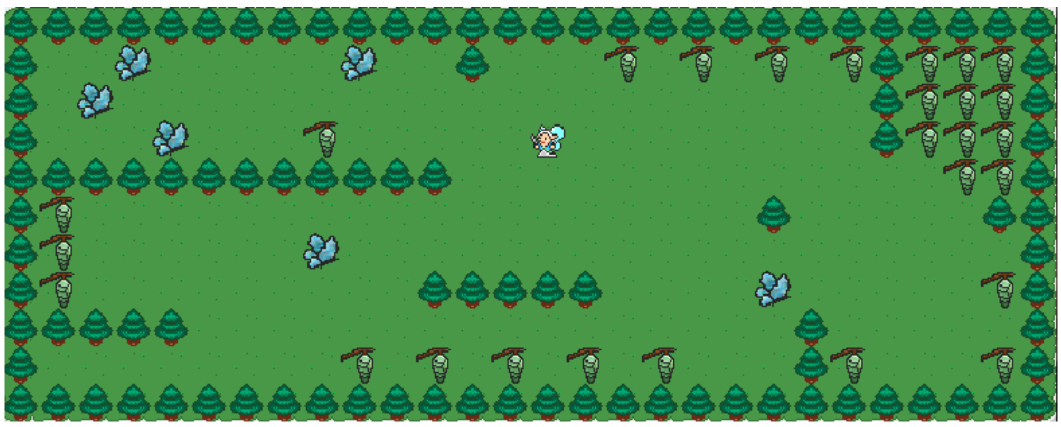


Figure A.3: Screenshot of *Butterflies* at $t = 0$: 6 butterflies, 27 cocoons, 102 trees, 206 locations.

A.4 *Camel Race*

The game (Fig. A.4) consists of a race, so the objective is to be the first one to reach the goal. The avatar and NPCs are a **camel** and can move up, down, right, and left. The score is increased or decreased by 1 when a camel reaches a **goal**. If the player is the winner of the race, the score is increased. The map is limited by **walls**.

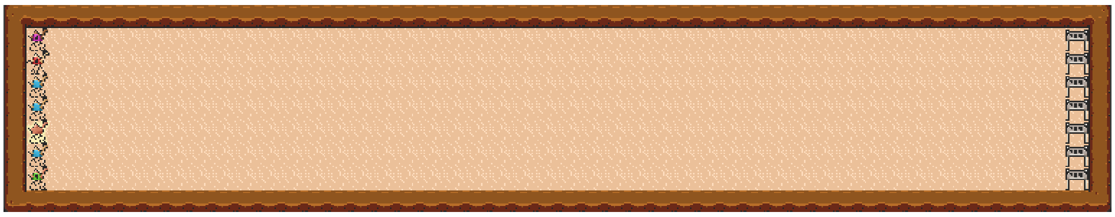


Figure A.4: Screenshot of *Camel Race* at $t = 0$: 6 camel opponents, 7 goals, 110 walls, 322 locations.

A.5 *Chase*

The game (Fig. A.5) is formed by an **avatar** that can move up, down, right, and left; **white birds**, that are NPCs that run away from the player; and **trees** that outline the map. The objective is to kill all the birds by colliding with them. When the player collides with a white bird, the score is increased by 1, and the bird is transformed into its **carcass**. When a white bird walks into a carcass, it becomes a **black bird**, which is an NPC that chases the player. If it collides with the avatar, the game ends, and the score decreases by 1.

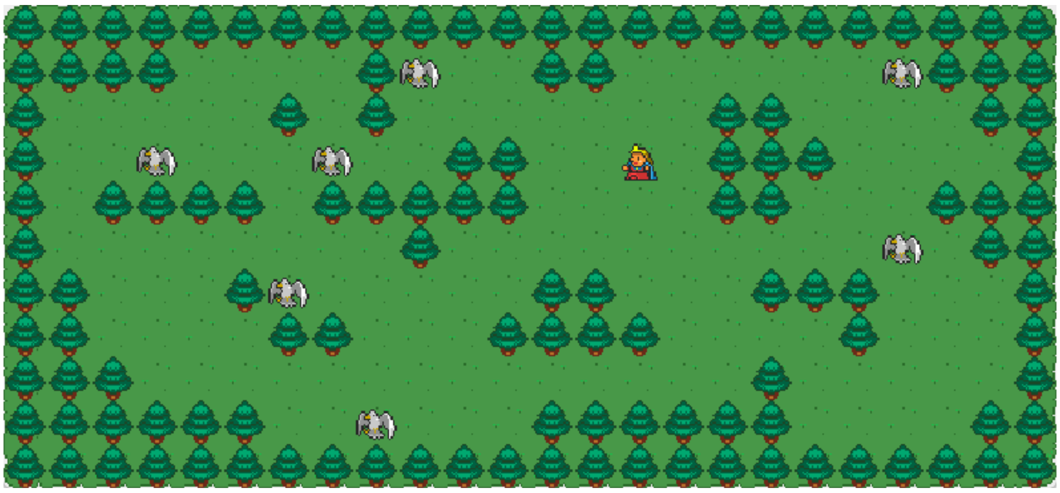


Figure A.5: Screenshot of *Chase* at $t = 0$: 7 white birds, 129 trees, 135 locations.

A.6 *Chopper*

The avatar is a **helicopter** that can move up, down, right, and left in a designated area of the map and shoots **bombs** to destroy **tanks** located in the ground. The tanks are NPCs that shoot missiles to destroy the **satellites** floating in space. The objective is to destroy the tanks without being hit by the missiles before the time runs out or all the satellites are destroyed. The **clouds** serve as a protective barrier for the satellites and can also be destroyed by the tanks. **Supplies** are resources that need to be collected to be able to shoot, increasing the score by 5. There are two spawners: **ammo portal**, for supplies, and **base**, for tanks. Destroying tanks increases the score by 1; losing satellites or being killed decreases the score by 1. Fig. A.6 shows a screenshot of the game.

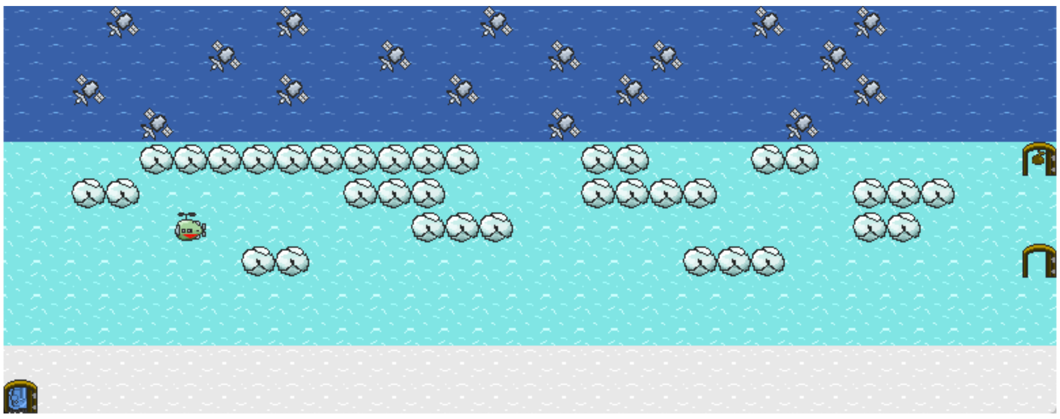


Figure A.6: Screenshot of *Chopper* at $t = 0$: 18 satellites, 1 initial tank, 1 initial supply, 36 clouds, 2 ammo portals, 1 base, 184 locations.

A.7 *Crossfire*

The game (Fig. A.7) is formed by an **avatar** that can move up, down, right, and left; **turrets**, that are NPCs that shoot **bombs** randomly; a **door**; and **walls** outlining the map. The objective is to reach the door without being hit by the bombs, which kills the player. Reaching the goal increases the score by 5, and being killed decreases it by 1.

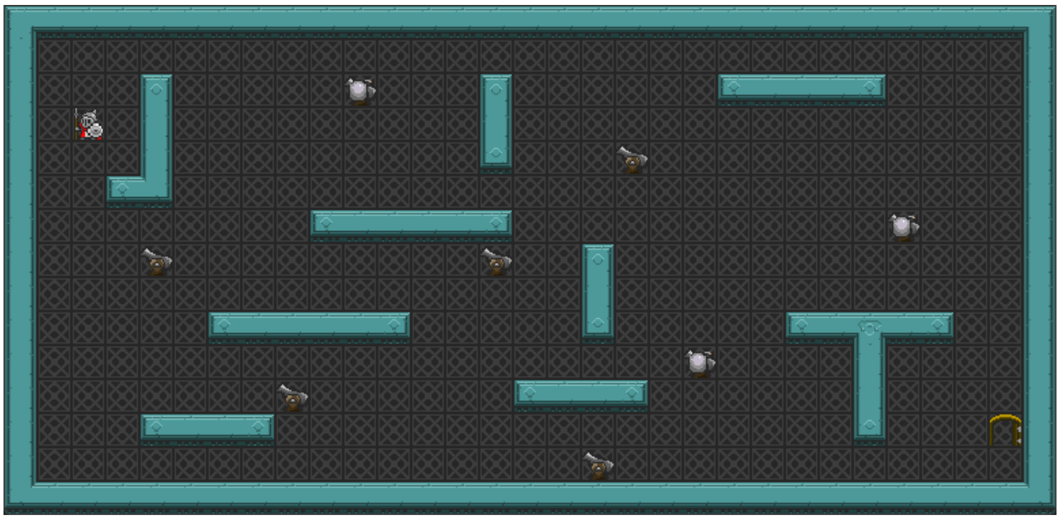


Figure A.7: Screenshot of *Crossfire* at $t = 0$: 8 turrets, 1 door, 3 initial bombs, 132 walls, 333 locations.

A.8 *Digdug*

The game (Fig. A.8) is formed by an **avatar** that can move up, down, right, and left and use a **shovel**; **monsters** that move randomly; **gems**; **gold**; and **breakable walls**. The goal is to kill all the monsters and collect all the gems and gold pieces. The player can use the shovel to break walls, collect gold or shoot a **boulder**. Gems are collected by colliding with them, an action that increases the score by 1. The gold disappears when hit with the shovel, which does not increase the score but creates a **falling rock** that can kill the player and monsters. Monsters are generated every 200 game ticks from monster **spawners** and are killed when hit with a boulder, which increases the score by 2. The player also dies if colliding with a monster, which decreases the score by 1.

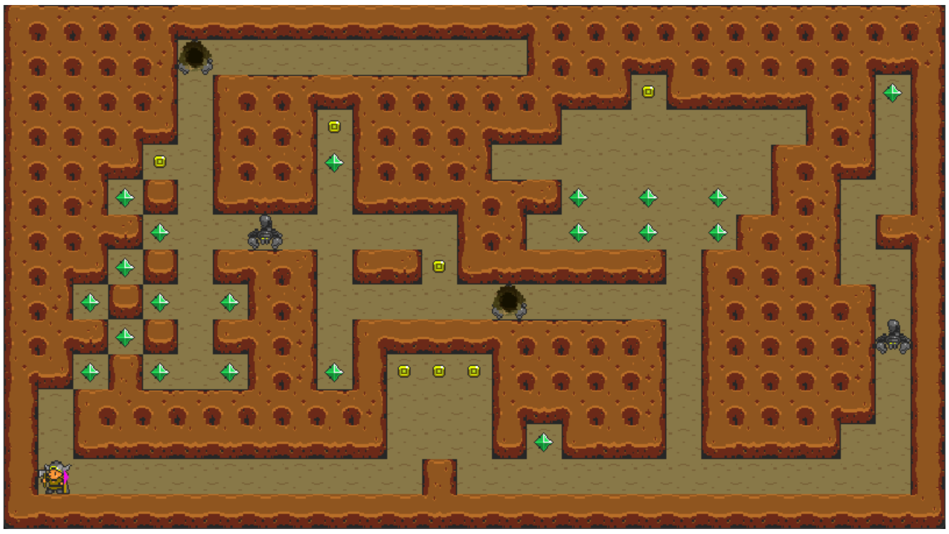


Figure A.8: Screenshot of *Digdug* at $t = 0$: 20 gems, 7 gold pieces, 2 initial monsters, 2 monster spawners, 267 breakable walls, 405 locations.

A.9 *Escape*

The game (Fig. A.9) is formed by an **avatar**, which is a *mouse* that can move up, down, right, and left; **holes** in the ground; **boxes** and **cheese**. The objective is to reach the cheese by pushing away the boxes that are blocking the path. These can be destroyed by pushing them into the holes. The player dies if they step on the hole, which also decreases the score by 1. When the player reaches the cheese, the score increases by 1. The timeout for this game is 1000 game ticks.

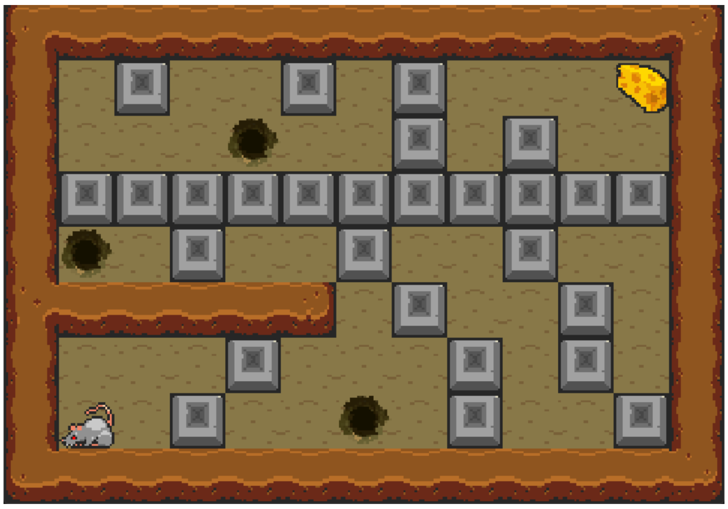


Figure A.9: Screenshot of *Escape* at $t = 0$: 1 cheese, 3 holes, 27 boxes, 45 walls, 74 locations.

A.10 *Hungry Birds*

The game (Fig. A.10) is formed by an **avatar**, which is a *bird* that can move up, down, right, and left; **worms**; a **goal sign**; and **trees** that shape the map. Every game tick, the health of the player decreases. The aim is to reach the goal to *exit* the maze before the health points are completely depleted, which kills the player. Colliding with the worms increases health points, and reaching the goal increases the score by 100.

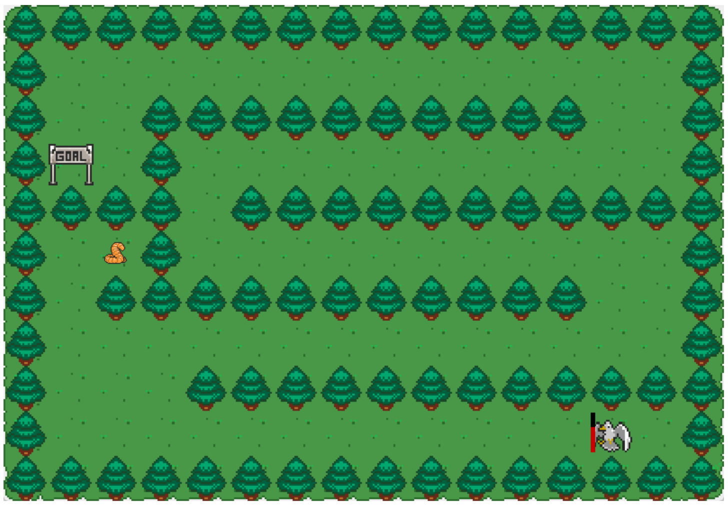


Figure A.10: Screenshot of *Hungry Birds* at $t = 0$: 1 worm, 1 goal, 97 trees, 79 locations.

A.11 *Infection*

The game (Fig. A.11) is formed by an **avatar** that can move up, down, right, and left and use a **sword**; 3 types of random NPCs: **doctors**, **healthy people** and **infected people**; **virus containers** that are immovable objects; **entrances** that spawn doctors every 100 game ticks; and **walls** that outline the map. The goal is to infect all healthy people before the time runs out. The player becomes a carrier of the virus by colliding with its containers, infecting healthy people when entering in contact with them (+2 score change). The doctors can cure the player and the infected people (-1 score change) but can be killed with the sword, which increases the score by 2.

A.12 *Intersection*

The game (Fig. A.12) is formed by an **avatar** that can move up, down, right, and left; **cars** that move horizontally or vertically based on the predetermined road direction; a **goal sign**; and **trees** blocking the path. The objective of the game is to reach the goal as many times as possible before the time runs out. When the goal is reached, it is re-spawned in a different map location and the score increases by 10. If the player is hit by a car, they lose a life, the avatar is transported back to the start point, and the score decreases by 5. Losing 5 lives kills the avatar and ends the game. The timeout is 1000 game ticks.

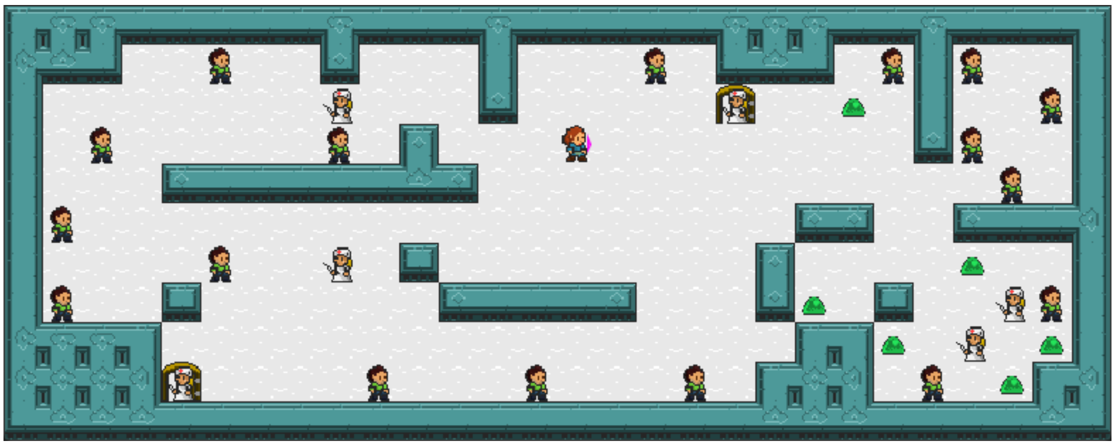


Figure A.11: Screenshot of *Infection* at $t = 0$: 17 healthy people, 6 virus containers, 4 initial doctors, 2 doctor entrances, 121 walls, 187 locations.

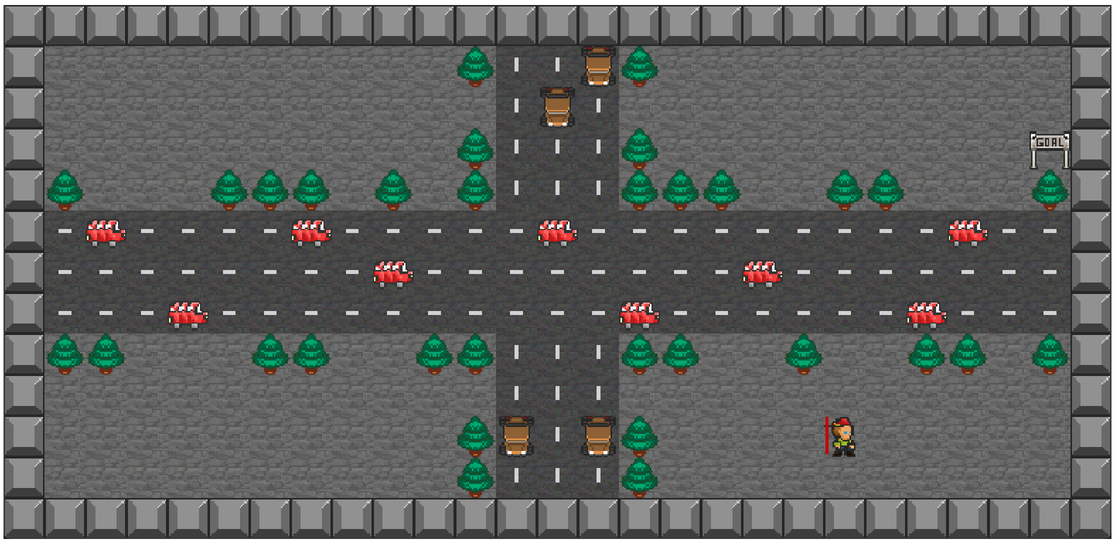


Figure A.12: Screenshot of *Intersection* at $t = 0$: 1 goal, 13 initial cars, 32 trees, 76 walls, 243 locations.

A.13 *Lemmings*

The game (Fig. A.13) is formed by an **avatar** that can move up, down, right, and left and use a **shovel**; **lemmings** that are NPCs moving towards a **door**; **holes** in the ground; and **breakable walls**. Lemmings are spawned into the level through the **entrances**. The goal is to help the lemmings to reach the door by breaking the walls with the shovel. Lemmings die if they fall into a hole (-2 score points), and breaking a wall decreases the score by 1. The player can also die by falling into a hole, which decreases the score by 5 and ends the game. For each lemming able to reach the door, the score is increased by 2.

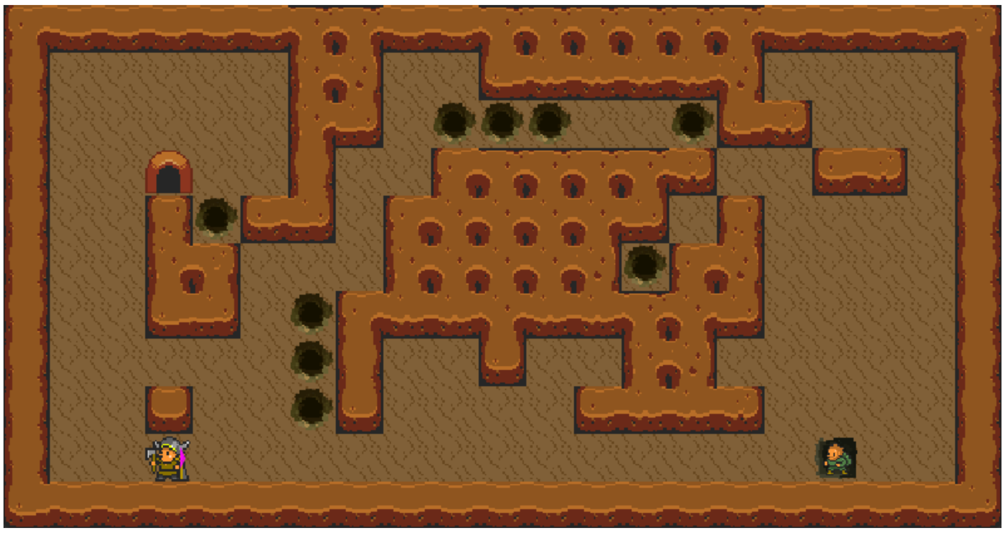


Figure A.13: Screenshot of *Lemmings* at $t = 0$: 1 door, 1 initial lemming, 1 entrance, 9 holes, 121 breakable walls, 222 locations.

A.14 *Missile Command*

The game (Fig. A.14) is formed by an **avatar**, which is a *spaceship* that can move up, down, right, and left and shoot **explosives**; **fire missiles** that are NPCs; and **cities**. The fire missiles are directed to the cities, which are destroyed when colliding with them, decreasing the score by 1. The goal is to eliminate the missiles by hitting them with explosives (+2 score change) before the cities disappear. The map is limited by **walls**.

A.15 *Modality*

The game (Fig. A.15) is formed by an **avatar** that can move up, down, right, and left; a **hole**; a **bush**; and **two types of terrain**. The goal is to push the bush into the hole. The player can walk over the two types of surfaces, but they can only cross from one to the other by a **specific point** in the map. There are **walls** shaping the level. The bush can be pushed through the different surfaces freely.

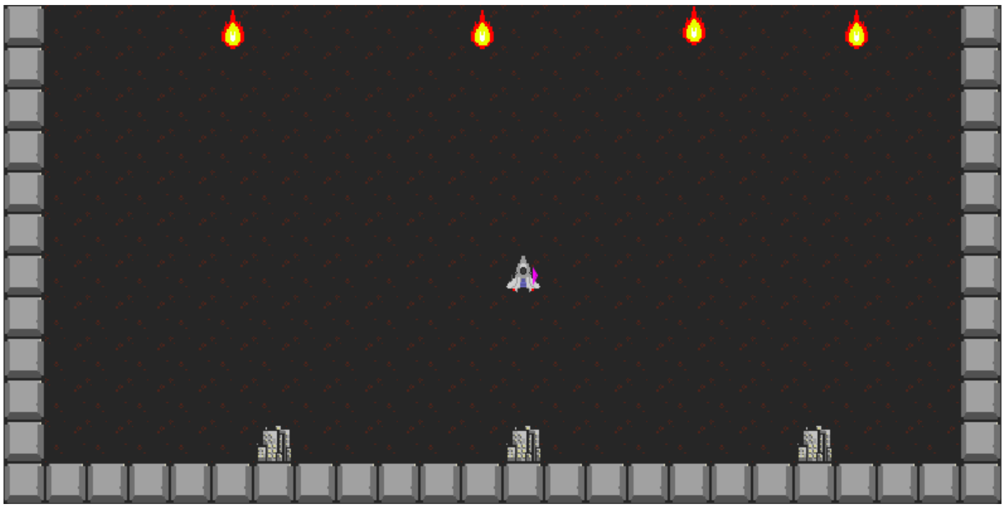


Figure A.14: Screenshot of *Missile Command* at $t = 0$: 3 cities, 4 fire missiles, 46 walls, 242 locations.

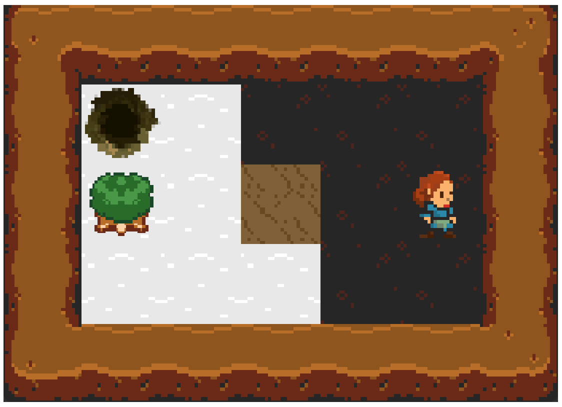


Figure A.15: Screenshot of *Modality* at $t = 0$: 1 hole, 1 bush, 7 tiles of surface type 1, 7 tiles of surface type 2, 1 crossing point, 20 walls, 15 locations.

A.16 *Plaque Attack*

The game (Fig. A.16) is formed by an **avatar** that can move up, down, right, and left and shoot **blue toothpaste**; **teeth** that can be *clean* or *damaged*; two types of food: **hamburgers** and **hot-dogs**; **trolleys** that spawn the food; and **walls** that shape the map. The goal is to destroy all the food by shooting at it (+2 score change) before they damage the teeth. A tooth is damaged when colliding with the food (−3 score change) but the player can clean it back by colliding with it, increasing the score by 1.

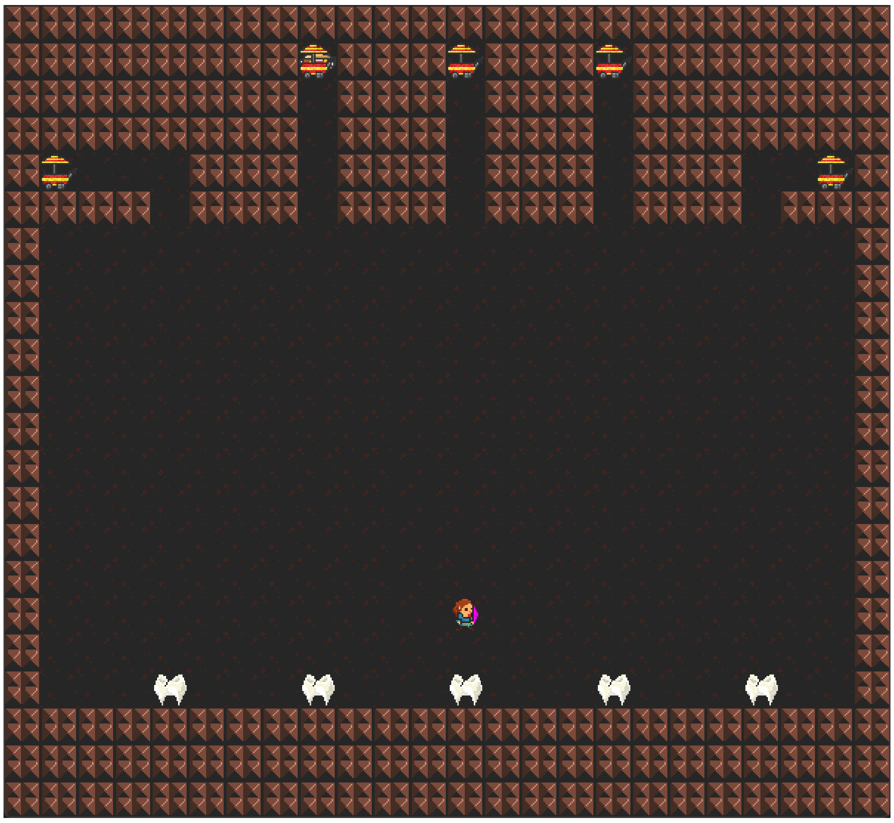


Figure A.16: Screenshot of *Plaque Attack* at $t = 0$: 5 teeth (initially clean), 5 trolleys, 1 initial hamburger, 218 walls, 310 locations.

A.17 *Roguelike*

The game (Fig. A.17) is formed by an **avatar** that can move up, down, right, and left and use a **sword**; two types of enemies: **spiders** that are random NPCs and **ghosts** that chase the player; **hearts**; **gold pieces**; **weapons**; **cities**; **keys**; **locked doors** and a **goal sign**. The map is shaped by **walls**. The objective of the game is to reach the goal, which increases the score by 10. The locked doors block the path of the player, but they can be opened by collecting the keys scattered around the map. The player has a certain number of health points that get reduced every time they collide with an enemy. If they lose all the health points, the game is over. It is possible to increase the number of health points by collecting hearts or exchanging gold in the city. It is required to collect weapons (+2 score change) to use the sword to kill enemies. Killing a spider increases the score by 2 and a ghost by 1. Collecting a key or gold increases the score by 1.

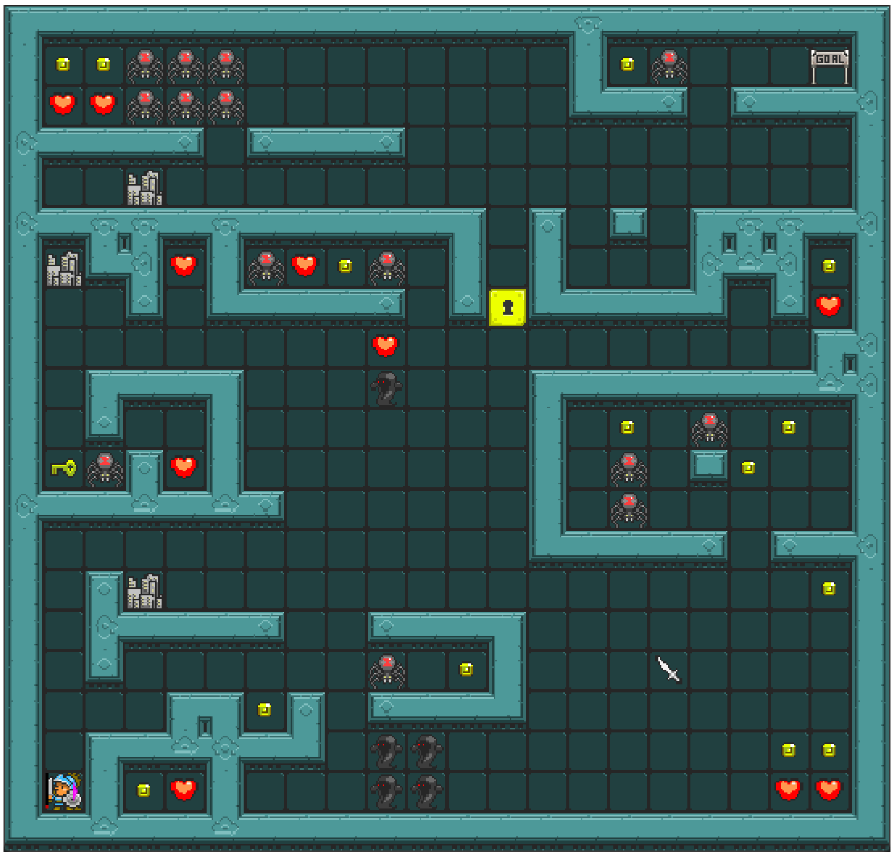


Figure A.17: Screenshot of *Roguelike* at $t = 0$: 1 goal, 1 locked door, 1 key, 3 cities, 14 pieces of gold, 10 hearts, 1 weapon, 14 spiders, 5 ghosts, 196 walls, 266 locations.

A.18 *Seaquest*

The game (Fig. A.18) is formed by and **avatar** that is a *submarine* that can move up, down, right, and left and shoot **torpedos**; three types of enemies: **whales**, **sharks** and **piranhas**; **divers** that are random NPCs; **whirlpools** that spawn the NPCs; and two types of environments: **ocean** and **surface**. The goal is to rescue divers and stay alive until the time runs out. The divers are rescued by colliding with them and bringing them to the surface, which increases the score by 1000. The player can carry only a maximum of 4 divers at the same time. The submarine has a limited capacity of oxygen to stay in the ocean, so it needs to refill it by going to the surface. If the oxygen level falls to 0 or the player collides with an enemy, the game ends. Enemies can be killed with a torpedo, increasing the score by 1. The timeout is 1000 game ticks.

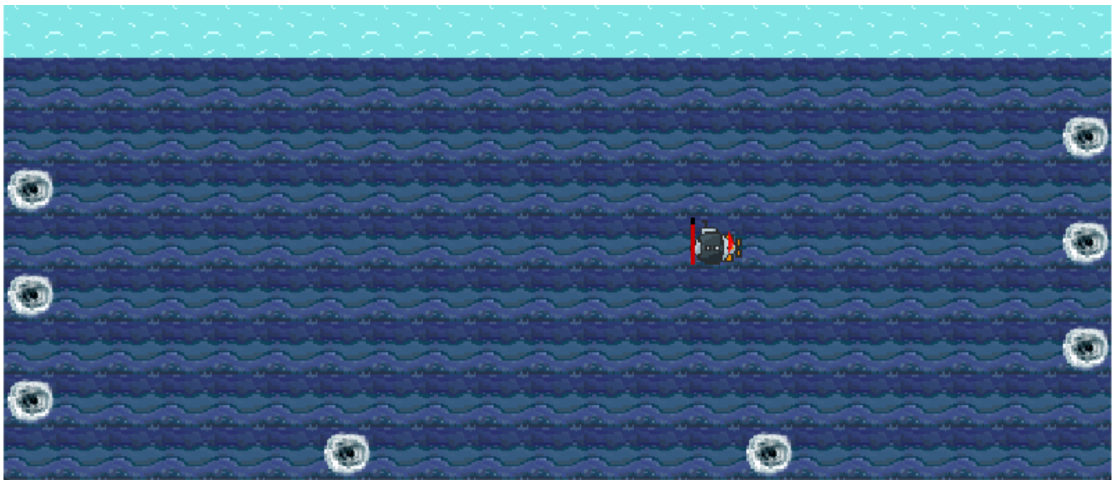


Figure A.18: Screenshot of *Seaquest* at $t = 0$: 22 tiles of surface, 168 tiles of ocean, 8 whirlpools, 189 locations.

A.19 *Survive Zombies*

The game (Fig. A.19) is formed by an **avatar** that can move up, down, right, and left; two type of NPCs: **zombies** that chase the player and **priests** that move randomly; two NPCs spawners: **tombs** for zombies and **cloaks** for priests; **hearts** and **walls** that shape the map. The goal is to survive until the time runs out. The player has some health points that lower every time they collide with a zombie, which also decreases the score by 1. Zombies disappear after hurting the player or by colliding with the priests, who transform them into hearts. The player can recover health points by collecting hearts (+1 score change), but they should avoid the tombs as colliding with them decreases the score by 1. The timeout for this game is 1000 game ticks.

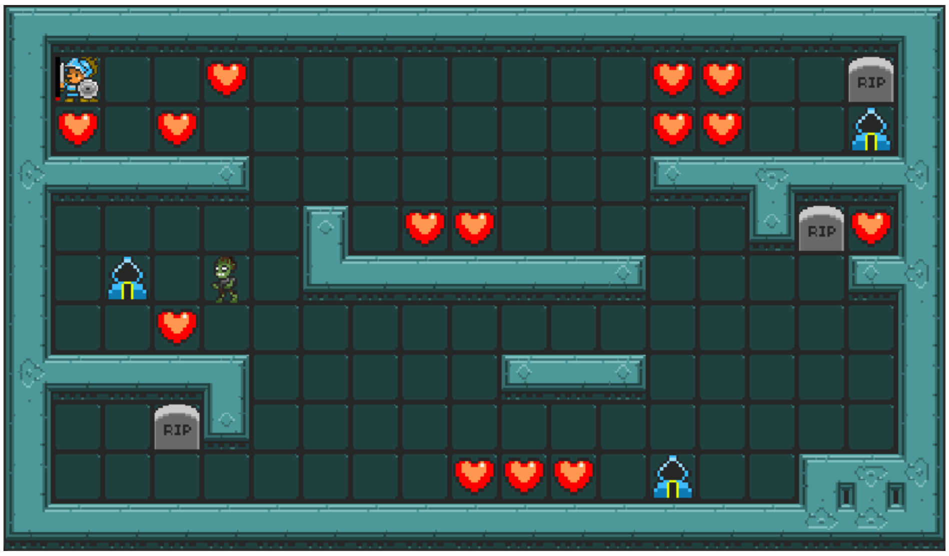


Figure A.19: Screenshot of *Survive Zombies* at $t = 0$: 3 tombs, 3 cloaks, 14 initial hearts, 1 initial zombie, 85 walls, 121 locations.

A.20 *Wait For Breakfast*

The game (Fig. A.20) is formed by an **avatar** that can move up, down, right, and left; **chairs**, **tables**; a **waiter**; a **kitchen door** and an **exit**. It is not possible to collide or interact with the chairs or tables. The goal is to go to the empty table and wait for the waiter to serve the food. If the player leaves, the game is lost. The timeout for this game is 1000 game ticks.

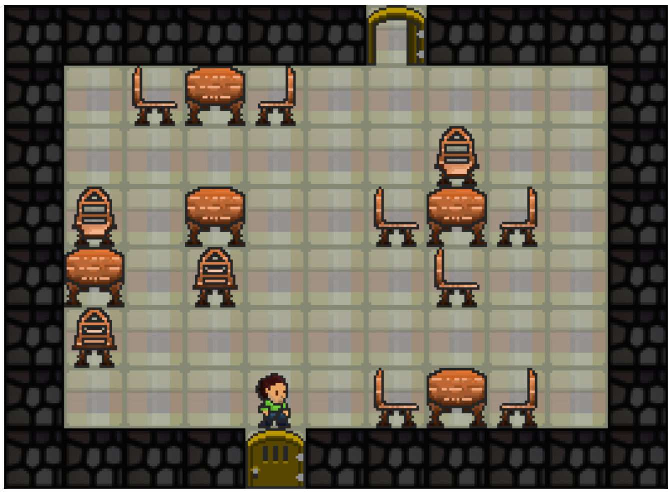


Figure A.20: Screenshot of *Wait for Breakfast* at $t = 0$: 11 chairs, 5 tables, 1 kitchen door, 1 exit door, 50 locations.

Appendix B

List of Research Resources

This appendix gathers a list of the resources related to each of the experiments executed, including links to the repositories containing the code and results, demos and references to the relevant appendix.

B.1 Heuristic Diversification

Resources for the *heuristic diversification* experiments described in Chapter 4:

- [Guerrero Romero, 2017]. Github repository that contains the code, raw results, and the scripts that generate the tables and graphs.
- Appendix C gathers the tables and graphs displaying the final results per game for each of the heuristics.
- [Guerrero Romero, 2018]. Demo showing an example of gameplay of *Butterflies* when an RS agent is provided with each of the heuristics.

B.2 Generation of the *Team* of Agents

Resources for the experiments described in Chapter 6:

- [Guerrero Romero, 2021a, main branch]. Github repository that contains the code corresponding to the agents, heuristics, and MAP-Elites implementation.
- [Guerrero Romero, 2021b]. Github repository that contains the scripts used to process the results of the experiments.
- [Guerrero Romero, 2021d]. OSF repository that contains the *jar* executable and configuration files used for each game and set of experiments: B2, B3, Z5, D5 and S4.
- Appendix D gathers all the graphs that represent the resulting MAP-Elites for each game and set of experiments: B2, B3, Z5, D5 and S4.

- [Guerrero Romero, 2021e]. Interactive demo that allows going through the resulting maps and selecting the agents generated for each to inspect details about them and their gameplay. It is also possible to download a standalone and the files required to run them locally.

B.3 *Team* Portability and Level Testing

Resources for the portability and level testing experiments included in Chapter 7:

- [Guerrero Romero, 2021a, secondary branch]. Github repository that contains the code of the executable created to run the portability and level testing experiments and the definition of the maps used.

- [Guerrero Romero, 2021b]. Github repository that contains the script used to generate the graphs included in the chapter.

- [Guerrero Romero, 2021c]. OSF repository that contains the config file, executables, and results.

- Appendix E gathers the tables containing the resulting portability stats for each agent and level. For each game, two types of tables are provided: one containing the resulting stats of each *behaviour-type* agent across all levels and another including the collective resulting stats of every agent in each level. The tables with the resulting stats for the exploratory level testing experiments fit in the corresponding Section 7.4, so they are not included in the appendix.

B.4 Case study

Resources for the exploratory case study described in Chapter 8.

- [Guerrero Romero and Kumari, 2020]. OSF repository that contains the standalone *jar* executable of each version of the game and their VGDL source code.

- Appendix F gathers the Information Sheet and Consent Form provided to the participants; as well as the list of questions used, taken from the IMI and PENS questionnaires. PENS is not open source, so I merely mention the number of the question used, while in IMI I include the questions used and their adaptation to our questionnaire.

Appendix C

Full Results: Heuristic Diversification in GVGAI

This appendix contains the full results of the *heuristic diversification* experiments described in Chapter 4. The tables and graphs included are organised by heuristics, as follows:

- WMH results per game (C.1)
- EMH results per game (C.2)
- KDH results per game (C.3)
- KEH results per game (C.4)
- KEH individual resulting predictions per agent and game (C.5)

The details and nomenclature of the data displayed in the tables and graphs are provided in each corresponding section.

C.1 Per Game Results: WMH

Each table contains the WMH results of one of the games displaying the information listed below. The data is the final average resulting from the 20 gameplays of each agent. The values in parenthesis in the table represent the corresponding *Std. Deviation.*

- Total number of points received in the game (Points)
- Agent (Controller)
- Rate of wins (% Wins)
- Average of score achieved (Score)
- Average of End of Game (EoG) ticks for winning play-throughs (EoG victories)
- Average of End of Game (EoG) ticks for losing play-throughs (EoG defeats)

For each game, it is also included the resulting graphs for the stats related to the heuristic: victories, score, EoG ticks on victories, and EoG ticks on defeats. They show the resulting stats of the 20 gameplays achieved by each agent, in order: OLETS (blue), OLMCTS (red), OSLA (green), RHEA (purple), and RS (orange).

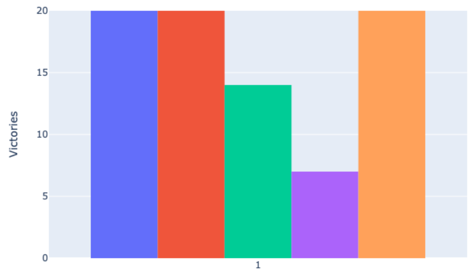
The games are included in alphabetic order.

WMH: *Aliens*					
Points	**Controller**	**% Wins**	**Score**	**EoG victories**	**EoG defeats**
25	**OLETS**	100.00%	77.95 (0.56)	705.05 (12.26)	-
18	**OLMCTS**	100.00%	76.55 (1.30)	489.45 (8.58)	-
15	**RS**	100.00%	76.20 (1.34)	500.90 (13.30)	-
12	**OSLA**	70.00%	63.15 (1.00)	781.71 (31.17)	873.67 (40.86)
10	**RHEA**	35.00%	49.80 (4.04)	737.86 (28.24)	592.38 (82.11)

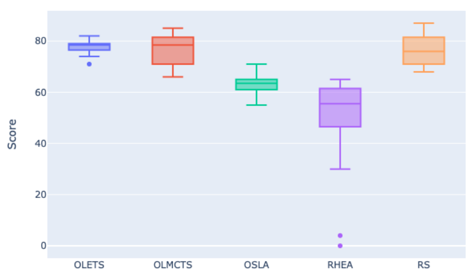
Table C.1

WMH: *Bait*					
Points	**Controller**	**% Wins**	**Score**	**EoG victories**	**EoG defeats**
25	**OLETS**	50.00%	2.50 (0.56)	8.90 (0.09)	1500.00 (0)
18	**OLMCTS**	50.00%	2.50 (0.56)	16.90 (1.43)	1500.00 (0)
15	**RS**	50.00%	2.50 (0.56)	22.60 (2.51)	1500.00 (0)
12	**OSLA**	30.00%	1.50 (0.51)	37.33 (5.78)	1500.00 (0)
10	**RHEA**	30.00%	1.50 (0.51)	90.67 (17.13)	1500.00 (0)

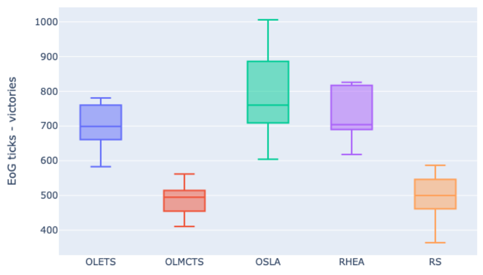
Table C.2



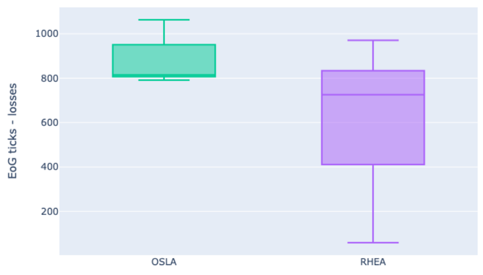
(a) *Victories*



(b) *Score*

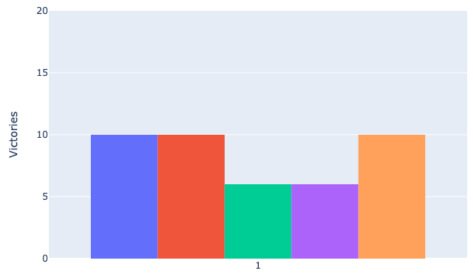


(c) *EoG ticks - victories*

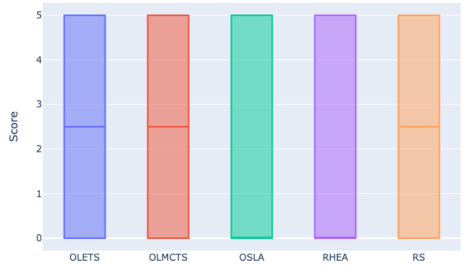


(d) *EoG ticks - defeats*

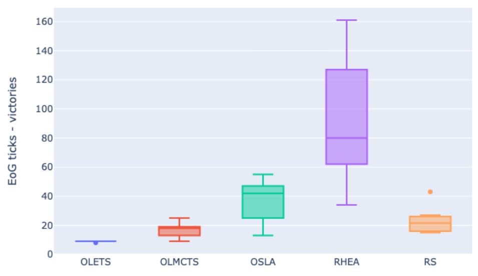
Figure C.1: WMH *Aliens* graphs



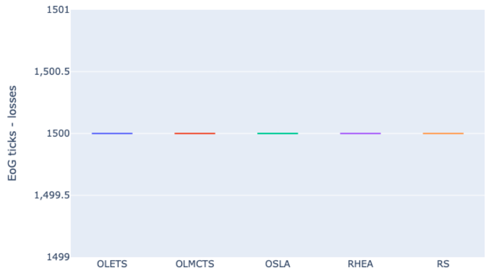
(a) *Victories*



(b) *Score*



(c) *EoG ticks - victories*

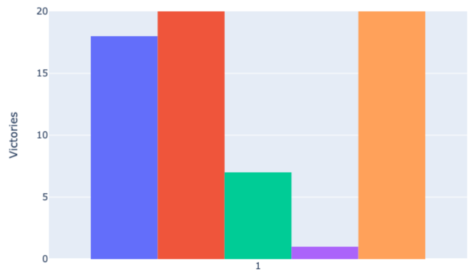


(d) *EoG ticks - defeats*

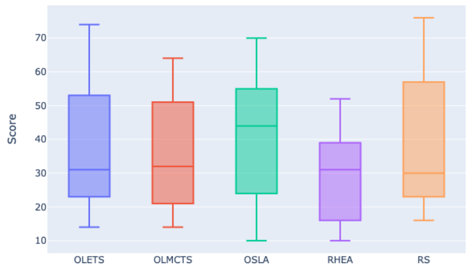
Figure C.2: WMH *Bait* graphs

WMH: *Butterflies*					
Points	**Controller**	**% Wins**	**Score**	**EoG victories**	**EoG defeats**
25	**RS**	100.00%	36.90 (3.98)	299.60 (40.82)	-
18	**OLMCTS**	100.00%	36.60 (3.78)	255.40 (28.23)	-
15	**OLETS**	90.00%	37.10 (3.80)	292.83 (52.88)	210.50 (33.59)
12	**OSLA**	35.00%	40.40 (4.29)	987.57 (102.22)	795.23 (136.05)
10	**RHEA**	5.00%	29.20 (2.86)	849.00 (0)	818.68 (108.01)

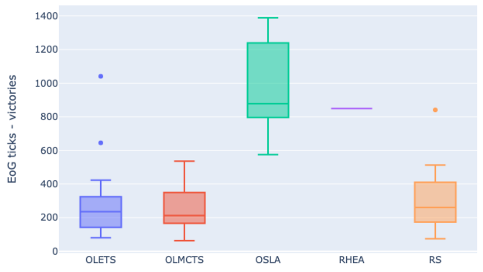
Table C.3



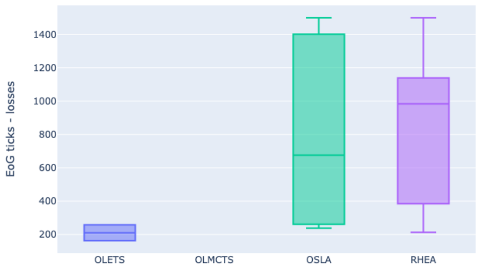
(a) *Victories*



(b) *Score*



(c) *EoG ticks - victories*

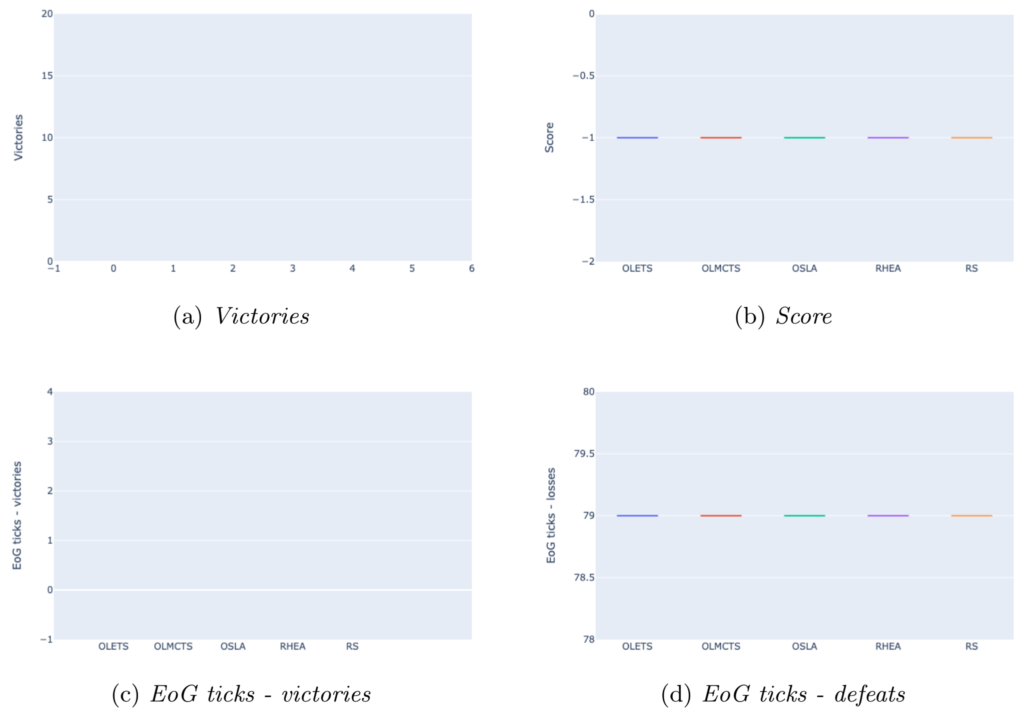


(d) *EoG ticks - defeats*

Figure C.3: WMH *Butterflies* graphs

WMH: *Camel Race*					
Points	**Controller**	**% Wins**	**Score**	**EoG victories**	**EoG defeats**
25	**OLETS**	0.00%	-1.00 (0)	-	79.00 (0)
25	**OLMCTS**	0.00%	-1.00 (0)	-	79.00 (0)
25	**OSLA**	0.00%	-1.00 (0)	-	79.00 (0)
25	**RHEA**	0.00%	-1.00 (0)	-	79.00 (0)
25	**RS**	0.00%	-1.00 (0)	-	79.00 (0)

Table C.4



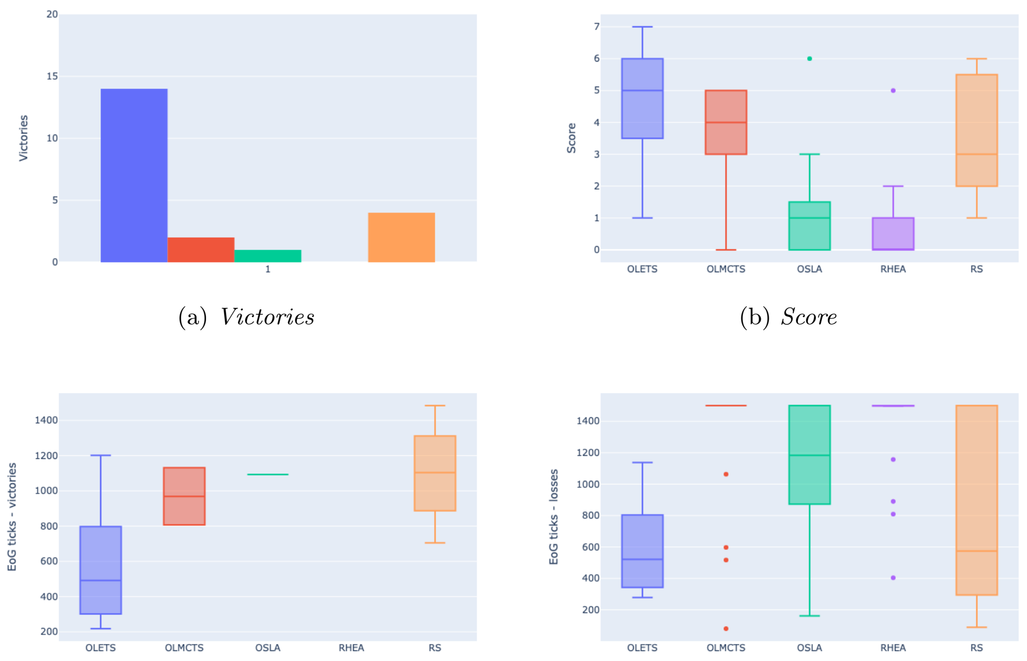
(a) *Victories* (b) *Score*

(c) *EoG ticks - victories* (d) *EoG ticks - defeats*

Figure C.4: WMH *Camel Race* graphs

WMH: *Chase*					
Points	**Controller**	**% Wins**	**Score**	**EoG victories**	**EoG defeats**
25	**OLETS**	70.00%	4.80 (0.36)	564.14 (81.89)	601.00 (122.06)
18	**RS**	20.00%	3.50 (0.39)	1099.25 (138.25)	849.62 (147.50)
15	**OLMCTS**	10.00%	3.50 (0.32)	969.00 (114.55)	1292.06 (99.55)
12	**OSLA**	5.00%	1.20 (0.32)	1093.00 (0)	1132.00 (93.81)
10	**RHEA**	0.00%	0.65 (0.26)	-	1362.80 (66.93)

Table C.5



(a) *Victories* (b) *Score*

(c) *EoG ticks - victories* (d) *EoG ticks - defeats*

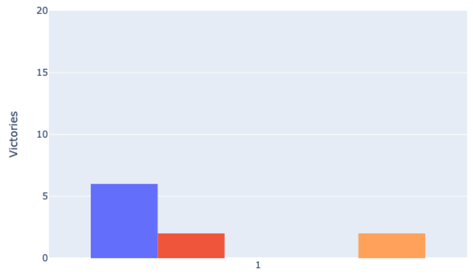
Figure C.5: WMH *Chase* graphs

WMH: *Chopper*					
Points	**Controller**	**% Wins**	**Score**	**EoG victories**	**EoG defeats**
25	**OLETS**	30.00%	6.90 (1.02)	1288.33 (51.33)	1480.21 (19.07)
18	**OLMCTS**	10.00%	2.05 (1.39)	1315.00 (104.65)	1440.39 (23.94)
15	**RS**	10.00%	0.85 (1.34)	1341.00 (30.41)	1343.89 (52.27)
12	**RHEA**	0.00%	-2.15 (0.86)	-	425.50 (52.70)
10	**OSLA**	0.00%	-9.40 (0.94)	-	1119.70 (44.57)

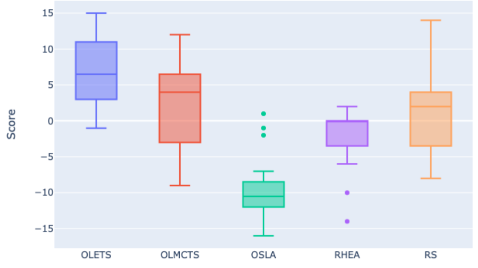
Table C.6

WMH: *Crossfire*					
Points	**Controller**	**% Wins**	**Score**	**EoG victories**	**EoG defeats**
25	**OLETS**	55.00%	2.70 (0.57)	735.27 (103.56)	1465.89 (32.16)
18	**OSLA**	5.00%	-0.70 (0.29)	726.00 (0)	381.00 (66.57)
15	**RS**	5.00%	-0.70 (0.29)	1197.00 (0)	491.84 (96.77)
12	**OLMCTS**	0.0%	0.00 (0)	-	1500.00 (0)
10	**RHEA**	0.00%	-1.00 (0)	-	84.25 (13.59)

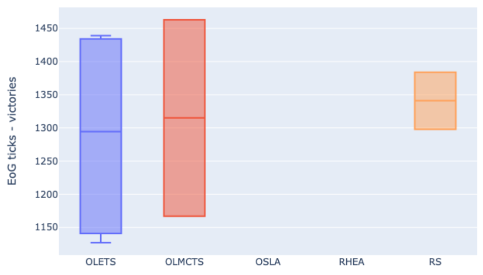
Table C.7



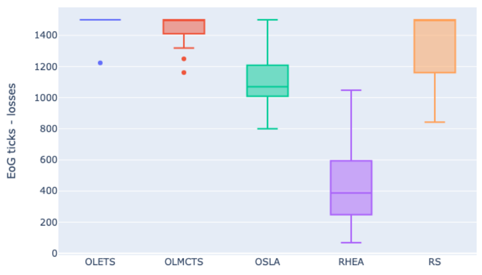
(a) *Victories*



(b) *Score*

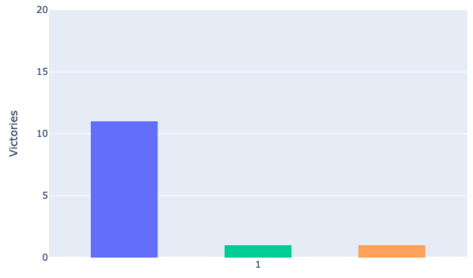


(c) *EoG ticks - victories*

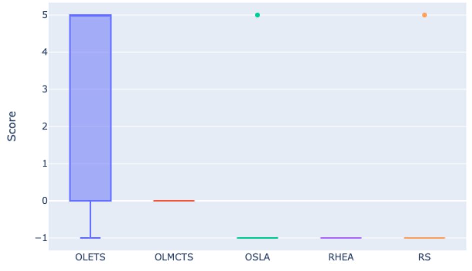


(d) *EoG ticks - defeats*

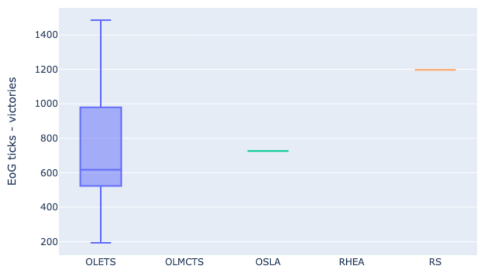
Figure C.6: WMH *Chopper* graphs



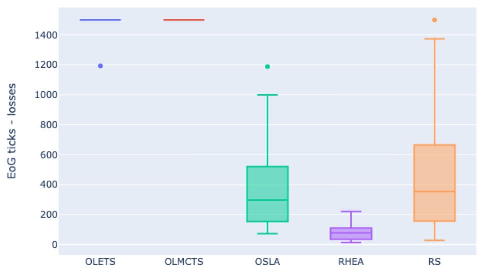
(a) *Victories*



(b) *Score*



(c) *EoG ticks - victories*

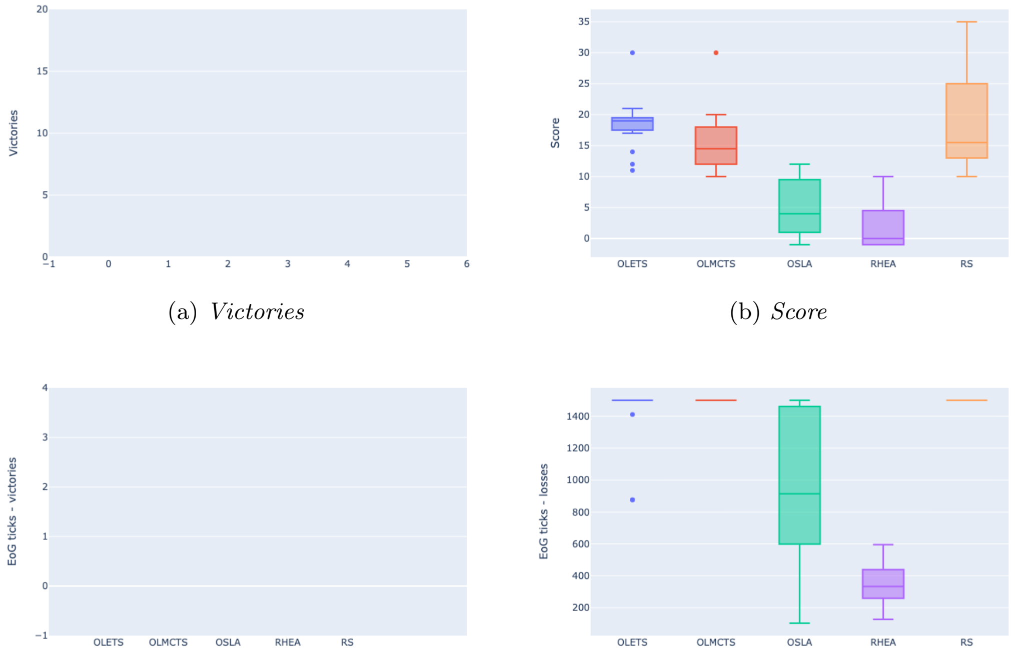


(d) *EoG ticks - defeats*

Figure C.7: WMH *Crossfire* graphs

WMH: *Digdug*					
Points	**Controller**	**% Wins**	**Score**	**EoG victories**	**EoG defeats**
25	**OLETS**	0.00%	18.40 (0.82)	-	1433.15 (41.75)
18	**RS**	0.00%	18.20 (1.55)	-	1500.00 (0)
15	**OLMCTS**	0.00%	15.35 (1.06)	-	1500.00 (0)
12	**OSLA**	0.00%	5.15 (0.98)	-	941.80 (97.14)
10	**RHEA**	0.00%	1.55 (0.76)	-	348.05 (29.66)

Table C.8



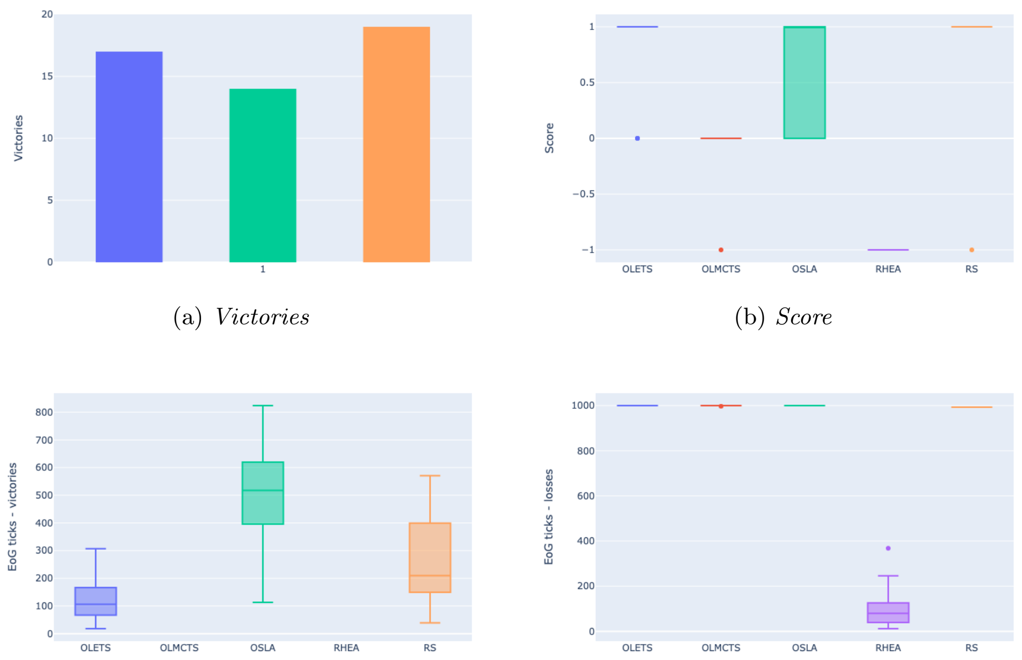
(a) *Victories* (b) *Score*

(c) *EoG ticks - victories* (d) *EoG ticks - defeats*

Figure C.8: WMH *Digdug* graphs

WMH: *Escape*					
Points	**Controller**	**% Wins**	**Score**	**EoG victories**	**EoG defeats**
25	**RS**	95.00%	0.90 (0.10)	257.58 (37.14)	993.00 (0)
18	**OLETS**	85.00%	0.85 (0.08)	118.59 (18.52)	1000.00 (0)
15	**OSLA**	70.00%	0.70 (0.10)	490.21 (51.94)	1000.00 (0)
12	**OLMCTS**	0.00%	-0.05 (0.05)	-	999.85 (0.15)
10	**RHEA**	0.00%	-1.00 (0)	-	104.40 (19.27)

Table C.9



(a) *Victories* (b) *Score*

(c) *EoG ticks - victories* (d) *EoG ticks - defeats*

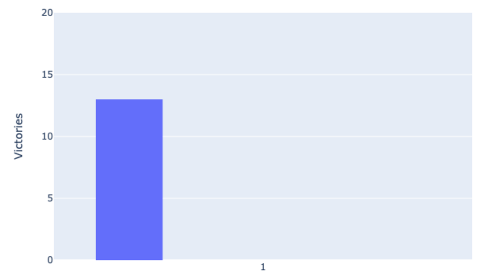
Figure C.9: WMH *Escape* graphs

WMH: *Hungry Birds*					
Points	**Controller**	**% Wins**	**Score**	**EoG victories**	**EoG defeats**
25	**OLETS**	65.00%	65.00 (10.67)	176.38 (22.77)	60.00 (0)
18	**OLMCTS**	0.00%	4.00 (2.68)	-	465.00 (64.82)
15	**RHEA**	0.00%	0.00 (0)	-	341.25 (50.00)
12	**RS**	0.00%	0.00 (0)	-	318.75 (24.00)
10	**OSLA**	0.00%	0.00 (0)	-	307.50 (47.46)

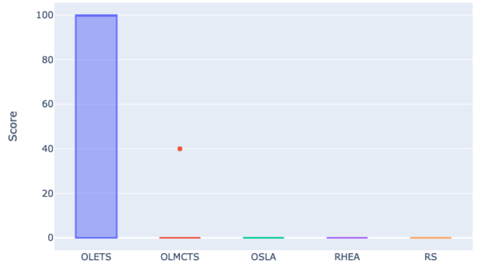
Table C.10

WMH: *Infection*					
Points	**Controller**	**% Wins**	**Score**	**EoG victories**	**EoG defeats**
25	**OLETS**	95.00%	25.30 (2.16)	720.00 (56.97)	1500.00 (0)
18	**OLMCTS**	90.00%	18.85 (1.92)	695.67 (57.00)	1500.00 (0)
15	**RS**	85.00%	19.50 (1.67)	876.00 (70.07)	1500.00 (0)
12	**OSLA**	85.00%	11.80 (1.26)	1032.53 (65.05)	1500.00 (0)
10	**RHEA**	55.00%	2.80 (1.29)	1040.36 (76.38)	1500.00 (0)

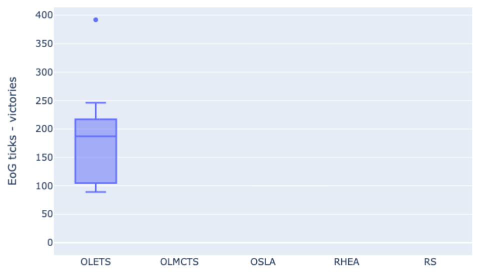
Table C.11



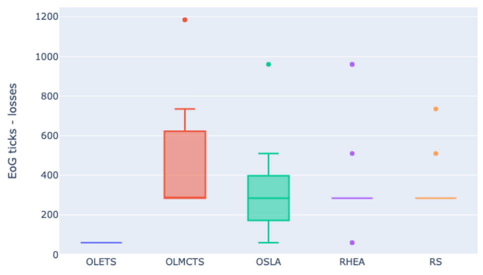
(a) *Victories*



(b) *Score*

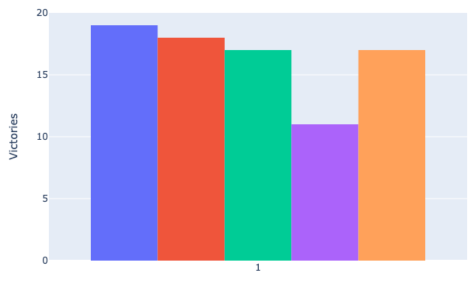


(c) *EoG ticks - victories*

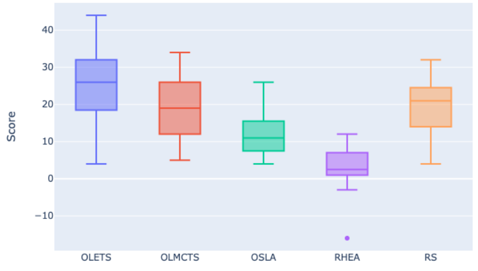


(d) *EoG ticks - defeats*

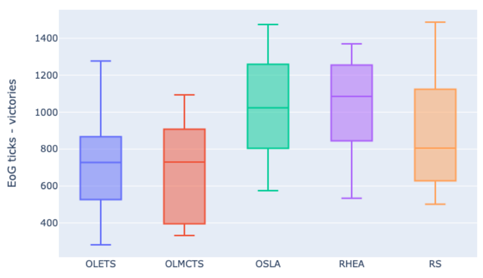
Figure C.10: WMH *Hungry Birds* graphs



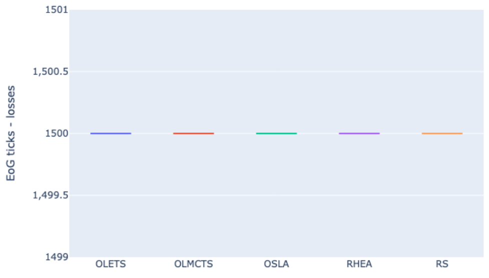
(a) *Victories*



(b) *Score*



(c) *EoG ticks - victories*

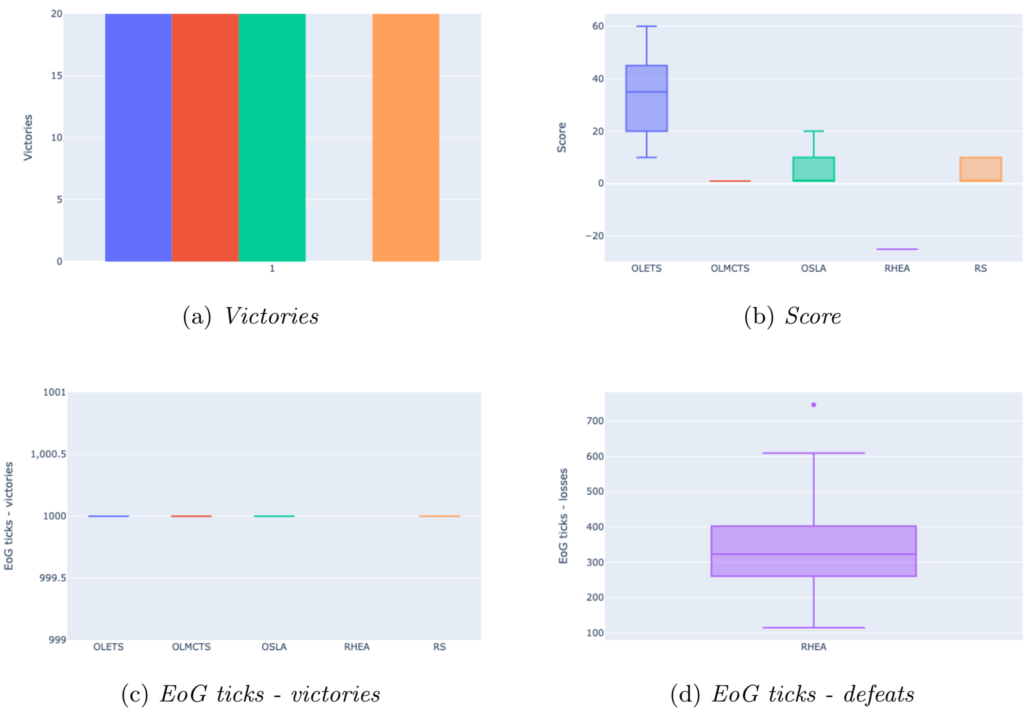


(d) *EoG ticks - defeats*

Figure C.11: WMH *Infection* graphs

WMH: *Intersection*					
Points	**Controller**	**% Wins**	**Score**	**EoG victories**	**EoG defeats**
25	**OLETS**	100.00%	34.50 (3.20)	1000.00 (0)	-
18	**OSLA**	100.00%	4.70 (1.39)	1000.00 (0)	-
15	**RS**	100.00%	4.15 (0.96)	1000.00 (0)	-
12	**OLMCTS**	100.00%	1.00 (0)	1000.00 (0)	-
10	**RHEA**	0.00%	-25.00 (0)	-	347.45 (32.89)

Table C.12

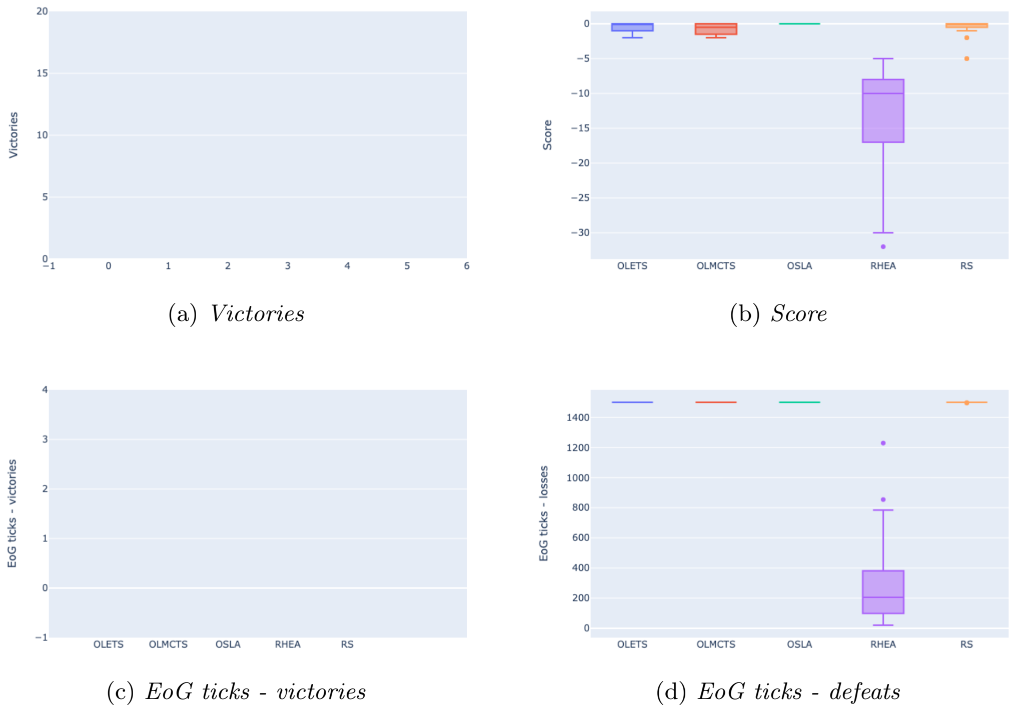

(a) *Victories* (b) *Score*

(c) *EoG ticks - victories* (d) *EoG ticks - defeats*

Figure C.12: WMH *Intersection* graphs

WMH: *Lemmings*					
Points	**Controller**	**% Wins**	**Score**	**EoG victories**	**EoG defeats**
25	**OSLA**	0.00%	0.00 (0)	-	1500.00 (0)
18	**OLETS**	0.00%	-0.45 (0.17)	-	1500.00 (0)
15	**OLMCTS**	0.00%	-0.75 (0.19)	-	1500.00 (0)
12	**RS**	0.00%	-0.75 (0.35)	-	1499.65 (0.26)
10	**RHEA**	0.00%	-13.20 (1.63)	-	315.30 (72.16)

Table C.13



(a) *Victories* (b) *Score*

(c) *EoG ticks - victories* (d) *EoG ticks - defeats*

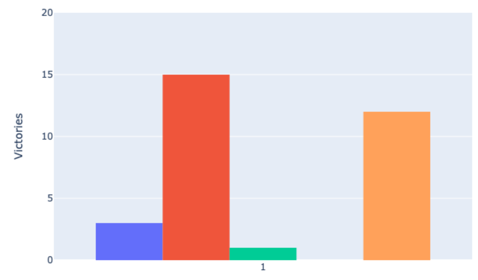
Figure C.13: WMH *Lemmings* graphs

WMH: *Missile Command*					
Points	**Controller**	**% Wins**	**Score**	**EoG victories**	**EoG defeats**
25	**OLMCTS**	75.00%	2.60 (0.54)	133.40 (10.31)	209.00 (0)
18	**RS**	60.00%	1.40 (0.50)	155.00 (13.20)	209.00 (0)
15	**OLETS**	15.00%	-2.05 (0.40)	143.00 (26.94)	117.12 (5.71)
12	**OSLA**	5.00%	-1.75 (0.29)	110.00 (0)	158.63 (13.76)
10	**RHEA**	0.00%	-2.70 (0.16)	-	124.85 (7.90)

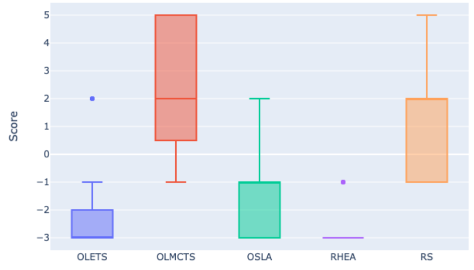
Table C.14

WMH: *Modality*					
Points	**Controller**	**% Wins**	**Score**	**EoG victories**	**EoG defeats**
25	**OLETS**	100.00%	1.00 (0)	7.20 (0.27)	-
18	**OLMCTS**	100.00%	1.00 (0)	10.95 (0.84)	-
15	**RS**	100.00%	1.00 (0)	15.40 (1.38)	-
12	**OSLA**	60.00%	0.60 (0.11)	130.42 (23.88)	1500.00 (0)
10	**RHEA**	50.00%	0.50 (0.11)	176.10 (48.73)	1500.00 (0)

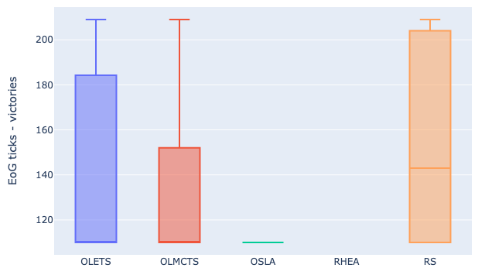
Table C.15



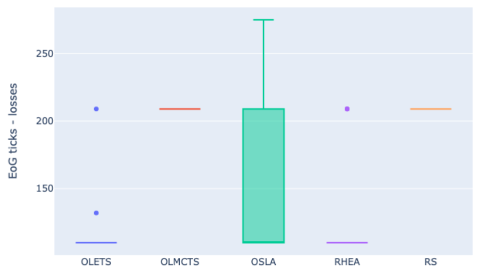
(a) *Victories*



(b) *Score*

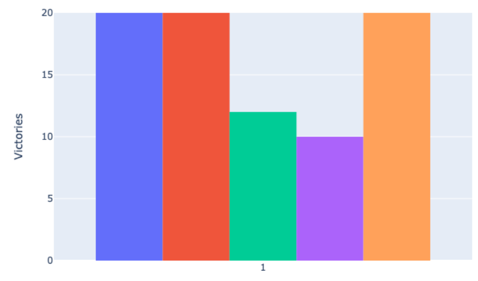


(c) *EoG ticks - victories*

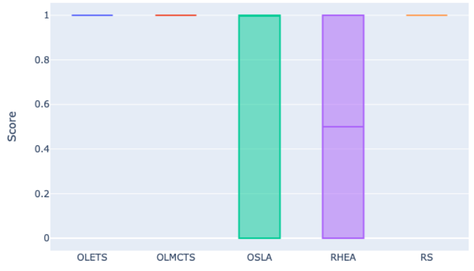


(d) *EoG ticks - defeats*

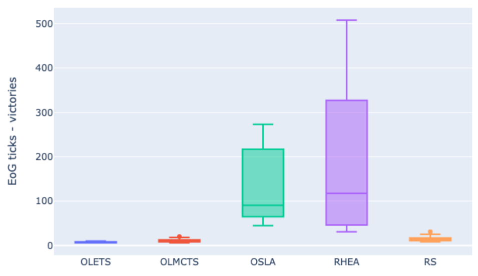
Figure C.14: WMH *Missile Command* graphs



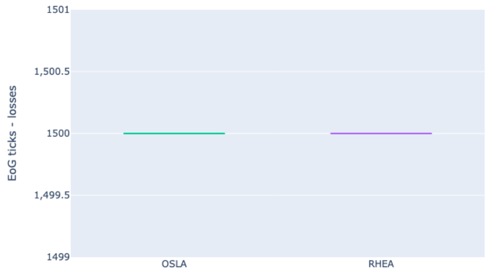
(a) *Victories*



(b) *Score*



(c) *EoG ticks - victories*

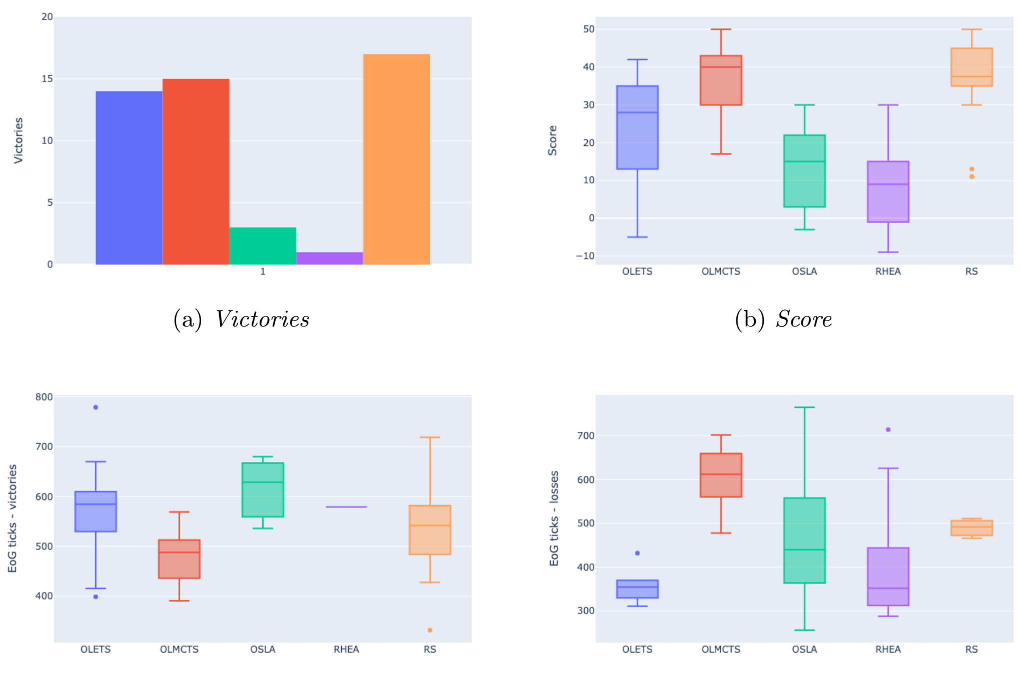


(d) *EoG ticks - defeats*

Figure C.15: WMH *Modality* graphs

WMH: *Plaque Attack*					
Points	**Controller**	**% Wins**	**Score**	**EoG victories**	**EoG defeats**
25	**RS**	85.00%	35.95 (2.58)	532.00 (20.78)	489.67 (10.65)
18	**OLMCTS**	75.00%	36.25 (2.16)	482.27 (13.66)	605.00 (33.15)
15	**OLETS**	70.00%	23.50 (3.09)	573.29 (25.53)	358.67 (15.62)
12	**OSLA**	15.00%	13.05 (2.41)	615.00 (34.42)	467.76 (34.16)
10	**RHEA**	5.00%	7.85 (2.22)	579.00 (0)	404.05 (27.89)

Table C.16



(a) *Victories*

(b) *Score*

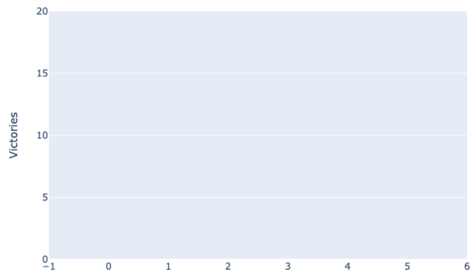
(c) *EoG ticks - victories*

(d) *EoG ticks - defeats*

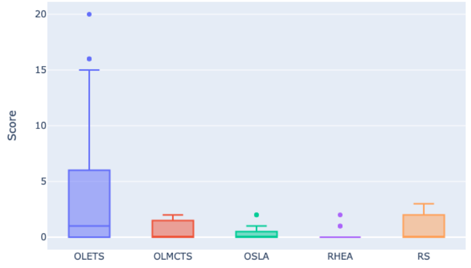
Figure C.16: WMH *Plaque Attack* graphs

WMH: *Roguelike*					
Points	**Controller**	**% Wins**	**Score**	**EoG victories**	**EoG defeats**
25	**OLETS**	0.00%	4.35 (1.46)	-	499.10 (119.84)
18	**RS**	0.00%	0.75 (0.22)	-	714.05 (105.24)
15	**OLMCTS**	0.00%	0.70 (0.19)	-	880.50 (93.96)
12	**OSLA**	0.00%	0.35 (0.15)	-	501.40 (55.88)
10	**RHEA**	0.00%	0.20 (0.11)	-	538.20 (100.91)

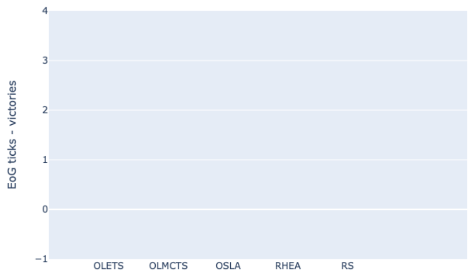
Table C.17



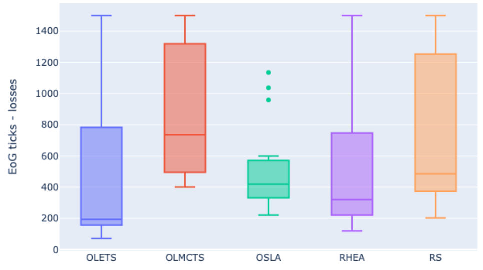
(a) *Victories*



(b) *Score*



(c) *EoG ticks - victories*



(d) *EoG ticks - defeats*

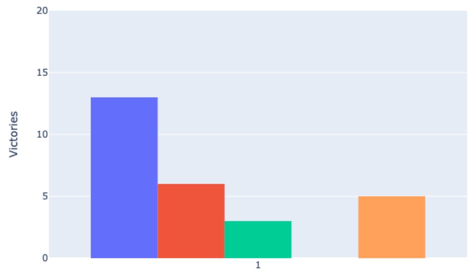
Figure C.17: WMH *Roguelike* graphs

WMH: *Seaquest*					
Points	**Controller**	**% Wins**	**Score**	**EoG victories**	**EoG defeats**
25	**OLETS**	65.00%	1184.00 (192.44)	1000.00 (0)	483.71 (131.51)
18	**OLMCTS**	30.00%	442.25 (149.97)	1000.00 (0)	610.50 (49.96)
15	**RS**	25.00%	231.20 (91.72)	1000.00 (0)	375.13 (44.08)
12	**OSLA**	15.00%	71.40 (49.37)	1000.00 (0)	529.47 (50.13)
10	**RHEA**	0.00%	4.20 (0.78)	-	244.15 (30.78)

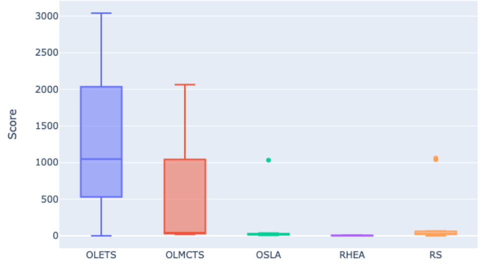
Table C.18

WMH: *Survive Zombies*					
Points	**Controller**	**% Wins**	**Score**	**EoG victories**	**EoG defeats**
25	**RS**	90.00%	7.05 (0.64)	1000.00 (0)	773.00 (82.02)
18	**OLETS**	90.00%	6.70 (0.64)	1000.00 (0)	771.50 (24.40)
15	**OLMCTS**	85.00%	6.20 (0.82)	1000.00 (0)	597.00 (51.86)
12	**OSLA**	85.00%	6.05 (0.74)	1000.00 (0)	697.33 (60.49)
10	**RHEA**	15.00%	2.35 (0.94)	1000.00 (0)	427.41 (45.49)

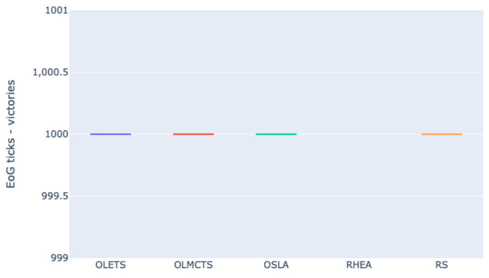
Table C.19



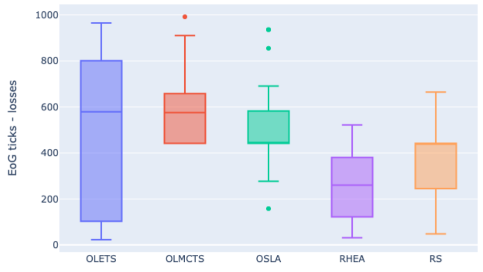
(a) *Victories*



(b) *Score*

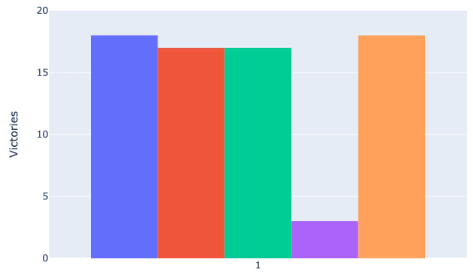


(c) *EoG ticks - victories*

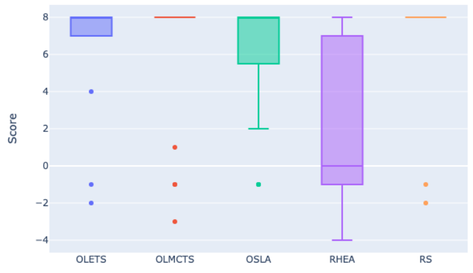


(d) *EoG ticks - defeats*

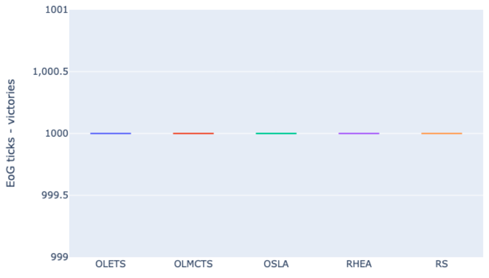
Figure C.18: WMH *Seaquest* graphs



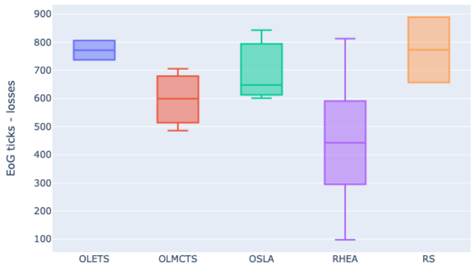
(a) *Victories*



(b) *Score*



(c) *EoG ticks - victories*

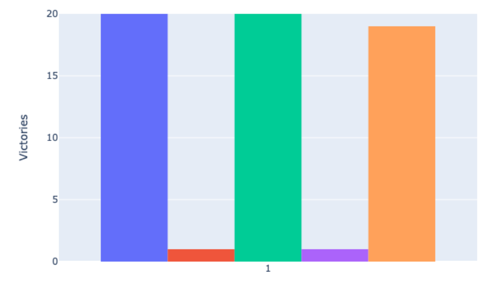


(d) *EoG ticks - defeats*

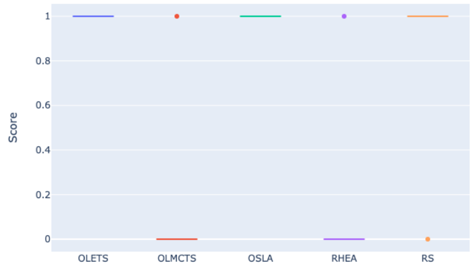
Figure C.19: WMH *Survive Zombies* graphs

WMH: *Wait for Breakfast*					
Points	**Controller**	**% Wins**	**Score**	**EoG victories**	**EoG defeats**
25	**OLETS**	100.00%	1.00 (0)	92.65 (13.71)	-
18	**OSLA**	100.00%	1.00 (0)	218.70 (65.76)	-
15	**RS**	95.00%	0.95 (0.05)	111.68 (40.76)	1500.00 (0)
12	**RHEA**	5.00%	0.05 (0.05)	50.00 (0)	95.05 (76.12)
10	**OLMCTS**	5.00%	0.05 (0.05)	1495.00 (0)	1500.00 (0)

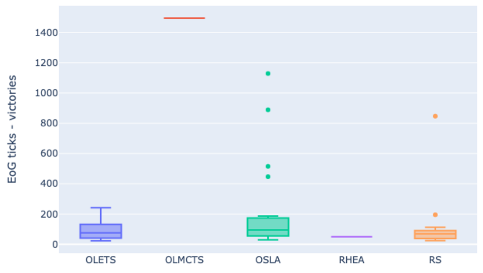
Table C.20



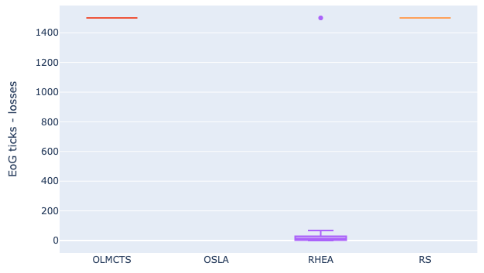
(a) *Victories*



(b) *Score*



(c) *EoG ticks - victories*



(d) *EoG ticks - defeats*

Figure C.20: WMH *Wait for Breakfast* graphs

C.2 Per Game Results: EMH

Each table contains the EMH results of one of the games displaying the information listed below. The data is the final average resulting from the 20 gameplays of each agent. The values in parenthesis in the table represent the corresponding *Std. Deviation*.

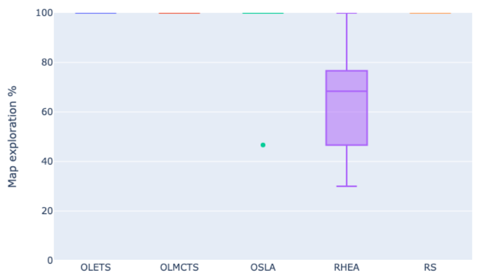
- Total number of points received in the game (Points)
- Agent (Controller)
- Average of percentage of the available level explored (% Explored)
- Average of game-ticks to the last new exploration (Game-tick last exploration)

For each game, it is also included the resulting graphs for the stats related to the heuristic: map exploration percentage, and game ticks when the last new exploration happened. They show the resulting stats of the 20 gameplays achieved by each agent, in order: OLETS (blue), OLMCTS (red), OSLA (green), RHEA (purple), and RS (orange).

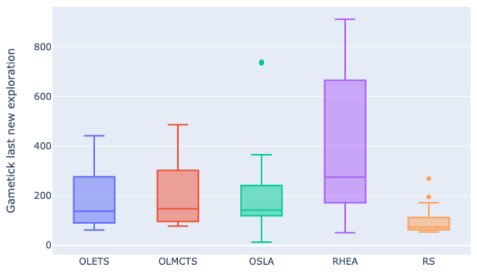
The games are included in alphabetic order.

EMH: *Aliens*			
Points	**Controller**	**% Explored**	**Game-ticks last exploration**
25	**RS**	100.00% (0)	98.10 (12.11)
18	**OLETS**	100.00% (0)	184.35 (24.35)
15	**OLMCTS**	100.00% (0)	202.45 (28.23)
12	**OSLA**	97.33% (2.60)	216.25 (42.62)
10	**RHEA**	64.00% (4.37)	395.00 (61.09)

Table C.21



(a) *Map exploration %*

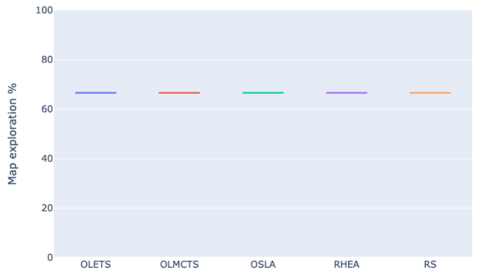


(b) *Game-ticks last new exploration*

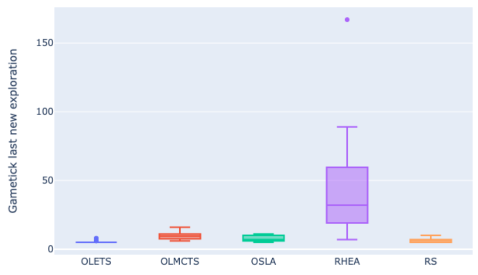
Figure C.21: EMH *Aliens* graphs

EMH: *Bait*			
Points	**Controller**	**% Explored**	**Game-ticks last exploration**
25	**OLETS**	66.67% (0)	5.25 (0.16)
18	**RS**	66.67% (0)	6.35 (0.32)
15	**OSLA**	66.67% (0)	7.80 (0.50)
12	**OLMCTS**	66.67% (0)	9.75 (0.57)
10	**RHEA**	66.67% (0)	44.95 (8.36)

Table C.22



(a) *Map exploration %*

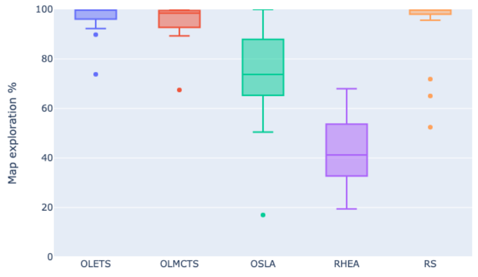


(b) *Game-ticks last new exploration*

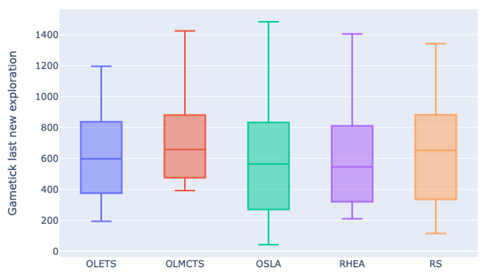
Figure C.22: EMH *Bait* graphs

EMH: *Butterflies*			
Points	**Controller**	**% Explored**	**Game-ticks last exploration**
25	**OLETS**	96.87% (1.35)	632.30 (62.30)
18	**OLMCTS**	95.27% (1.62)	746.30 (70.68)
15	**RS**	93.91% (2.99)	626.35 (73.45)
12	**OSLA**	73.67% (4.32)	621.90 (93.30)
10	**RHEA**	42.60% (3.16)	589.15 (70.07)

Table C.23



(a) *Map exploration %*

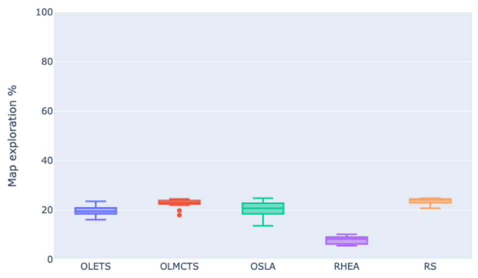


(b) *Game-ticks last new exploration*

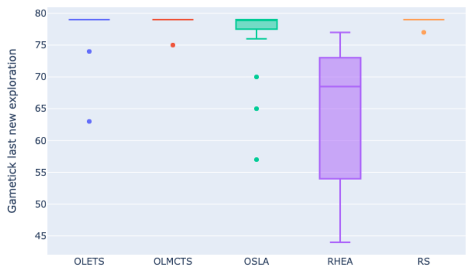
Figure C.23: EMH *Butterflies* graphs

EMH: *Camel Race*			
Points	**Controller**	**% Explored**	**Game-ticks last exploration**
25	**RS**	23.71% (0.27)	78.90 (0.10)
18	**OLMCTS**	22.87% (0.34)	78.80 (0.19)
15	**OSLA**	20.61% (0.67)	76.45 (1.27)
12	**OLETS**	19.83% (0.42)	77.95 (0.80)
10	**RHEA**	7.83% (0.34)	65.15 (2.30)

Table C.24



(a) *Map exploration %*

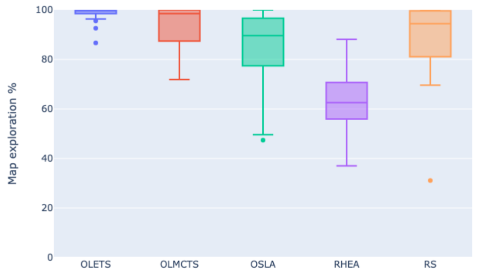


(b) *Game-ticks last new exploration*

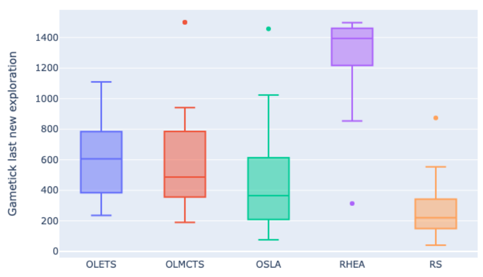
Figure C.24: EMH *Camel Race* graphs

EMH: *Chase*			
Points	**Controller**	**% Explored**	**Game-ticks last exploration**
25	**OLETS**	98.26% (0.73)	613.25 (55.40)
18	**OLMCTS**	93.30% (1.99)	611.20 (81.18)
15	**RS**	88.59% (3.56)	269.40 (41.26)
12	**OSLA**	84.19% (3.59)	451.25 (74.65)
10	**RHEA**	63.70% (2.72)	1282.05 (60.82)

Table C.25



(a) *Map exploration %*

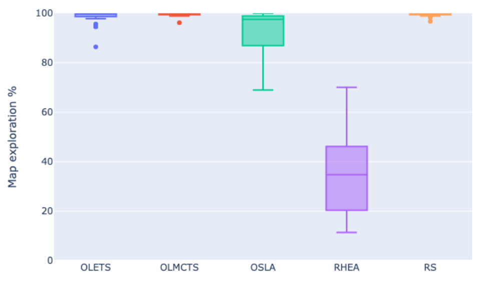


(b) *Game-ticks last new exploration*

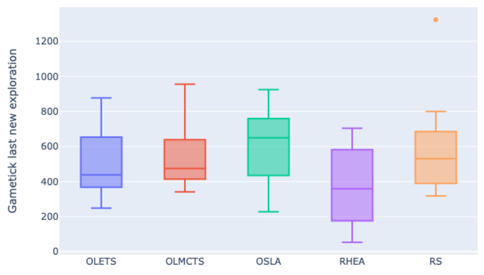
Figure C.25: EMH *Chase* graphs

EMH: *Chopper*			
Points	**Controller**	**% Explored**	**Game-ticks last exploration**
25	**OLMCTS**	99.59% (0.19)	549.20 (39.37)
18	**RS**	99.54% (0.19)	567.65 (51.22)
15	**OLETS**	98.40% (0.74)	502.05 (39.63)
12	**OSLA**	92.20% (2.10)	605.75 (44.36)
10	**RHEA**	35.54% (3.75)	369.50 (47.88)

Table C.26



(a) *Map exploration %*

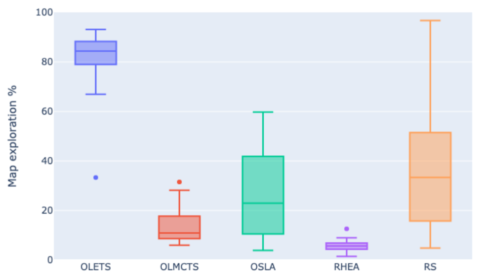


(b) *Game-ticks last new exploration*

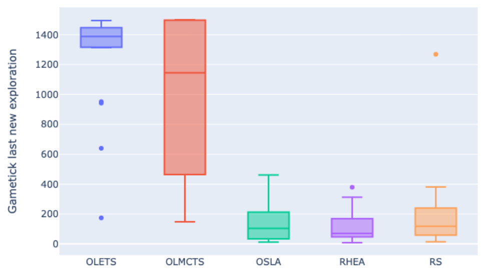
Figure C.26: EMH *Chopper* graphs

EMH: *Crossfire*			
Points	**Controller**	**% Explored**	**Game-ticks last exploration**
25	**OLETS**	81.13% (2.87)	1260.75 (73.42)
18	**RS**	37.81% (5.63)	197.80 (59.29)
15	**OSLA**	26.64% (4.02)	136.40 (27.63)
12	**OLMCTS**	13.75% (1.65)	962.95 (107.54)
10	**RHEA**	5.84% (0.55)	113.35 (22.05)

Table C.27



(a) *Map exploration %*

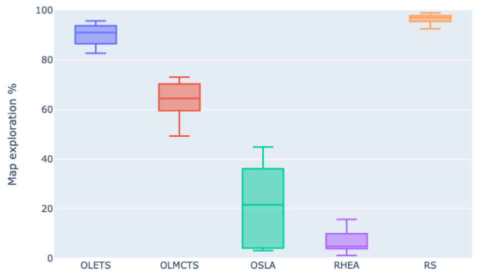


(b) *Game-ticks last new exploration*

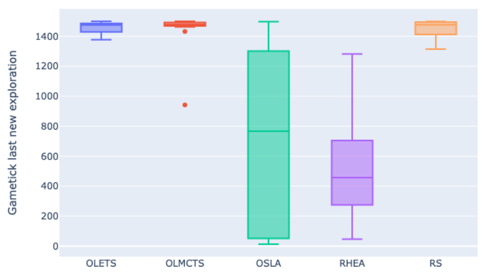
Figure C.27: EMH *Crossfire* graphs

EMH: *Digdug*			
Points	**Controller**	**% Explored**	**Game-ticks last exploration**
25	**RS**	96.56% (0.37)	1448.90 (12.92)
18	**OLETS**	90.41% (0.89)	1457.50 (8.69)
15	**OLMCTS**	64.19% (1.45)	1453.10 (26.45)
12	**OSLA**	20.83% (3.26)	702.05 (131.43)
10	**RHEA**	7.11% (1.02)	515.95 (73.94)

Table C.28



(a) *Map exploration %*

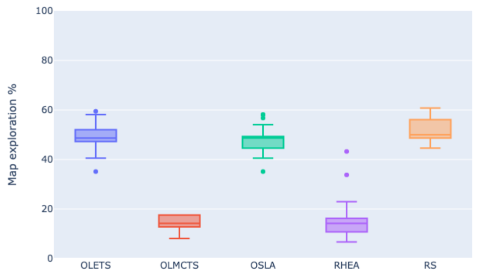


(b) *Game-ticks last new exploration*

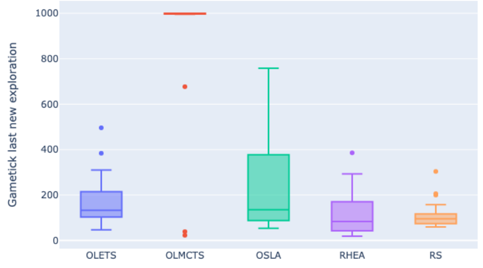
Figure C.28: EMH *Digdug* graphs

EMH: *Escape*			
Points	**Controller**	**% Explored**	**Game-ticks last exploration**
25	**RS**	51.62% (1.00)	113.60 (13.23)
18	**OLETS**	49.19% (1.27)	173.10 (25.48)
15	**OSLA**	47.97% (1.26)	239.90 (44.05)
12	**RHEA**	16.01% (1.85)	118.30 (20.90)
10	**OLMCTS**	14.32% (0.69)	886.20 (65.64)

Table C.29



(a) *Map exploration %*

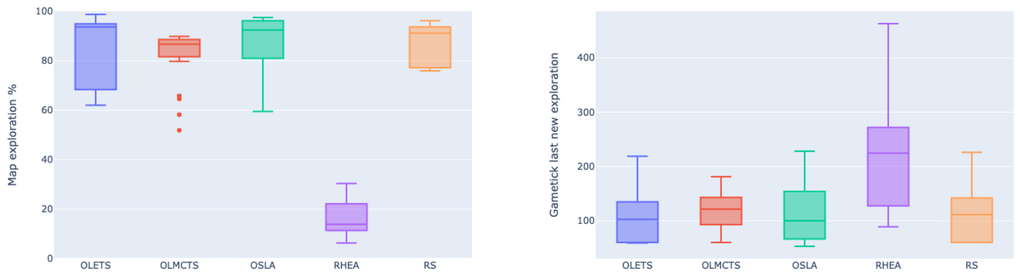


(b) *Game-ticks last new exploration*

Figure C.29: EMH *Escape* graphs

EMH: *Hungry Birds*			
Points	**Controller**	**% Explored**	**Game-ticks last exploration**
25	**OSLA**	88.10% (2.40)	113.95 (11.70)
18	**RS**	87.91% (1.69)	112.45 (9.81)
15	**OLETS**	84.05% (3.08)	101.35 (9.38)
12	**OLMCTS**	81.46% (2.50)	116.30 (7.84)
10	**RHEA**	16.39% (1.53)	212.30 (20.72)

Table C.30

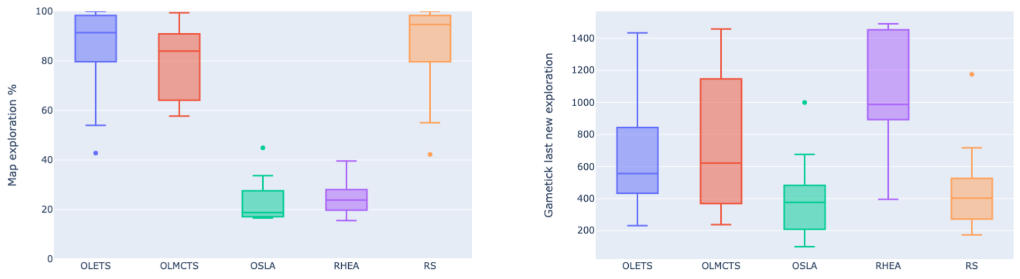


(a) *Map exploration %* (b) *Game-ticks last new exploration*

Figure C.30: EMH *Hungry Birds* graphs

EMH: *Infection*			
Points	**Controller**	**% Explored**	**Game-ticks last exploration**
25	**RS**	85.64% (3.91)	434.75 (50.62)
18	**OLETS**	85.43% (3.66)	661.95 (71.50)
15	**OLMCTS**	79.06% (3.09)	753.30 (95.62)
12	**RHEA**	24.60% (1.29)	1095.10 (69.88)
10	**OSLA**	22.75% (1.75)	385.50 (48.67)

Table C.31

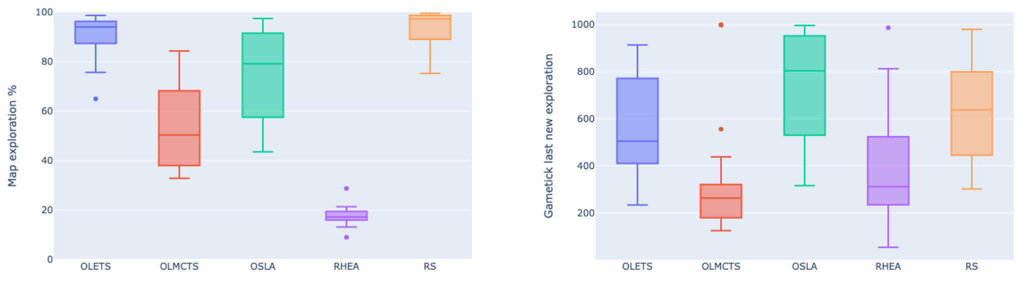


(a) *Map exploration %* (b) *Game-ticks last new exploration*

Figure C.31: EMH *Infection* graphs

EMH: *Intersection*			
Points	**Controller**	**% Explored**	**Game-ticks last exploration**
25	**RS**	93.68% (1.57)	630.70 (48.73)
18	**OLETS**	89.79% (1.97)	567.95 (48.76)
15	**OSLA**	74.34% (4.00)	739.45 (52.01)
12	**OLMCTS**	53.54% (3.57)	333.00 (54.24)
10	**RHEA**	17.70% (0.84)	410.30 (53.70)

Table C.32

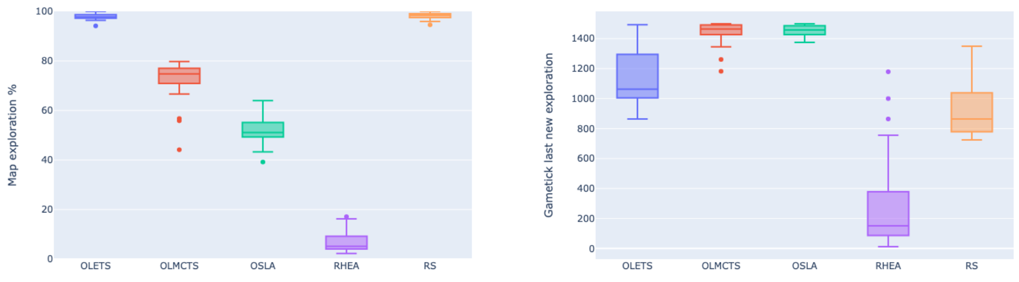


(a) *Map exploration %* (b) *Game-ticks last new exploration*

Figure C.32: EMH *Intersection* graphs

EMH: *Lemmings*			
Points	**Controller**	**% Explored**	**Game-ticks last exploration**
25	**RS**	98.18% (0.29)	928.90 (41.71)
18	**OLETS**	97.75% (0.27)	1142.55 (40.24)
15	**OLMCTS**	71.31% (1.97)	1435.65 (18.47)
12	**OSLA**	51.58% (1.20)	1451.85 (8.42)
10	**RHEA**	7.14% (0.99)	309.50 (76.45)

Table C.33

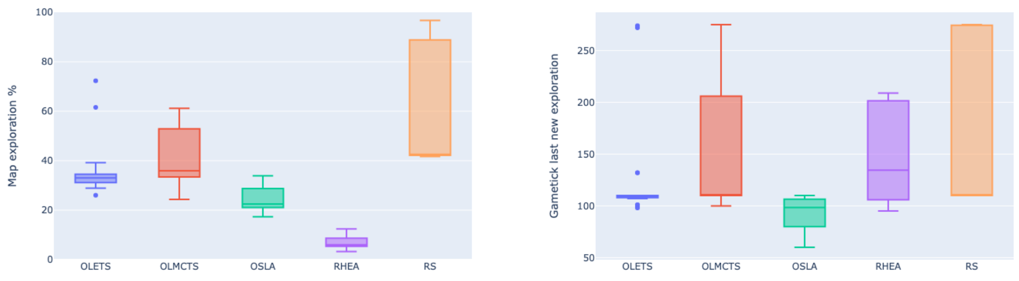


(a) *Map exploration %* (b) *Game-ticks last new exploration*

Figure C.33: EMH *Lemmings* graphs

EMH: *Missile Command*			
Points	**Controller**	**% Explored**	**Game-ticks last exploration**
25	**RS**	61.45% (5.30)	174.05 (17.59)
18	**OLMCTS**	40.76% (2.44)	148.70 (12.65)
15	**OLETS**	36.16% (2.40)	126.20 (11.10)
12	**OSLA**	24.07% (1.02)	92.30 (3.72)
10	**RHEA**	6.90% (0.50)	149.05 (10.15)

Table C.34

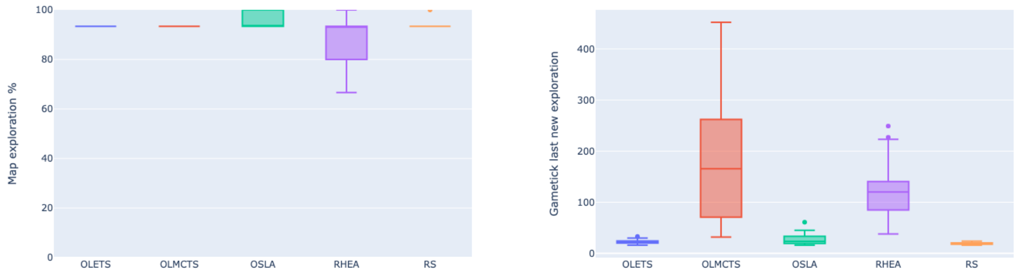


(a) *Map exploration %* (b) *Game-ticks last new exploration*

Figure C.34: EMH *Missile Command* graphs

EMH: *Modality*			
Points	**Controller**	**% Explored**	**Game-ticks last exploration**
25	**OSLA**	96.00% (0.73)	27.60 (2.52)
18	**RS**	93.67% (0.32)	19.20 (0.46)
15	**OLETS**	93.33% (0)	22.85 (0.92)
12	**OLMCTS**	93.33% (0)	185.50 (27.29)
10	**RHEA**	88.00% (2.24)	123.85 (12.44)

Table C.35

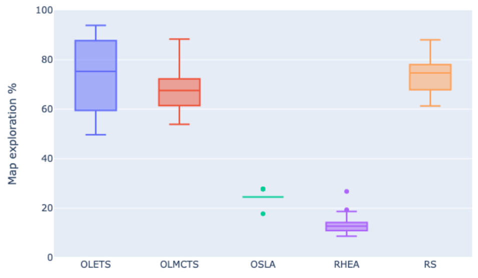


(a) *Map exploration %* (b) *Game-ticks last new exploration*

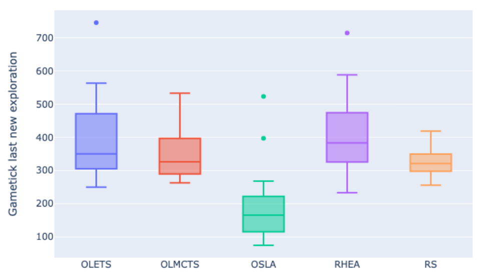
Figure C.35: EMH *Modality* graphs

EMH: *Plaque Attack*			
Points	**Controller**	**% Explored**	**Game-ticks last exploration**
25	**RS**	74.05% (1.49)	325.80 (8.42)
18	**OLETS**	73.66% (3.12)	392.85 (26.96)
15	**OLMCTS**	67.60% (1.88)	346.95 (16.89)
12	**OSLA**	24.82% (0.46)	190.30 (23.72)
10	**RHEA**	13.60% (0.90)	411.75 (25.82)

Table C.36



(a) *Map exploration %*

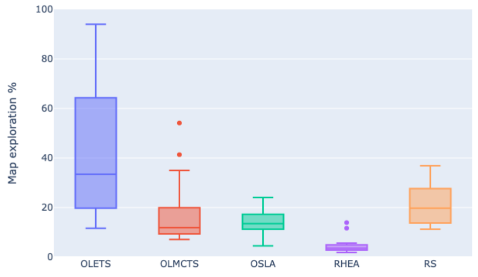


(b) *Game-ticks last new exploration*

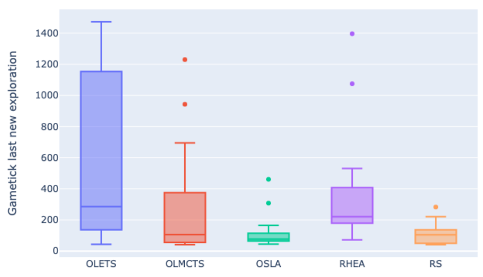
Figure C.36: EMH *Plaque Attack* graphs

EMH: *Roguelike*			
Points	**Controller**	**% Explored**	**Game-ticks last exploration**
25	**OLETS**	42.22% (5.50)	551.70 (116.54)
18	**RS**	20.71% (1.77)	109.90 (14.47)
15	**OLMCTS**	17.61% (2.86)	268.20 (73.45)
12	**OSLA**	14.12% (1.15)	112.70 (21.89)
10	**RHEA**	4.51% (0.67)	343.90 (73.38)

Table C.37



(a) *Map exploration %*

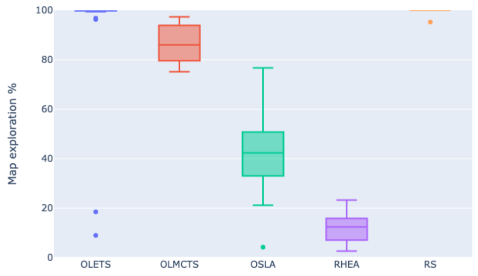


(b) *Game-ticks last new exploration*

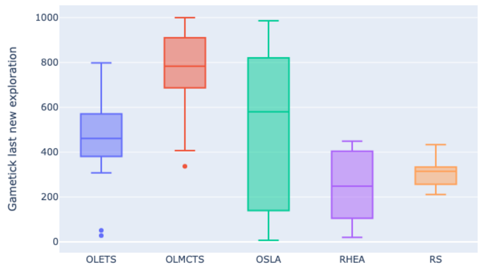
Figure C.37: EMH *Roguelike* graphs

EMH: ***Seaquest***			
Points	**Controller**	**% Explored**	**Game-ticks last exploration**
25	**RS**	99.76% (0.23)	306.80 (13.39)
18	**OLETS**	91.01% (5.77)	462.60 (41.60)
15	**OLMCTS**	86.67% (1.62)	762.55 (44.17)
12	**OSLA**	42.28% (3.35)	499.25 (74.56)
10	**RHEA**	11.80% (1.23)	257.60 (32.92)

Table C.38



(a) *Map exploration %*

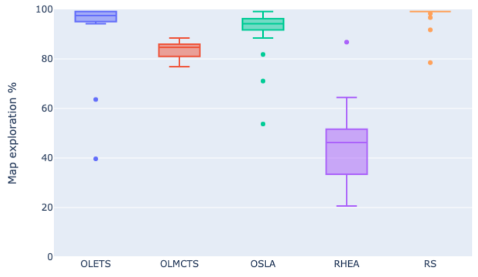


(b) *Game-ticks last new exploration*

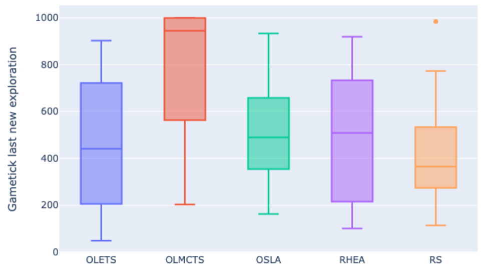
Figure C.38: EMH *Seaquest* graphs

EMH: ***Survive Zombies***			
Points	**Controller**	**% Explored**	**Game-ticks last exploration**
25	**RS**	97.60% (1.05)	426.65 (47.69)
18	**OLETS**	92.73% (3.20)	439.65 (58.64)
15	**OSLA**	90.99% (2.38)	508.95 (48.06)
12	**OLMCTS**	83.43% (0.76)	813.15 (54.72)
10	**RHEA**	43.64% (3.55)	494.30 (54.95)

Table C.39



(a) *Map exploration %*

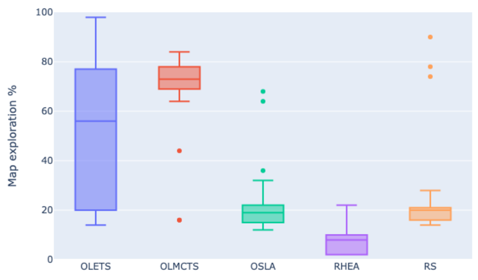


(b) *Game-ticks last new exploration*

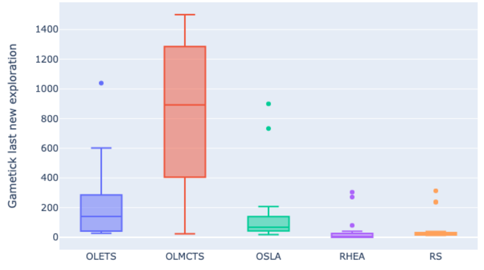
Figure C.39: EMH *Survive Zombies* graphs

EMH: *Wait for Breakfast*			
Points	**Controller**	**% Explored**	**Game-ticks last exploration**
25	**OLMCTS**	67.30% (4.25)	850.10 (106.67)
18	**OLETS**	50.30% (6.50)	220.00 (54.62)
15	**RS**	27.70% (5.06)	58.30 (19.51)
12	**OSLA**	23.60% (3.43)	151.85 (50.98)
10	**RHEA**	7.70% (1.31)	40.30 (18.96)

Table C.40



(a) *Map exploration %*



(b) *Game-ticks last new exploration*

Figure C.40: EMH *Wait for Breakfast* graphs

C.3 Per Game Results: KDH

Each table contains the KDH results of one of the games displaying the information listed below. The data is the final average resulting from the 20 gameplays of each agent. The values in parenthesis in the table represent the corresponding *Std. Deviation*.

- Total number of points received in the game (Points)
- Agent (Controller)
- Average of the total number of sprites acknowledged (Sprites Ack.)
- Average of the unique interactions (Unique Int.)
- Average of the curiosity collisions (CC)
- Average of the curiosity actions-onto (CA)
- Average of game-ticks to the last sprite acknowledged (Game-ticks Ack.)
- Average of game-ticks to the last unique interaction (Game-ticks Int.)
- Average of game-ticks to the last curiosity collision achieved (Game-ticks CC)
- Average of game-ticks to the last curiosity action-onto achieved (Game-ticks CA)

After the tables, for each game, it is also included the resulting graphs for the stats related to the heuristic: sprites acknowledged, unique interactions, curiosity collisions, and curiosity actions-onto. They show the resulting stats of the 20 gameplays achieved by each agent, in order: OLETS (blue), OLMCTS (red), OSLA (green), RHEA (purple), and RS (orange).

The games are included in alphabetic order.

KDH: *Aliens*									
Points	**Controller**	**Sprites Ack.**	**Unique Int.**	**CC**	**CA**	**Game-ticks Ack.**	**Game-ticks Int.**	**Game-ticks CC**	**Game-ticks CA**
25	**RS**	4.00 (0)	2.85 (0.08)	0.85 (0.08)	56.05 (0.62)	0.00 (0)	919.75 (73.65)	898.40 (84.91)	633.95 (46.83)
18	**OLMCTS**	4.00 (0)	2.80 (0.09)	0.80 (0.09)	56.65 (0.97)	1.25 (0.39)	856.30 (83.16)	832.40 (93.62)	588.90 (39.90)
15	**RHEA**	4.00 (0)	2.70 (0.10)	0.70 (0.10)	36.85 (2.27)	1.05 (0.32)	440.35 (74.93)	409.90 (82.13)	587.80 (55.12)
12	**OLETS**	4.00 (0)	2.65 (0.11)	0.65 (0.11)	46.50 (1.43)	0.65 (0.23)	681.85 (96.05)	646.00 (106.84)	717.25 (26.33)
10	**OSLA**	4.00 (0)	2.45 (0.11)	0.45 (0.11)	40.50 (1.21)	38.45 (6.01)	453.75 (89.50)	400.75 (99.80)	753.15 (20.05)

Table C.41

KDH: *Bait*									
Points	**Controller**	**Sprites Ack.**	**Unique Int.**	**CC**	**CA**	**Game-ticks Ack.**	**Game-ticks Int.**	**Game-ticks CC**	**Game-ticks CA**
25	**RHEA**	4.00 (0)	3.35 (0.11)	10.45 (0.17)	0.00 (0)	0.00 (0)	47.60 (12.83)	124.85 (18.74)	0.00 (0)
18	**OLMCTS**	4.00 (0)	3.30 (0.10)	10.50 (0.18)	0.00 (0)	0.00 (0)	6.40 (0.91)	395.80 (142.20)	0.00 (0)
15	**RS**	4.00 (0)	3.15 (0.08)	10.30 (0.16)	0.00 (0)	0.00 (0)	5.60 (0.62)	242.95 (117.80)	0.00 (0)
12	**OLETS**	4.00 (0)	3.00 (0)	10.00 (0)	0.00 (0)	0.00 (0)	4.45 (0.15)	22.60 (0.45)	0.00 (0)
10	**OSLA**	4.00 (0)	3.00 (0)	10.00 (0)	0.00 (0)	0.00 (0)	5.15 (0.54)	53.05 (6.06)	0.00 (0)

Table C.42

KDH: *Butterflies*									
Points	**Controller**	**Sprites Ack.**	**Unique Int.**	**CC**	**CA**	**Game-ticks Ack.**	**Game-ticks Int.**	**Game-ticks CC**	**Game-ticks CA**
25	**OLMCTS**	3.00 (0)	2.00 (0)	112.95 (5.52)	0.00 (0)	0.00 (0)	26.35 (6.73)	896.30 (91.31)	0.00 (0)
18	**RS**	3.00 (0)	2.00 (0)	111.55 (5.10)	0.00 (0)	0.00 (0)	16.90 (2.53)	892.15 (95.91)	0.00 (0)
15	**OLETS**	3.00 (0)	2.00 (0)	103.55 (4.84)	0.00 (0)	0.00 (0)	34.40 (8.19)	870.35 (94.15)	0.00 (0)
12	**RHEA**	3.00 (0)	2.00 (0)	97.75 (7.00)	0.00 (0)	0.00 (0)	23.80 (5.57)	705.85 (80.25)	0.00 (0)
10	**OSLA**	3.00 (0)	2.00 (0)	81.85 (5.76)	0.00 (0)	0.00 (0)	38.65 (5.09)	1040.65 (92.57)	0.00 (0)

Table C.43

KDH: *Camel Race*									
Points	**Controller**	**Sprites Ack.**	**Unique Int.**	**CC**	**CA**	**Game-ticks Ack.**	**Game-ticks Int.**	**Game-ticks CC**	**Game-ticks CA**
25	**OLETS**	7.00 (0)	1.00 (0)	24.55 (0.23)	0.00 (0)	68.75 (0.26)	0.00 (0)	73.40 (0.39)	0.00 (0)
18	**OLMCTS**	7.00 (0)	1.00 (0)	23.25 (1.02)	0.00 (0)	70.00 (0)	0.60 (0.16)	76.30 (1.14)	0.00 (0)
15	**RS**	7.00 (0)	1.00 (0)	14.35 (0.55)	0.00 (0)	70.00 (0)	1.60 (0.43)	76.30 (0.64)	0.00 (0)
12	**RHEA**	7.00 (0)	1.00 (0)	14.25 (0.55)	0.00 (0)	70.00 (0)	1.90 (0.59)	69.20 (1.76)	0.00 (0)
10	**OSLA**	7.00 (0)	1.00 (0)	8.95 (0.51)	0.00 (0)	79.00 (0)	0.00 (0)	58.60 (3.39)	0.00 (0)

Table C.44

KDH: *Chase*									
Points	**Controller**	**Sprites Ack.**	**Unique Int.**	**CC**	**CA**	**Game-ticks Ack.**	**Game-ticks Int.**	**Game-ticks CC**	**Game-ticks CA**
25	**RS**	4.00 (0)	3.00 (0)	83.20 (4.97)	0.00 (0)	194.35 (32.63)	567.90 (78.53)	567.90 (78.53)	0.00 (0)
18	**RHEA**	4.00 (0)	2.95 (0.05)	75.05 (4.56)	0.00 (0)	244.30 (23.83)	421.20 (42.29)	456.90 (47.31)	0.00 (0)
15	**OLMCTS**	4.00 (0)	2.90 (0.07)	87.15 (5.01)	0.00 (0)	159.35 (21.25)	647.00 (87.29)	680.60 (80.76)	0.00 (0)
12	**OLETS**	4.00 (0)	2.85 (0.08)	88.05 (3.71)	0.00 (0)	298.55 (36.96)	761.50 (84.10)	830.25 (75.27)	0.00 (0)
10	**OSLA**	3.60 (0.15)	2.50 (0.15)	66.30 (3.89)	0.00 (0)	622.45 (96.71)	661.05 (98.02)	1008.35 (92.47)	0.00 (0)

Table C.45

KDH: *Chopper*									
Points	**Controller**	**Sprites Ack.**	**Unique Int.**	**CC**	**CA**	**Game-ticks Ack.**	**Game-ticks Int.**	**Game-ticks CC**	**Game-ticks CA**
25	**RS**	9.00 (0)	6.05 (0.09)	110.85 (2.49)	26.30 (0.95)	83.40 (5.27)	556.05 (85.18)	1332.35 (44.21)	1346.25 (35.70)
18	**OLETS**	9.00 (0)	5.95 (0.05)	106.20 (2.16)	19.70 (0.99)	92.55 (7.36)	443.45 (34.65)	1153.00 (55.32)	1233.40 (42.84)
15	**OSLA**	9.00 (0)	5.45 (0.11)	63.80 (4.33)	16.95 (1.05)	140.15 (11.19)	567.00 (56.62)	1124.10 (52.07)	1122.30 (46.50)
12	**RHEA**	8.95 (0.05)	6.40 (0.20)	69.65 (5.24)	9.30 (1.34)	89.30 (9.96)	636.20 (55.14)	670.20 (56.74)	596.65 (60.92)
10	**OLMCTS**	8.95 (0.05)	6.05 (0.19)	109.20 (5.65)	22.55 (1.57)	64.20 (5.79)	641.30 (97.75)	1238.60 (82.23)	1240.85 (83.18)

Table C.46

KDH: *Crossfire*									
Points	**Controller**	**Sprites Ack.**	**Unique Int.**	**CC**	**CA**	**Game-ticks Ack.**	**Game-ticks Int.**	**Game-ticks CC**	**Game-ticks CA**
25	**RS**	4.00 (0)	2.00 (0)	47.40 (6.29)	0.00 (0)	0.00 (0)	333.90 (51.72)	333.90 (51.72)	0.00 (0)
18	**OSLA**	4.00 (0)	2.00 (0)	34.30 (4.24)	0.00 (0)	0.00 (0)	427.20 (69.87)	427.20 (69.87)	0.00 (0)
15	**OLMCTS**	4.00 (0)	2.00 (0)	27.75 (2.47)	0.00 (0)	0.00 (0)	260.50 (38.31)	260.50 (38.31)	0.00 (0)
12	**RHEA**	4.00 (0)	2.00 (0)	21.90 (2.17)	0.00 (0)	0.00 (0)	134.05 (17.69)	134.05 (17.69)	0.00 (0)
10	**OLETS**	4.00 (0)	1.30 (0.10)	106.20 (5.72)	0.00 (0)	0.00 (0)	300.80 (121.59)	1258.35 (78.49)	0.00 (0)

Table C.47

KDH: *Digdug*									
Points	**Controller**	**Sprites Ack.**	**Unique Int.**	**CC**	**CA**	**Game-ticks Ack.**	**Game-ticks Int.**	**Game-ticks CC**	**Game-ticks CA**
25	**RHEA**	8.95 (0.05)	8.25 (0.20)	53.10 (6.48)	156.70 (22.24)	249.60 (32.39)	574.80 (86.25)	800.15 (112.20)	799.10 (112.92)
18	**RS**	8.95 (0.05)	8.05 (0.18)	81.95 (5.11)	250.45 (16.24)	192.40 (21.38)	640.60 (105.05)	1316.55 (81.76)	1313.50 (82.58)
15	**OLMCTS**	8.95 (0.05)	7.90 (0.17)	113.75 (6.96)	336.25 (21.82)	137.60 (15.73)	641.45 (104.18)	1291.55 (85.03)	1297.10 (86.54)
12	**OLETS**	8.80 (0.09)	7.60 (0.13)	203.50 (5.41)	427.90 (7.59)	215.05 (35.03)	590.20 (100.27)	1452.65 (11.51)	1480.05 (10.90)
10	**OSLA**	7.55 (0.17)	6.20 (0.22)	103.60 (10.23)	180.95 (19.38)	226.50 (70.77)	582.55 (85.59)	912.20 (113.71)	910.30 (117.08)

Table C.48

KDH: *Escape*									
Points	**Controller**	**Sprites Ack.**	**Unique Int.**	**CC**	**CA**	**Game-ticks Ack.**	**Game-ticks Int.**	**Game-ticks CC**	**Game-ticks CA**
25	**RHEA**	4.00 (0)	3.00 (0)	19.30 (3.15)	0.00 (0)	0.00 (0)	140.45 (39.45)	140.45 (39.45)	0.00 (0)
18	**RS**	4.00 (0)	2.60 (0.11)	61.05 (1.22)	0.00 (0)	0.00 (0)	598.75 (108.31)	712.75 (79.62)	0.00 (0)
15	**OLMCTS**	4.00 (0)	2.55 (0.11)	18.85 (2.23)	0.00 (0)	0.00 (0)	553.50 (109.55)	958.65 (27.95)	0.00 (0)
12	**OSLA**	4.00 (0)	2.10 (0.07)	58.70 (1.19)	0.00 (0)	0.00 (0)	104.85 (66.73)	693.20 (41.57)	0.00 (0)
10	**OLETS**	4.00 (0)	2.00 (0)	54.40 (1.38)	0.00 (0)	0.00 (0)	3.75 (0.20)	436.75 (46.07)	0.00 (0)

Table C.49

KDH: ***Hungry Birds***									
Points	**Controller**	**Sprites Ack.**	**Unique Int.**	**CC**	**CA**	**Game-ticks Ack.**	**Game-ticks Int.**	**Game-ticks CC**	**Game-ticks CA**
25	**RHEA**	4.00 (0)	3.15 (0.16)	58.70 (2.12)	0.00 (0)	0.00 (0)	123.05 (22.14)	240.80 (6.61)	0.00 (0)
18	**RS**	4.00 (0)	2.75 (0.10)	63.20 (1.45)	0.00 (0)	0.00 (0)	49.00 (11.74)	302.60 (40.68)	0.00 (0)
15	**OLMCTS**	4.00 (0)	2.70 (0.10)	56.05 (2.61)	0.00 (0)	0.00 (0)	58.25 (16.42)	239.90 (15.91)	0.00 (0)
12	**OLETS**	4.00 (0)	2.30 (0.10)	71.65 (0.94)	0.00 (0)	0.00 (0)	21.45 (6.03)	250.35 (3.37)	0.00 (0)
10	**OSLA**	4.00 (0)	2.05 (0.05)	18.45 (1.73)	0.00 (0)	0.00 (0)	14.95 (11.85)	245.30 (17.07)	0.00 (0)

Table C.50

KDH: ***Infection***									
Points	**Controller**	**Sprites Ack.**	**Unique Int.**	**CC**	**CA**	**Game-ticks Ack.**	**Game-ticks Int.**	**Game-ticks CC**	**Game-ticks CA**
25	**RHEA**	7.00 (0)	5.25 (0.16)	59.25 (3.75)	5.60 (0.55)	0.00 (0)	369.25 (70.37)	994.05 (71.52)	659.75 (77.10)
18	**RS**	7.00 (0)	4.90 (0.16)	57.65 (3.37)	5.20 (0.56)	0.00 (0)	431.45 (77.25)	1015.30 (61.12)	592.25 (78.18)
15	**OLMCTS**	7.00 (0)	4.75 (0.21)	62.60 (3.82)	6.60 (0.57)	0.00 (0)	459.60 (96.45)	860.90 (73.18)	645.75 (68.26)
12	**OLETS**	7.00 (0)	4.65 (0.22)	57.40 (3.42)	6.80 (0.39)	0.00 (0)	655.55 (78.54)	1012.35 (76.66)	797.15 (65.27)
10	**OSLA**	7.00 (0)	4.45 (0.24)	34.35 (2.01)	4.30 (0.50)	1.80 (0.35)	553.85 (79.83)	1102.30 (70.18)	742.00 (81.36)

Table C.51

KDH: ***Intersection***									
Points	**Controller**	**Sprites Ack.**	**Unique Int.**	**CC**	**CA**	**Game-ticks Ack.**	**Game-ticks Int.**	**Game-ticks CC**	**Game-ticks CA**
25	**RS**	6.00 (0)	5.60 (0.11)	87.30 (4.15)	0.00 (0)	0.00 (0)	317.60 (59.30)	783.30 (40.30)	0.00 (0)
18	**OLETS**	6.00 (0)	5.40 (0.15)	74.20 (3.58)	0.00 (0)	0.00 (0)	303.30 (67.48)	856.50 (30.22)	0.00 (0)
15	**OSLA**	6.00 (0)	4.90 (0.17)	65.25 (2.88)	0.00 (0)	0.00 (0)	325.05 (69.06)	947.70 (15.49)	0.00 (0)
12	**RHEA**	6.00 (0)	4.90 (0.07)	35.60 (1.42)	0.00 (0)	0.00 (0)	79.95 (18.18)	191.40 (15.71)	0.00 (0)
10	**OLMCTS**	6.00 (0)	4.90 (0.07)	35.35 (1.01)	0.00 (0)	0.00 (0)	46.65 (7.28)	658.60 (64.02)	0.00 (0)

Table C.52

KDH: *Lemmings*									
Points	**Controller**	**Sprites Ack.**	**Unique Int.**	**CC**	**CA**	**Game-ticks Ack.**	**Game-ticks Int.**	**Game-ticks CC**	**Game-ticks CA**
25	**RHEA**	7.00 (0)	2.65 (0.11)	60.50 (5.46)	71.50 (6.58)	0.00 (0)	450.95 (103.82)	926.55 (104.78)	918.85 (106.27)
18	**RS**	7.00 (0)	2.25 (0.10)	86.85 (3.03)	97.50 (2.82)	0.00 (0)	378.65 (144.10)	1281.75 (45.33)	1235.45 (44.34)
15	**OLMCTS**	7.00 (0)	2.20 (0.09)	85.30 (1.77)	105.45 (2.12)	0.00 (0)	301.85 (133.29)	1101.75 (63.60)	1078.70 (47.40)
12	**OLETS**	7.00 (0)	2.00 (0)	100.65 (2.02)	110.85 (2.13)	0.00 (0)	4.25 (0.17)	998.20 (48.19)	1010.20 (47.07)
10	**OSLA**	7.00 (0)	2.00 (0)	47.00 (2.40)	57.90 (3.15)	0.00 (0)	7.90 (1.85)	1317.30 (39.57)	1319.95 (39.87)

Table C.53

KDH: *Missile Command*									
Points	**Controller**	**Sprites Ack.**	**Unique Int.**	**CC**	**CA**	**Game-ticks Ack.**	**Game-ticks Int.**	**Game-ticks CC**	**Game-ticks CA**
25	**OLMCTS**	5.00 (0)	1.65 (0.11)	7.15 (1.29)	1.50 (0.21)	0.00 (0)	74.75 (8.73)	134.10 (18.91)	67.10 (12.62)
18	**RS**	5.00 (0)	1.65 (0.13)	5.40 (1.04)	1.35 (0.18)	0.00 (0)	82.50 (11.13)	120.65 (19.30)	73.05 (12.13)
15	**OLETS**	5.00 (0)	1.55 (0.11)	3.90 (0.77)	0.95 (0.11)	0.00 (0)	72.15 (7.92)	100.10 (19.27)	57.10 (8.97)
12	**RHEA**	5.00 (0)	1.50 (0.11)	3.60 (0.82)	1.00 (0.12)	0.00 (0)	78.15 (12.10)	97.35 (19.11)	57.00 (9.19)
10	**OSLA**	5.00 (0)	1.10 (0.14)	0.90 (0.28)	0.75 (0.12)	0.00 (0)	77.25 (15.69)	61.05 (20.08)	39.75 (7.24)

Table C.54

KDH: *Modality*									
Points	**Controller**	**Sprites Ack.**	**Unique Int.**	**CC**	**CA**	**Game-ticks Ack.**	**Game-ticks Int.**	**Game-ticks CC**	**Game-ticks CA**
25	**RS**	7.00 (0)	5.00 (0)	23.10 (0.07)	0.00 (0)	0.00 (0)	10.50 (0.89)	62.25 (2.31)	0.00 (0)
18	**OLMCTS**	7.00 (0)	5.00 (0)	23.00 (0)	0.00 (0)	1.30 (0.32)	27.70 (5.08)	147.90 (56.72)	0.00 (0)
15	**RHEA**	7.00 (0)	5.00 (0)	20.60 (0.94)	0.00 (0)	3.55 (0.78)	81.65 (10.91)	283.45 (33.38)	0.00 (0)
12	**OLETS**	6.95 (0.05)	5.00 (0)	23.25 (0.10)	0.00 (0)	9.90 (1.70)	7.20 (0.94)	57.30 (1.04)	0.00 (0)
10	**OSLA**	6.70 (0.10)	5.00 (0)	23.70 (0.10)	0.00 (0)	53.55 (12.63)	23.35 (3.49)	288.20 (47.55)	0.00 (0)

Table C.55

KDH: *Plaque Attack*									
Points	**Controller**	**Sprites Ack.**	**Unique Int.**	**CC**	**CA**	**Game-ticks Ack.**	**Game-ticks Int.**	**Game-ticks CC**	**Game-ticks CA**
25	**RS**	6.00 (0)	3.40 (0.11)	16.45 (1.05)	46.15 (2.66)	217.95 (5.59)	224.75 (34.38)	466.65 (37.69)	501.70 (38.25)
18	**OSLA**	6.00 (0)	3.35 (0.11)	5.65 (0.67)	38.65 (2.08)	247.65 (10.09)	222.15 (26.86)	371.45 (41.65)	478.80 (36.81)
15	**OLETS**	6.00 (0)	3.30 (0.10)	16.15 (0.83)	45.60 (2.67)	234.40 (13.94)	206.75 (30.50)	481.80 (33.55)	514.50 (34.65)
12	**RHEA**	6.00 (0)	3.30 (0.10)	14.15 (0.83)	45.25 (1.85)	230.50 (6.54)	206.15 (30.63)	461.45 (23.40)	511.60 (24.61)
10	**OLMCTS**	6.00 (0)	3.15 (0.08)	20.00 (1.66)	48.70 (2.67)	219.50 (5.16)	135.65 (17.68)	435.90 (34.62)	476.60 (33.30)

Table C.56

KDH: *Roguelike*									
Points	**Controller**	**Sprites Ack.**	**Unique Int.**	**CC**	**CA**	**Game-ticks Ack.**	**Game-ticks Int.**	**Game-ticks CC**	**Game-ticks CA**
25	**OLMCTS**	10.05 (0.05)	3.05 (0.15)	16.80 (1.60)	0.00 (0)	40.10 (39.08)	462.35 (80.56)	462.35 (80.56)	0.00 (0)
18	**RS**	10.05 (0.05)	3.00 (0.27)	15.40 (1.62)	0.30 (0.29)	9.85 (9.60)	428.85 (71.06)	428.85 (71.06)	48.55 (47.32)
15	**RHEA**	10.00 (0)	3.00 (0.12)	14.15 (0.83)	0.00 (0)	0.00 (0)	330.90 (56.10)	330.90 (56.10)	0.00 (0)
12	**OLETS**	10.00 (0)	2.85 (0.18)	15.50 (1.91)	0.00 (0)	0.00 (0)	397.80 (45.78)	397.80 (45.78)	0.00 (0)
10	**OSLA**	10.00 (0)	2.20 (0.09)	9.25 (0.56)	0.00 (0)	0.00 (0)	354.55 (29.34)	354.60 (29.32)	0.00 (0)

Table C.57

KDH: *Seaquest*									
Points	**Controller**	**Sprites Ack.**	**Unique Int.**	**CC**	**CA**	**Game-ticks Ack.**	**Game-ticks Int.**	**Game-ticks CC**	**Game-ticks CA**
25	**RS**	10.00 (0)	5.05 (0.09)	21.30 (1.25)	29.55 (2.49)	0.00 (0)	360.90 (35.93)	678.90 (51.71)	716.30 (58.39)
18	**OLMCTS**	10.00 (0)	4.90 (0.17)	23.60 (1.61)	36.90 (2.82)	2.15 (0.51)	314.75 (41.03)	678.70 (56.08)	788.00 (55.86)
15	**OLETS**	10.00 (0)	4.90 (0.12)	23.55 (1.57)	31.00 (2.63)	4.05 (1.07)	340.40 (35.91)	809.70 (53.23)	854.60 (59.24)
12	**OSLA**	10.00 (0)	4.75 (0.16)	12.25 (1.98)	15.75 (1.12)	92.40 (16.31)	397.15 (35.63)	545.45 (52.77)	590.20 (45.10)
10	**RHEA**	10.00 (0)	4.35 (0.37)	12.15 (2.34)	17.15 (3.09)	3.05 (0.92)	340.45 (50.25)	431.30 (68.97)	437.90 (74.13)

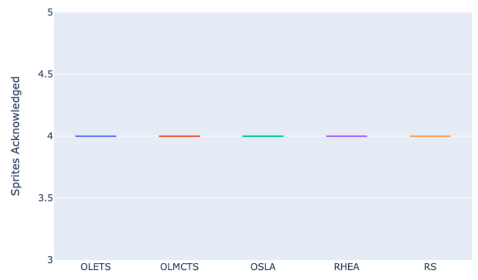
Table C.58

KDH: *Survive Zombies*									
Points	**Controller**	**Sprites Ack.**	**Unique Int.**	**CC**	**CA**	**Game-ticks Ack.**	**Game-ticks Int.**	**Game-ticks CC**	**Game-ticks CA**
25	**RHEA**	7.00 (0)	3.45 (0.11)	74.25 (7.77)	0.00 (0)	0.00 (0)	150.85 (44.64)	436.70 (72.82)	0.00 (0)
18	**OLMCTS**	7.00 (0)	3.05 (0.05)	107.65 (4.23)	0.00 (0)	0.00 (0)	45.10 (8.97)	783.45 (57.27)	0.00 (0)
15	**RS**	7.00 (0)	3.00 (0)	113.05 (6.83)	0.00 (0)	0.00 (0)	29.85 (3.89)	803.00 (66.91)	0.00 (0)
12	**OLETS**	7.00 (0)	3.00 (0)	105.45 (5.60)	0.00 (0)	0.00 (0)	99.85 (13.99)	711.55 (62.98)	0.00 (0)
10	**OSLA**	7.00 (0)	3.00 (0)	83.45 (5.07)	0.00 (0)	4.25 (0.99)	149.95 (24.81)	769.70 (47.88)	0.00 (0)

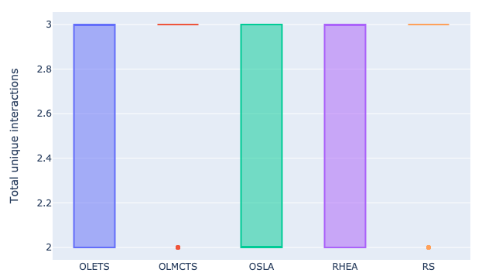
Table C.59

KDH: *Wait for Breakfast*									
Points	**Controller**	**Sprites Ack.**	**Unique Int.**	**CC**	**CA**	**Game-ticks Ack.**	**Game-ticks Int.**	**Game-ticks CC**	**Game-ticks CA**
25	**RS**	14.00 (0)	4.15 (0.15)	62.65 (9.70)	0.00 (0)	180.15 (52.60)	778.55 (77.10)	1154.35 (51.46)	0.00 (0)
18	**OLETS**	14.00 (0)	3.45 (0.21)	62.50 (10.76)	0.00 (0)	101.90 (32.52)	222.05 (89.65)	445.50 (87.29)	0.00 (0)
15	**OSLA**	14.00 (0)	3.30 (0.24)	30.45 (1.32)	0.00 (0)	110.10 (36.89)	461.10 (139.12)	573.30 (139.72)	0.00 (0)
12	**OLMCTS**	13.50 (0.19)	3.20 (0.26)	88.45 (12.39)	0.00 (0)	209.15 (44.07)	543.80 (126.52)	895.85 (138.08)	0.00 (0)
10	**RHEA**	13.20 (0.21)	2.35 (0.25)	8.65 (2.70)	0.00 (0)	22.65 (5.32)	132.65 (47.31)	204.35 (81.53)	0.00 (0)

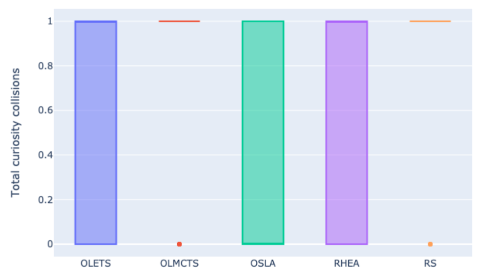
Table C.60



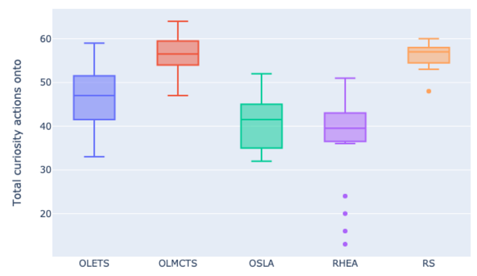
(a) *Sprites acknowledged*



(b) *Unique interactions*

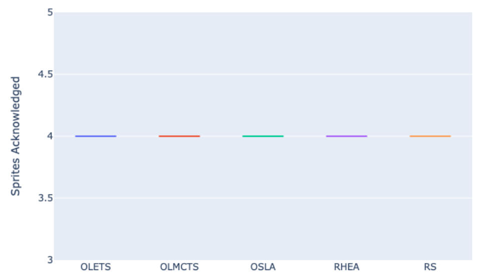


(c) *Curiosity collisions*

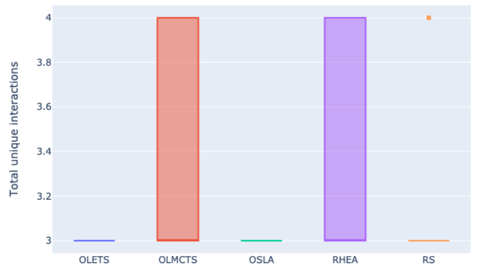


(d) *Curiosity actions-onto*

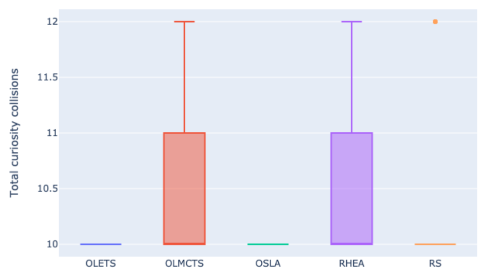
Figure C.41: KDH *Aliens* graphs



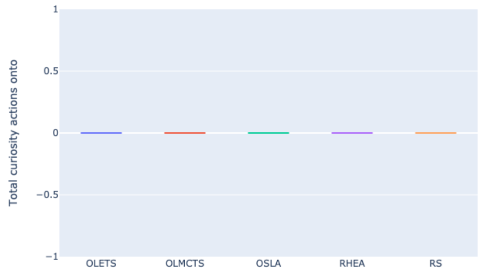
(a) *Sprites acknowledged*



(b) *Unique interactions*

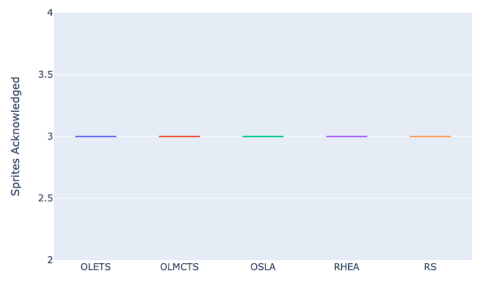


(c) *Curiosity collisions*

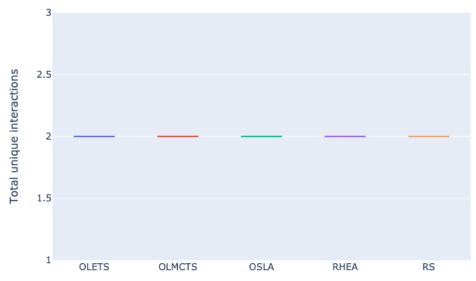


(d) *Curiosity actions-onto*

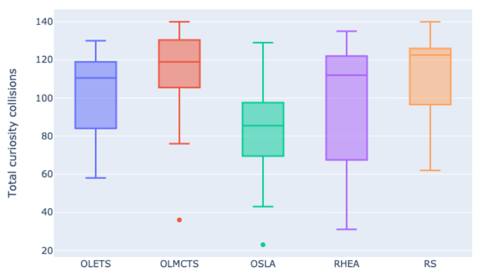
Figure C.42: KDH *Bait* graphs



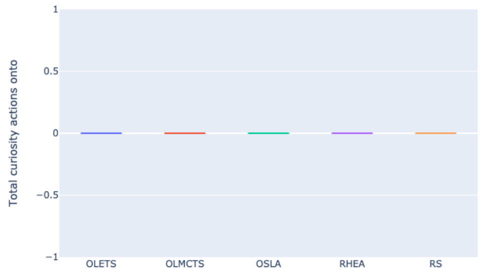
(a) *Sprites acknowledged*



(b) *Unique interactions*

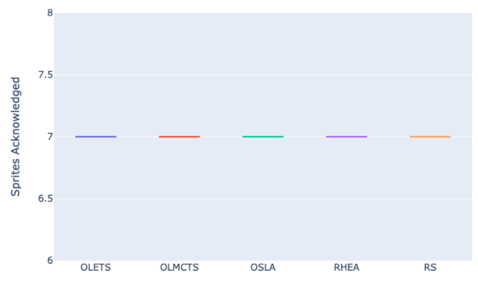


(c) *Curiosity collisions*

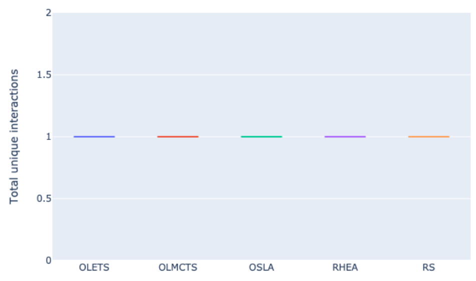


(d) *Curiosity actions-onto*

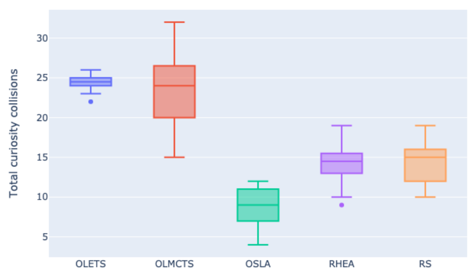
Figure C.43: KDH *Butterflies* graphs



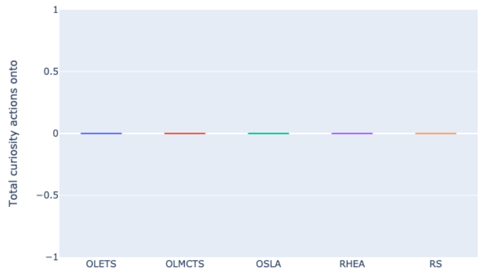
(a) *Sprites acknowledged*



(b) *Unique interactions*

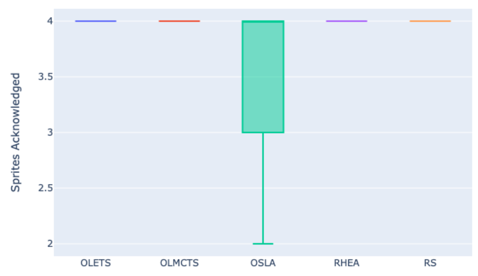


(c) *Curiosity collisions*

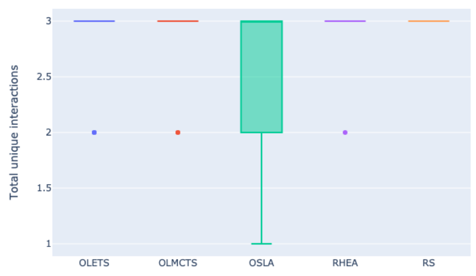


(d) *Curiosity actions-onto*

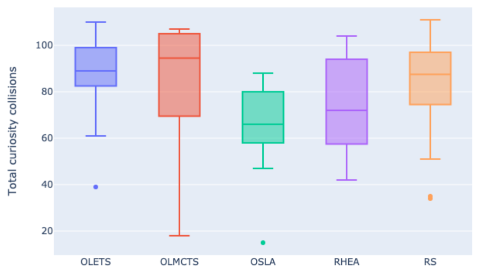
Figure C.44: KDH *Camel Race* graphs



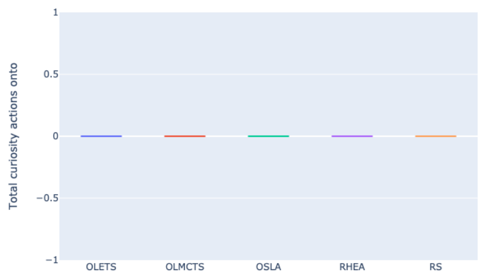
(a) *Sprites acknowledged*



(b) *Unique interactions*

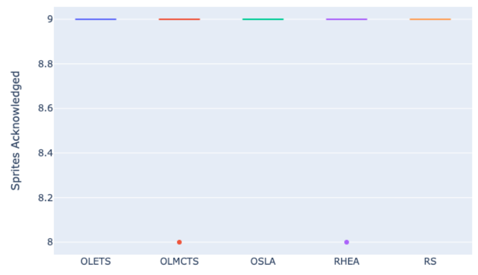


(c) *Curiosity collisions*

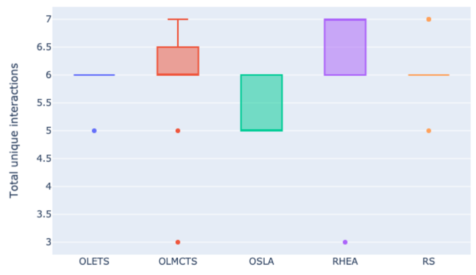


(d) *Curiosity actions-onto*

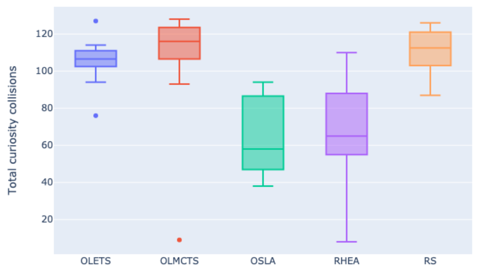
Figure C.45: KDH *Chase* graphs



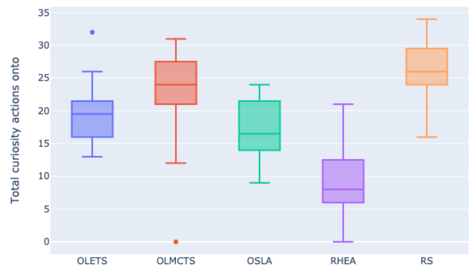
(a) *Sprites acknowledged*



(b) *Unique interactions*

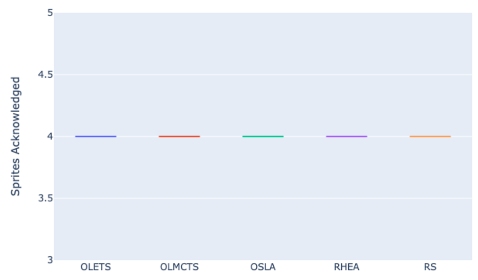


(c) *Curiosity collisions*

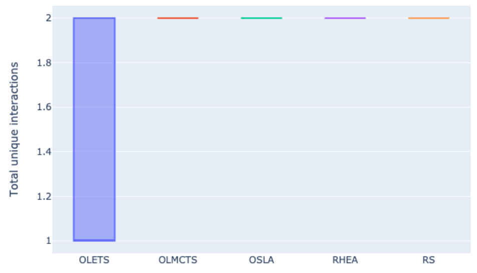


(d) *Curiosity actions-onto*

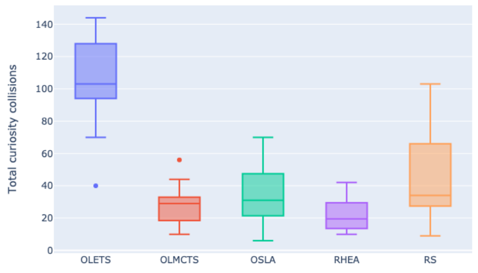
Figure C.46: KDH *Chopper* graphs



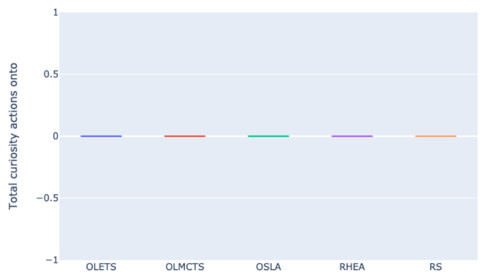
(a) *Sprites acknowledged*



(b) *Unique interactions*

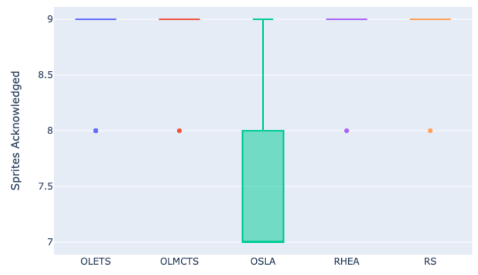


(c) *Curiosity collisions*

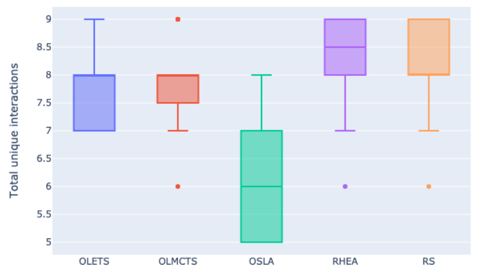


(d) *Curiosity actions-onto*

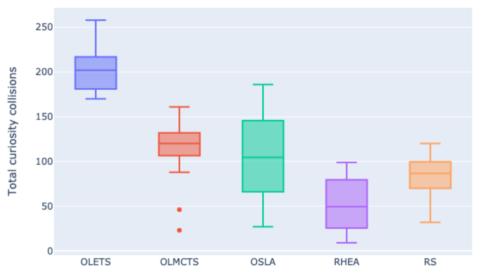
Figure C.47: KDH *Crossfire* graphs



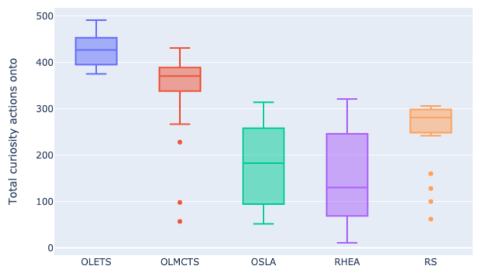
(a) *Sprites acknowledged*



(b) *Unique interactions*

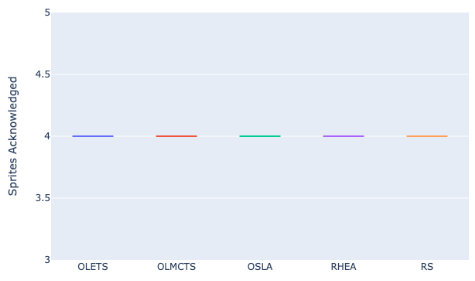


(c) *Curiosity collisions*

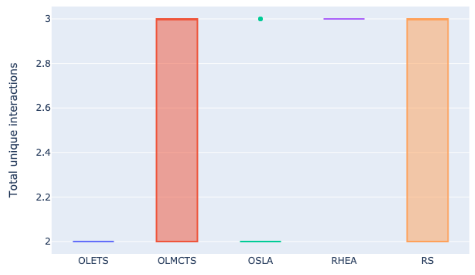


(d) *Curiosity actions-onto*

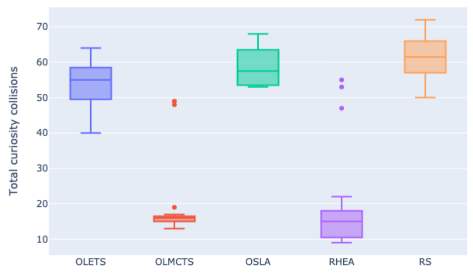
Figure C.48: KDH *Digdug* graphs



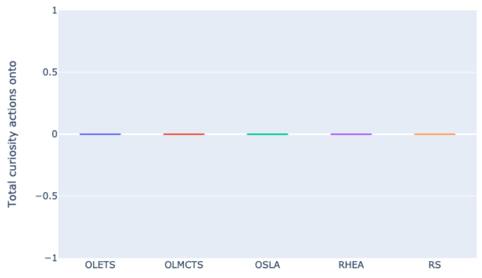
(a) *Sprites acknowledged*



(b) *Unique interactions*

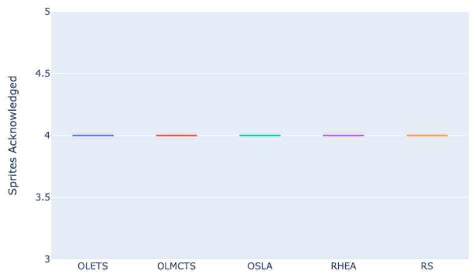


(c) *Curiosity collisions*

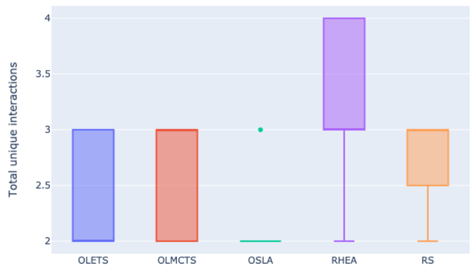


(d) *Curiosity actions-onto*

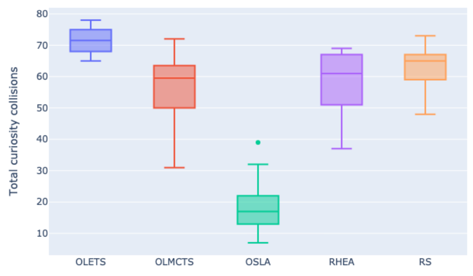
Figure C.49: KDH *Escape* graphs



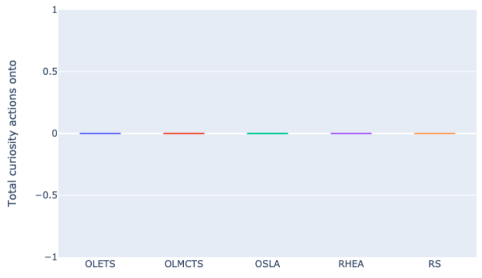
(a) *Sprites acknowledged*



(b) *Unique interactions*

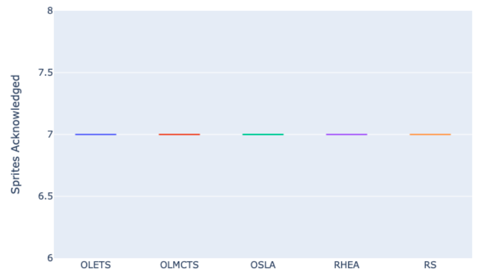


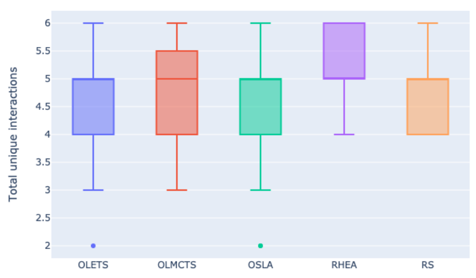
(c) *Curiosity collisions*

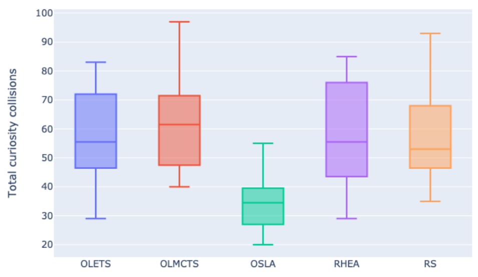


(d) *Curiosity actions-onto*

Figure C.50: KDH *Hungry Birds* graphs


(a) *Sprites acknowledged*


(b) *Unique interactions*


(c) *Curiosity collisions*

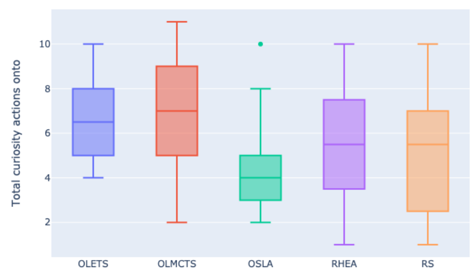
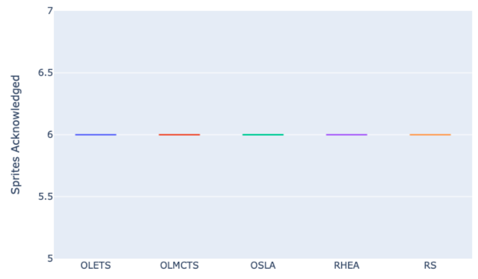
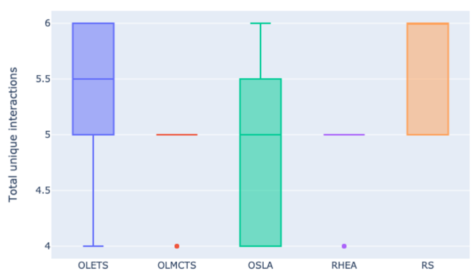
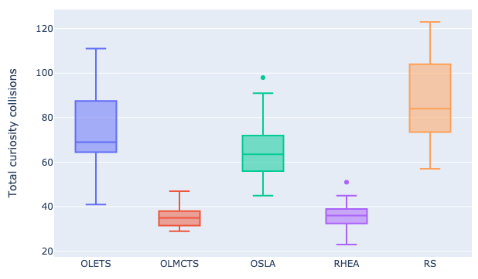

(d) *Curiosity actions-onto*

Figure C.51: KDH *Infection* graphs


(a) *Sprites acknowledged*


(b) *Unique interactions*


(c) *Curiosity collisions*

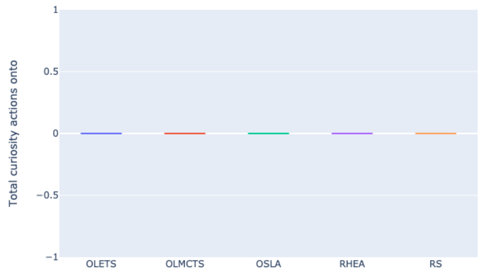
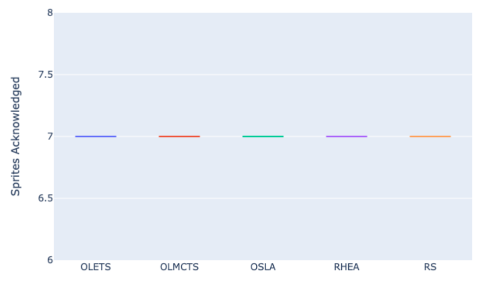
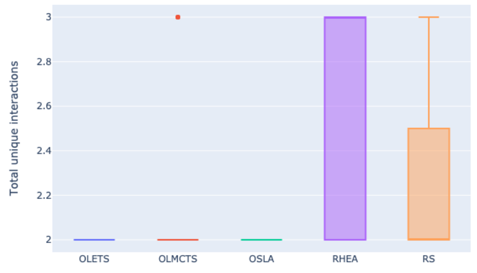

(d) *Curiosity actions-onto*

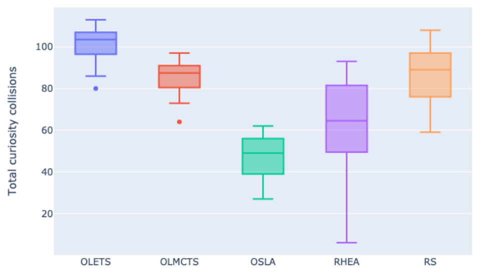
Figure C.52: KDH *Intersection* graphs



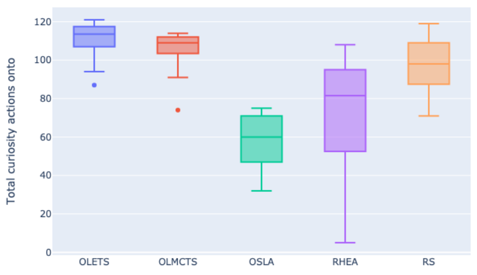
(a) *Sprites acknowledged*



(b) *Unique interactions*

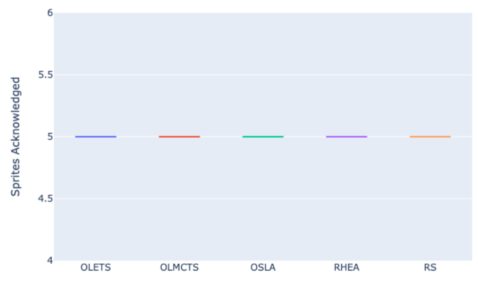


(c) *Curiosity collisions*

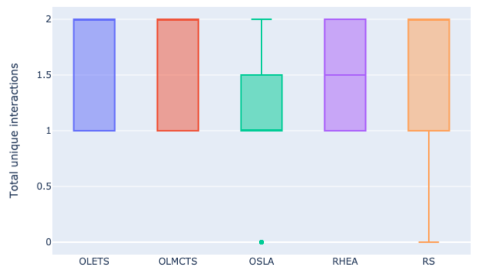


(d) *Curiosity actions-onto*

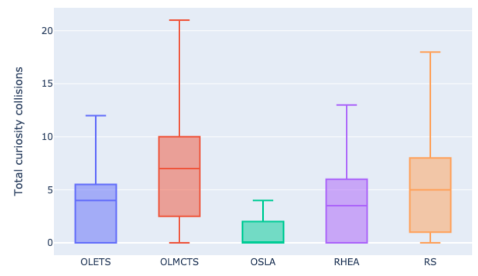
Figure C.53: KDH *Lemmings* graphs



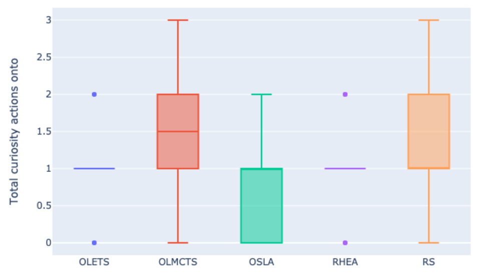
(a) *Sprites acknowledged*



(b) *Unique interactions*

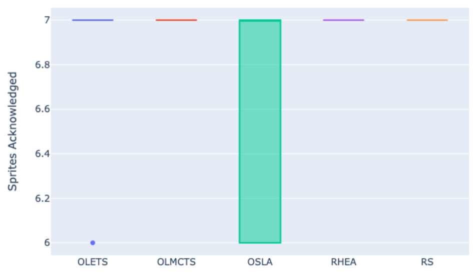


(c) *Curiosity collisions*

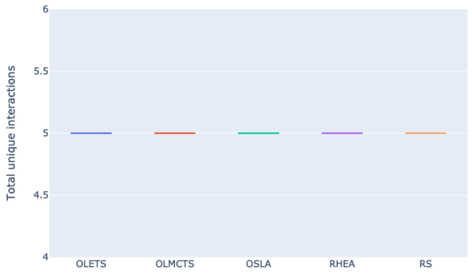


(d) *Curiosity actions-onto*

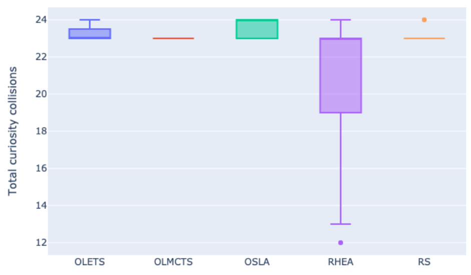
Figure C.54: KDH *Missile Command* graphs



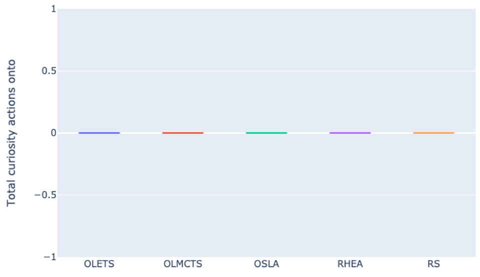
(a) *Sprites acknowledged*



(b) *Unique interactions*

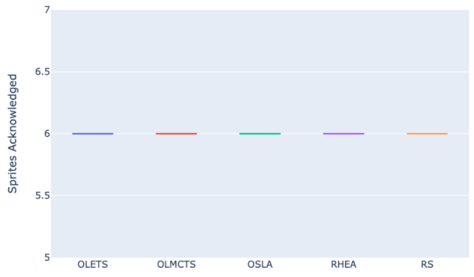


(c) *Curiosity collisions*

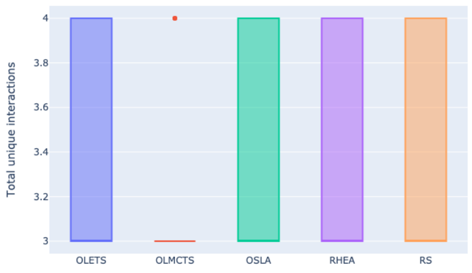


(d) *Curiosity actions-onto*

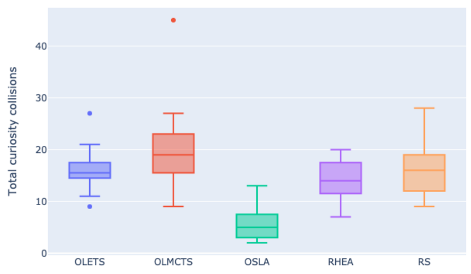
Figure C.55: KDH *Modality* graphs



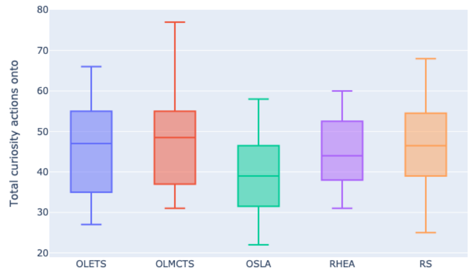
(a) *Sprites acknowledged*



(b) *Unique interactions*

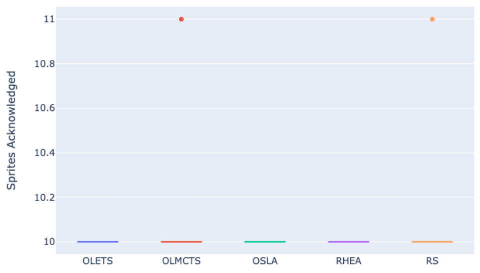


(c) *Curiosity collisions*

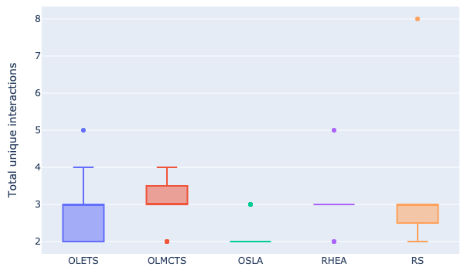


(d) *Curiosity actions-onto*

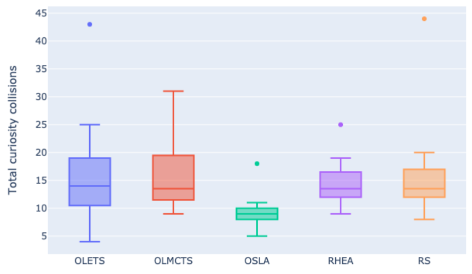
Figure C.56: KDH *Plaque Attack* graphs



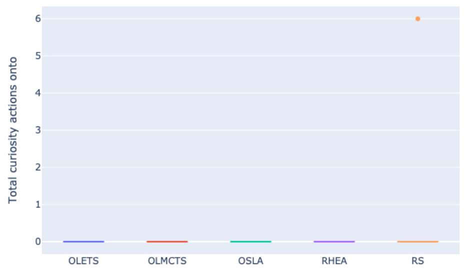
(a) *Sprites acknowledged*



(b) *Unique interactions*

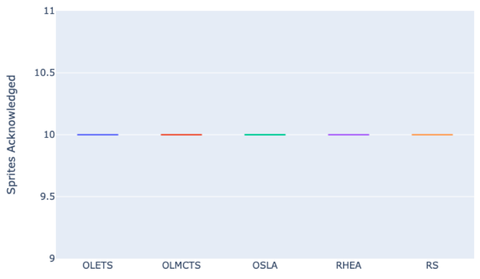


(c) *Curiosity collisions*

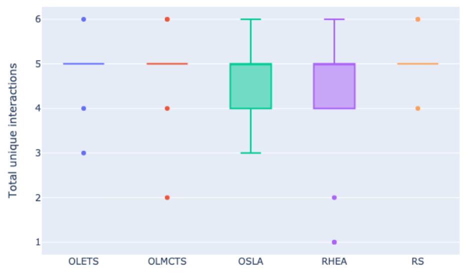


(d) *Curiosity actions-onto*

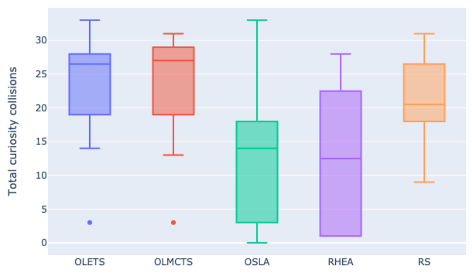
Figure C.57: KDH *Roguelike* graphs



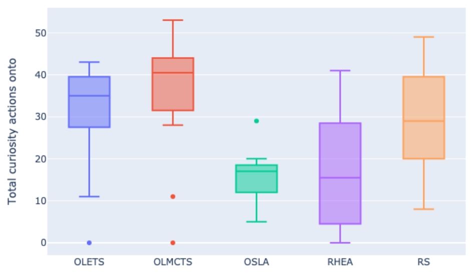
(a) *Sprites acknowledged*



(b) *Unique interactions*

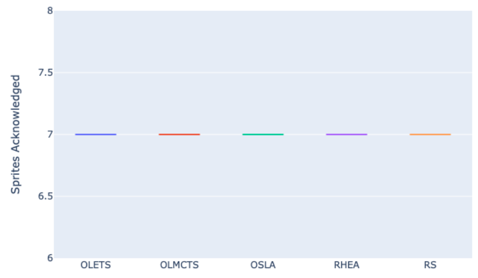


(c) *Curiosity collisions*

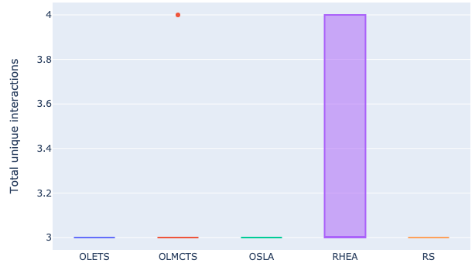


(d) *Curiosity actions-onto*

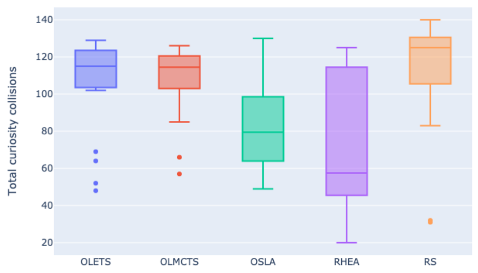
Figure C.58: KDH *Seaquest* graphs



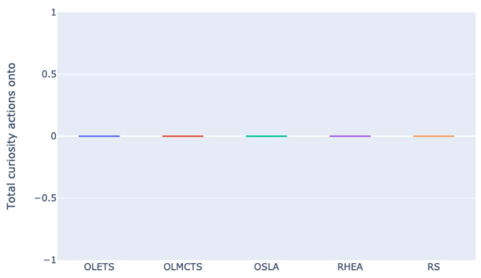
(a) *Sprites acknowledged*



(b) *Unique interactions*

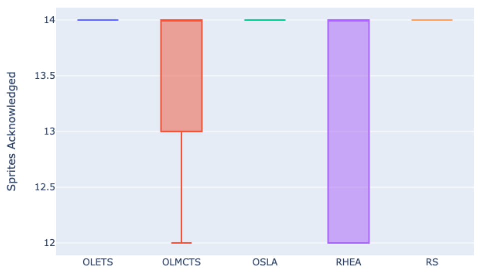


(c) *Curiosity collisions*

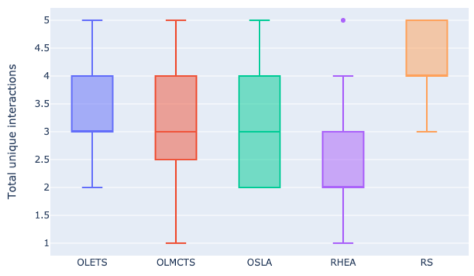


(d) *Curiosity actions-onto*

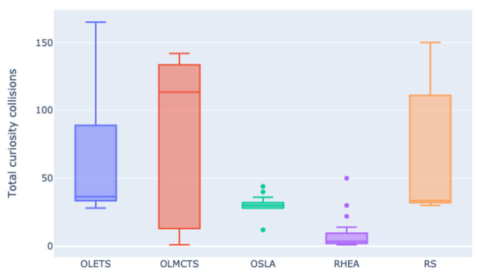
Figure C.59: KDH *Survive Zombies* graphs



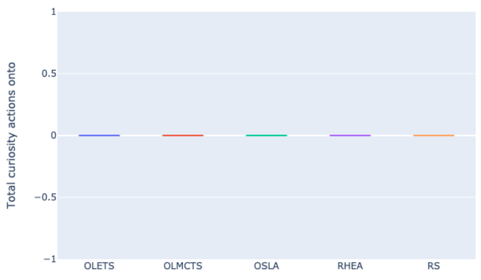
(a) *Sprites acknowledged*



(b) *Unique interactions*



(c) *Curiosity collisions*



(d) *Curiosity actions-onto*

Figure C.60: KDH *Wait for Breakfast* graphs

C.4 Per Game Results: KEH

Each table contains the KEH results of one of the games displaying the information listed below. The data is the final average resulting from the 20 gameplays of each agent.

- Total number of points received in the game (Points)
- Agent (Controller)
- Average of the square error obtained overall estimations (Avg. Square Error)
- Total number of interactions estimated (Interactions Estimated)

The games are included in alphabetic order.

KEH: *Aliens*			
Points	**Controller**	**Avg. Square Error**	**Interactions Estimated**
25	**RS**	1.11E-01	4
18	**OLMCTS**	1.18E-01	4
15	**RHEA**	1.20E-01	4
12	**OSLA**	1.21E-01	4
10	**OLETS**	1.25E-01	4

Table C.61

KEH: *Bait*			
Points	**Controller**	**Avg. Square Error**	**Interactions Estimated**
25	**RHEA**	2.04E+00	4
18	**RS**	2.43E+00	4
15	**OLMCTS**	2.54E+00	4
12	**OLETS**	3.25E+00	3
10	**OSLA**	4.33E+00	2

Table C.62

KEH: *Butterflies*			
Points	**Controller**	**Avg. Square Error**	**Interactions Estimated**
25	**RS**	2.38E-01	2
18	**RHEA**	2.41E-01	2
15	**OLETS**	2.50E-01	2
12	**OSLA**	2.52E-01	2
10	**OLMCTS**	2.53E-01	2

Table C.63

KEH: ***Camel Race***			
Points	**Controller**	**Avg. Square Error**	**Interactions Estimated**
25	**OLETS**	2.25E-05	1
18	**OSLA**	6.15E-05	1
15	**RHEA**	3.38E-04	1
12	**RS**	4.66E-04	1
10	**OLMCTS**	8.49E-04	1

Table C.64

KEH: ***Chase***			
Points	**Controller**	**Avg. Square Error**	**Interactions Estimated**
25	**OLMCTS**	1.67E-01	3
18	**RS**	1.67E-01	3
15	**RHEA**	1.67E-01	3
12	**OLETS**	5.00E-01	1
10	**OSLA**	5.00E-01	1

Table C.65

KEH: ***Chopper***			
Points	**Controller**	**Avg. Square Error**	**Interactions Estimated**
25	**RHEA**	7.26E-02	7
18	**OLMCTS**	7.27E-02	7
15	**OLETS**	7.50E-02	7
12	**RS**	7.51E-02	7
10	**OSLA**	8.69E-02	6

Table C.66

KEH: ***Crossfire***			
Points	**Controller**	**Avg. Square Error**	**Interactions Estimated**
25	**OLMCTS**	8.30E-06	3
18	**OSLA**	4.33E+00	2
15	**RS**	4.33E+00	2
12	**RHEA**	4.33E+00	2
10	**OLETS**	4.33E+00	2

Table C.67

KEH: ***Digdug***			
Points	**Controller**	**Avg. Square Error**	**Interactions Estimated**
25	**OSLA**	1.50E-01	10
18	**OLETS**	1.51E-01	10
15	**OLMCTS**	1.52E-01	10
12	**RS**	1.53E-01	10
10	**RHEA**	1.55E-01	10

Table C.68

KEH: ***Escape***			
Points	**Controller**	**Avg. Square Error**	**Interactions Estimated**
25	**RHEA**	0.00E+00	4
18	**OLETS**	6.71E-08	4
15	**RS**	5.81E-07	4
12	**OLMCTS**	2.50E-01	3
10	**OSLA**	5.00E-01	1

Table C.69

KEH: ***Hungry Birds***			
Points	**Controller**	**Avg. Square Error**	**Interactions Estimated**
25	**OLETS**	1.40E-10	2
18	**OLMCTS**	1.50E-10	2
15	**RS**	3.27E-10	2
12	**RHEA**	3.32E-10	2
10	**OSLA**	7.47E-06	1

Table C.70

KEH: ***Infection***			
Points	**Controller**	**Avg. Square Error**	**Interactions Estimated**
25	**OLMCTS**	8.33E-02	6
18	**OLETS**	8.60E-02	6
15	**RHEA**	8.82E-02	6
12	**RS**	9.07E-02	6
10	**OSLA**	9.14E-02	5

Table C.71

KEH: ***Intersection***			
Points	**Controller**	**Avg. Square Error**	**Interactions Estimated**
25	**OSLA**	1.40E-07	2
18	**RS**	2.17E+00	6
15	**OLMCTS**	2.17E+00	6
12	**RHEA**	2.19E+00	6
10	**OLETS**	1.06E+01	5

Table C.72

KEH: ***Lemmings***			
Points	**Controller**	**Avg. Square Error**	**Interactions Estimated**
25	**OSLA**	1.07E-05	3
18	**RHEA**	1.31E-05	3
15	**OLMCTS**	3.17E-05	3
12	**RS**	5.99E-05	3
10	**OLETS**	9.54E-05	3

Table C.73

KEH: ***Missile Command***			
Points	**Controller**	**Avg. Square Error**	**Interactions Estimated**
25	**RS**	1.70E-01	2
18	**OLMCTS**	1.85E-01	2
15	**RHEA**	1.91E-01	2
12	**OSLA**	2.50E-01	2
10	**OLETS**	1.25E+00	1

Table C.74

KEH: ***Modality***			
Points	**Controller**	**Avg. Square Error**	**Interactions Estimated**
25	**OSLA**	2.67E-08	2
18	**OLETS**	1.32E-07	5
15	**OLMCTS**	1.48E-06	5
12	**RS**	4.25E-03	5
10	**RHEA**	7.48E-03	5

Table C.75

KEH: ***Plaque Attack***			
Points	**Controller**	**Avg. Square Error**	**Interactions Estimated**
25	**OLMCTS**	2.15E-01	4
18	**RHEA**	2.57E-01	4
15	**RS**	2.91E-01	4
12	**OLETS**	7.09E-01	4
10	**OSLA**	1.29E+00	4

Table C.76

KEH: ***Roguelike***			
Points	**Controller**	**Avg. Square Error**	**Interactions Estimated**
25	**OLETS**	4.80E-09	3
18	**OSLA**	6.94E-08	3
15	**OLMCTS**	5.35E-03	5
12	**RHEA**	7.65E-03	5
10	**RS**	3.67E-02	6

Table C.77

KEH: ***Seaquest***			
Points	**Controller**	**Avg. Square Error**	**Interactions Estimated**
25	**OSLA**	2.75E-02	8
18	**RHEA**	4.34E-02	8
15	**RS**	4.78E-02	8
12	**OLETS**	1.68E-01	8
10	**OLMCTS**	3.99E-01	8

Table C.78

KEH: ***Survive Zombies***			
Points	**Controller**	**Avg. Square Error**	**Interactions Estimated**
25	**RHEA**	9.12E-02	4
18	**OLMCTS**	1.04E-01	4
15	**OLETS**	1.36E-01	4
12	**RS**	1.42E-01	4
10	**OSLA**	3.17E-01	3

Table C.79

KEH: ***Wait for Breakfast***			
Points	**Controller**	**Avg. Square Error**	**Interactions Estimated**
25	**OLMCTS**	4.01E-02	7
18	**OSLA**	8.05E-02	5
15	**RHEA**	9.13E-02	7
12	**RS**	9.69E-02	7
10	**OLETS**	1.20E-01	7

Table C.80

C.5 Per Game Estimations: KEH

Each table contains the resulting predictions of one game and agent displaying, for each of the sprites labelled with their corresponding *stype*, the information listed below. The final prediction is the average (and corresponding standard deviation) resulting from the 20 gameplays of the agent.

- ID of the sprite (Stype)
- Ground truth, estimation of the agent, and accuracy of the prediction for each of the possible types of interactions with that sprite:
  - collision win (CW)
  - collision score (CS)
  - action-onto win (AW)
  - action-onto score (AS)

The values in parenthesis represent the corresponding *Std. Deviation.* The games are included in alphabetical order, and for each of them is included 5 different tables, one per agent.

OSLA Predictions: *Aliens*												
Stype	**Ground Truth**				**Estimations**				**Accuracy Square Error**			
	CW	**CS**	**AW**	**AS**	**CW**	**CS**	**AW**	**AS**	**CW**	**CS**	**AW**	**AS**
9	-1	-1	1	2	-1.00 (0.00)	-0.93 (0.03)	0.02 (0.01)	2.00 (0.01)	0.00E+00	5.45E-03	9.61E-01	1.14E-05
3	-	-	0	1	-	-	-0.00 (0.00)	1.00 (0.00)	-	-	3.91E-07	3.91E-07
6	-1	-1	-	-	-1.00 (0.00)	-0.93 (0.09)	-	-	0.00E+00	4.44E-03	-	-

Table C.81

OLETS Predictions: *Aliens*												
Stype	**Ground Truth**				**Estimations**				**Accuracy Square Error**			
	CW	**CS**	**AW**	**AS**	**CW**	**CS**	**AW**	**AS**	**CW**	**CS**	**AW**	**AS**
9	-1	-1	1	2	-1.00 (0.00)	-0.81 (0.10)	0.02 (0.01)	1.99 (0.01)	0.00E+00	3.68E-02	9.62E-01	1.14E-04
3	-	-	0	1	-	-	-0.00 (0.00)	1.00 (0.00)	-	-	1.19E-09	1.19E-09
6	-1	-1	-	-	-1.00 (0.00)	-1.02 (0.02)	-	-	0.00E+00	2.46E-04	-	-

Table C.82

OLMCTS Predictions: *Aliens*												
Stype	**Ground Truth**				**Estimations**				**Accuracy Square Error**			
	CW	**CS**	**AW**	**AS**	**CW**	**CS**	**AW**	**AS**	**CW**	**CS**	**AW**	**AS**
9	-1	-1	1	2	-1.00 (0.00)	-0.92 (0.04)	0.03 (0.01)	2.00 (0.00)	0.00E+00	6.70E-03	9.36E-01	8.22E-06
3	-	-	0	1	-	-	-0.00 (0.00)	1.00 (0.00)	-	-	8.89E-09	8.89E-09
6	-1	-1	-	-	-1.00 (0.00)	-0.97 (0.02)	-	-	0.00E+00	6.41E-04	-	-

Table C.83

RHEA Predictions: *Aliens*												
Stype	**Ground Truth**				**Estimations**				**Accuracy Square Error**			
	CW	**CS**	**AW**	**AS**	**CW**	**CS**	**AW**	**AS**	**CW**	**CS**	**AW**	**AS**
9	-1	-1	1	2	-1.00 (0.00)	-0.89 (0.00)	0.03 (0.01)	2.00 (0.00)	0.00E+00	1.19E-02	9.47E-01	2.24E-07
3	-	-	0	1	-	-	0.00 (0.00)	1.00 (0.00)	-	-	4.26E-07	4.93E-06
6	-1	-1	-	-	-1.00 (0.00)	-0.95 (0.02)	-	-	0.00E+00	2.33E-03	-	-

Table C.84

RS Predictions: *Aliens*												
Stype	**Ground Truth**				**Estimations**				**Accuracy Square Error**			
	CW	**CS**	**AW**	**AS**	**CW**	**CS**	**AW**	**AS**	**CW**	**CS**	**AW**	**AS**
9	-1	-1	1	2	-1.00 (0.00)	-0.88 (0.02)	0.07 (0.00)	1.98 (0.00)	0.00E+00	1.36E-02	8.58E-01	3.47E-04
3	-	-	0	1	-	-	0.00 (0.00)	1.00 (0.00)	-	-	5.18E-09	6.92E-08
6	-1	-1	-	-	-1.00 (0.00)	-0.88 (0.07)	-	-	0.00E+00	1.51E-02	-	-

Table C.85

OSLA Predictions: *Bait*												
Stype	**Ground Truth**				**Estimations**				**Accuracy Square Error**			
	CW	**CS**	**AW**	**AS**	**CW**	**CS**	**AW**	**AS**	**CW**	**CS**	**AW**	**AS**
0	0	0	-	-	-0.00 (0.00)	0.00 (0.00)	-	-	6.12E-07	0.00E+00	-	-
8	1	5	-	-	-0.00 (0.00)	0.00 (0.00)	-	-	1.00E+00	2.50E+01	-	-

Table C.86

OLETS Predictions: *Bait*												
Stype	**Ground Truth**				**Estimations**				**Accuracy Square Error**			
	CW	**CS**	**AW**	**AS**	**CW**	**CS**	**AW**	**AS**	**CW**	**CS**	**AW**	**AS**
0	0	0	-	-	-0.00 (0.00)	0.00 (0.00)	-	-	1.11E-07	0.00E+00	-	-
8	1	5	-	-	-0.00 (0.00)	0.00 (0.00)	-	-	1.00E+00	2.50E+01	-	-
9	0	0	-	-	-0.00 (0.00)	0.00 (0.00)	-	-	1.18E-07	0.00E+00	-	-

Table C.87

OLMCTS Predictions: *Bait*												
Stype	**Ground Truth**				**Estimations**				**Accuracy Square Error**			
	CW	**CS**	**AW**	**AS**	**CW**	**CS**	**AW**	**AS**	**CW**	**CS**	**AW**	**AS**
0	0	0	-	-	-0.00 (0.00)	0.00 (0.00)	-	-	2.78E-07	0.00E+00	-	-
8	1	5	-	-	0.12 (0.06)	0.58 (0.31)	-	-	7.81E-01	1.95E+01	-	-
9	0	0	-	-	-0.00 (0.00)	0.00 (0.00)	-	-	8.88E-08	0.00E+00	-	-
7	0	0	-	-	0.00 (0.00)	0.00 (0.00)	-	-	0.00E+00	0.00E+00	-	-

Table C.88

RHEA Predictions: *Bait*												
Stype	**Ground Truth**				**Estimations**				**Accuracy Square Error**			
	CW	**CS**	**AW**	**AS**	**CW**	**CS**	**AW**	**AS**	**CW**	**CS**	**AW**	**AS**
0	0	0	-	-	-0.00 (0.00)	0.00 (0.00)	-	-	8.90E-08	0.00E+00	-	-
8	1	5	-	-	0.21 (0.07)	1.04 (0.35)	-	-	6.28E-01	1.57E+01	-	-
9	0	0	-	-	-0.00 (0.00)	0.00 (0.00)	-	-	1.40E-07	0.00E+00	-	-
7	0	0	-	-	0.00 (0.00)	0.00 (0.00)	-	-	0.00E+00	0.00E+00	-	-

Table C.89

RS Predictions: *Bait*												
Stype	**Ground Truth**				**Estimations**				**Accuracy Square Error**			
	CW	**CS**	**AW**	**AS**	**CW**	**CS**	**AW**	**AS**	**CW**	**CS**	**AW**	**AS**
0	0	0	-	-	-0.00 (0.00)	0.00 (0.00)	-	-	2.01E-07	0.00E+00	-	-
8	1	5	-	-	0.13 (0.04)	0.67 (0.21)	-	-	7.49E-01	1.87E+01	-	-
9	0	0	-	-	-0.00 (0.00)	0.00 (0.00)	-	-	7.07E-08	0.00E+00	-	-
7	0	0	-	-	0.00 (0.00)	0.00 (0.00)	-	-	0.00E+00	0.00E+00	-	-

Table C.90

OSLA Predictions: *Butterflies*												
Stype	**Ground Truth**				**Estimations**				**Accuracy Square Error**			
	CW	**CS**	**AW**	**AS**	**CW**	**CS**	**AW**	**AS**	**CW**	**CS**	**AW**	**AS**
0	0	0	-	-	-0.00 (0.00)	0.00 (0.00)	-	-	3.95E-06	2.66E-06	-	-
5	1	2	-	-	-0.00 (0.00)	2.05 (0.02)	-	-	1.00E+00	2.62E-03	-	-

Table C.91

OLETS Predictions: *Butterflies*												
Stype	**Ground Truth**				**Estimations**				**Accuracy Square Error**			
	CW	**CS**	**AW**	**AS**	**CW**	**CS**	**AW**	**AS**	**CW**	**CS**	**AW**	**AS**
0	0	0	-	-	-0.00 (0.00)	0.00 (0.00)	-	-	1.02E-06	1.81E-05	-	-
5	1	2	-	-	0.01 (0.01)	2.13 (0.03)	-	-	9.84E-01	1.60E-02	-	-

Table C.92

OLMCTS Predictions: *Butterflies*												
Stype	**Ground Truth**				**Estimations**				**Accuracy Square Error**			
	CW	**CS**	**AW**	**AS**	**CW**	**CS**	**AW**	**AS**	**CW**	**CS**	**AW**	**AS**
0	0	0	-	-	-0.00 (0.00)	0.01 (0.00)	-	-	1.10E-05	2.73E-05	-	-
5	1	2	-	-	-0.00 (0.00)	2.10 (0.02)	-	-	1.00E+00	9.18E-03	-	-

Table C.93

RHEA Predictions: *Butterflies*

Stype	Ground Truth				Estimations				Accuracy Square Error			
	CW	CS	AW	AS	CW	CS	AW	AS	CW	CS	AW	AS
0	0	0	-	-	-0.00 (0.00)	0.01 (0.00)	-	-	1.25E-05	1.20E-04	-	-
5	1	2	-	-	0.03 (0.02)	2.14 (0.03)	-	-	9.43E-01	1.97E-02	-	-

Table C.94

RS Predictions: *Butterflies*

Stype	Ground Truth				Estimations				Accuracy Square Error			
	CW	CS	AW	AS	CW	CS	AW	AS	CW	CS	AW	AS
0	0	0	-	-	-0.00 (0.00)	0.01 (0.00)	-	-	9.89E-06	5.23E-05	-	-
5	1	2	-	-	0.03 (0.01)	2.10 (0.01)	-	-	9.43E-01	9.96E-03	-	-

Table C.95

OSLA Predictions: *Camel Race*												
Stype	**Ground Truth**				**Estimations**				**Accuracy Square Error**			
	CW	**CS**	**AW**	**AS**	**CW**	**CS**	**AW**	**AS**	**CW**	**CS**	**AW**	**AS**
0	0	0	-	-	-0.01 (0.00)	-0.01 (0.00)	-	-	6.15E-05	6.15E-05	-	-

Table C.96

OLETS Predictions: *Camel Race*												
Stype	**Ground Truth**				**Estimations**				**Accuracy Square Error**			
	CW	**CS**	**AW**	**AS**	**CW**	**CS**	**AW**	**AS**	**CW**	**CS**	**AW**	**AS**
0	0	0	-	-	-0.00 (0.00)	-0.00 (0.00)	-	-	2.25E-05	2.25E-05	-	-

Table C.97

OLMCTS Predictions: *Camel Race*												
Stype	**Ground Truth**				**Estimations**				**Accuracy Square Error**			
	CW	**CS**	**AW**	**AS**	**CW**	**CS**	**AW**	**AS**	**CW**	**CS**	**AW**	**AS**
0	0	0	-	-	-0.03 (0.00)	-0.03 (0.00)	-	-	8.49E-04	8.49E-04	-	-

Table C.98

RHEA Predictions: *Camel Race*												
Stype	**Ground Truth**				**Estimations**				**Accuracy Square Error**			
	CW	**CS**	**AW**	**AS**	**CW**	**CS**	**AW**	**AS**	**CW**	**CS**	**AW**	**AS**
0	0	0	-	-	-0.02 (0.00)	-0.02 (0.00)	-	-	3.38E-04	3.38E-04	-	-

Table C.99

RS Predictions: *Camel Race*												
Stype	**Ground Truth**				**Estimations**				**Accuracy Square Error**			
	CW	**CS**	**AW**	**AS**	**CW**	**CS**	**AW**	**AS**	**CW**	**CS**	**AW**	**AS**
0	0	0	-	-	-0.02 (0.00)	-0.02 (0.00)	-	-	4.66E-04	4.66E-04	-	-

Table C.100

OSLA Predictions: *Chase*												
Stype	**Ground Truth**				**Estimations**				**Accuracy Square Error**			
	CW	**CS**	**AW**	**AS**	**CW**	**CS**	**AW**	**AS**	**CW**	**CS**	**AW**	**AS**
0	0	0	-	-	-0.00 (0.00)	0.00 (0.00)	-	-	2.08E-07	0.00E+00	-	-

Table C.101

OLETS Predictions: *Chase*												
Stype	**Ground Truth**				**Estimations**				**Accuracy Square Error**			
	CW	**CS**	**AW**	**AS**	**CW**	**CS**	**AW**	**AS**	**CW**	**CS**	**AW**	**AS**
0	0	0	-	-	-0.00 (0.00)	0.00 (0.00)	-	-	4.55E-08	0.00E+00	-	-

Table C.102

OLMCTS Predictions: *Chase*												
Stype	**Ground Truth**				**Estimations**				**Accuracy Square Error**			
	CW	**CS**	**AW**	**AS**	**CW**	**CS**	**AW**	**AS**	**CW**	**CS**	**AW**	**AS**
0	0	0	-	-	-0.00 (0.00)	-0.00 (0.00)	-	-	1.45E-06	1.81E-13	-	-
5	-1	-1	-	-	-1.00 (0.00)	-1.00 (0.00)	-	-	0.00E+00	0.00E+00	-	-
6	1	1	-	-	0.00 (0.00)	1.00 (0.00)	-	-	1.00E+00	0.00E+00	-	-

Table C.103

RHEA Predictions: *Chase*												
Stype	**Ground Truth**				**Estimations**				**Accuracy Square Error**			
	CW	**CS**	**AW**	**AS**	**CW**	**CS**	**AW**	**AS**	**CW**	**CS**	**AW**	**AS**
0	0	0	-	-	-0.00 (0.00)	-0.00 (0.00)	-	-	8.03E-07	5.00E-09	-	-
5	-1	-1	-	-	-1.00 (0.00)	-1.00 (0.00)	-	-	0.00E+00	0.00E+00	-	-
6	1	1	-	-	0.00 (0.00)	1.01 (0.00)	-	-	1.00E+00	4.19E-05	-	-

Table C.104

RS Predictions: *Chase*												
Stype	**Ground Truth**				**Estimations**				**Accuracy Square Error**			
	CW	**CS**	**AW**	**AS**	**CW**	**CS**	**AW**	**AS**	**CW**	**CS**	**AW**	**AS**
0	0	0	-	-	-0.00 (0.00)	-0.00 (0.00)	-	-	9.84E-07	1.33E-11	-	-
5	-1	-1	-	-	-1.00 (0.00)	-1.00 (0.00)	-	-	0.00E+00	0.00E+00	-	-
6	1	1	-	-	0.00 (0.00)	1.00 (0.00)	-	-	1.00E+00	9.25E-07	-	-

Table C.105

OSLA Predictions: *Chopper*												
Stype	**Ground Truth**				**Estimations**				**Accuracy Square Error**			
	CW	**CS**	**AW**	**AS**	**CW**	**CS**	**AW**	**AS**	**CW**	**CS**	**AW**	**AS**
8	-1	0	0	0	-1.00 (0.00)	-0.10 (0.05)	-0.15 (0.05)	-0.03 (0.01)	0.00E+00	9.25E-03	2.15E-02	8.00E-04
19	0	0	-	-	-0.01 (0.00)	-0.02 (0.00)	-	-	1.38E-04	4.24E-04	-	-
4	0	0	-	-	-0.00 (0.00)	-0.02 (0.00)	-	-	4.41E-06	3.35E-04	-	-
5	0	0	-	-	-0.01 (0.00)	-0.02 (0.00)	-	-	8.91E-05	2.43E-04	-	-
15	-	-	1	1	-	-	0.00 (0.00)	1.43 (0.17)	-	-	1.00E+00	1.84E-01

Table C.106

OLETS Predictions: *Chopper*												
Stype	**Ground Truth**				**Estimations**				**Accuracy Square Error**			
	CW	**CS**	**AW**	**AS**	**CW**	**CS**	**AW**	**AS**	**CW**	**CS**	**AW**	**AS**
4	0	0	-	-	-0.00 (0.00)	-0.02 (0.00)	-	-	1.67E-07	2.36E-04	-	-
5	0	0	-	-	-0.00 (0.00)	-0.01 (0.00)	-	-	5.42E-06	2.59E-05	-	-
8	-1	0	0	0	-1.00 (0.00)	-0.04 (0.01)	-0.13 (0.07)	-0.05 (0.01)	0.00E+00	1.48E-03	1.64E-02	2.21E-03
15	-	-	1	1	-	-	-0.01 (0.01)	1.04 (0.03)	-	-	1.03E+00	1.87E-03
18	0	0	-	-	-0.00 (0.00)	-0.00 (0.00)	-	-	2.21E-06	2.26E-05	-	-
19	0	0	-	-	-0.00 (0.00)	-0.02 (0.00)	-	-	1.35E-06	4.43E-04	-	-

Table C.107

OLMCTS Predictions: *Chopper*												
Stype	**Ground Truth**				**Estimations**				**Accuracy Square Error**			
	CW	**CS**	**AW**	**AS**	**CW**	**CS**	**AW**	**AS**	**CW**	**CS**	**AW**	**AS**
4	0	0	-	-	-0.00 (0.00)	-0.01 (0.00)	-	-	8.89E-07	1.14E-04	-	-
5	0	0	-	-	-0.01 (0.00)	-0.00 (0.00)	-	-	2.88E-05	2.50E-05	-	-
8	-1	0	0	0	-1.00 (0.00)	-0.02 (0.01)	-0.04 (0.01)	-0.04 (0.01)	0.00E+00	2.62E-04	2.02E-03	2.00E-03
15	-	-	1	1	-	-	-0.00 (0.00)	1.07 (0.02)	-	-	1.01E+00	4.40E-03
18	0	0	-	-	-0.00 (0.00)	-0.01 (0.00)	-	-	2.33E-06	1.19E-04	-	-
19	0	0	-	-	-0.00 (0.00)	-0.01 (0.00)	-	-	1.03E-05	1.34E-04	-	-

Table C.108

RHEA Predictions: *Chopper*												
Stype	**Ground Truth**				**Estimations**				**Accuracy Square Error**			
	CW	**CS**	**AW**	**AS**	**CW**	**CS**	**AW**	**AS**	**CW**	**CS**	**AW**	**AS**
4	0	0	-	-	-0.00 (0.00)	-0.01 (0.00)	-	-	1.82E-08	7.54E-05	-	-
5	0	0	-	-	-0.02 (0.01)	-0.00 (0.00)	-	-	2.97E-04	1.74E-05	-	-
8	-1	0	0	0	-1.00 (0.00)	-0.02 (0.00)	-0.11 (0.03)	-0.04 (0.01)	0.00E+00	2.51E-04	1.17E-02	1.51E-03
15	-	-	1	1	-	-	-0.00 (0.00)	1.03 (0.03)	-	-	1.00E+00	1.04E-03
18	0	0	-	-	-0.03 (0.02)	0.00 (0.00)	-	-	6.72E-04	2.33E-06	-	-
19	0	0	-	-	-0.01 (0.01)	-0.01 (0.00)	-	-	2.12E-04	1.00E-04	-	-

Table C.109

RS Predictions: *Chopper*												
Stype	**Ground Truth**				**Estimations**				**Accuracy Square Error**			
	CW	**CS**	**AW**	**AS**	**CW**	**CS**	**AW**	**AS**	**CW**	**CS**	**AW**	**AS**
4	0	0	-	-	-0.00 (0.00)	-0.01 (0.00)	-	-	4.12E-06	1.35E-04	-	-
5	0	0	-	-	-0.01 (0.00)	-0.01 (0.00)	-	-	1.03E-04	9.77E-05	-	-
8	-1	0	0	0	-1.00 (0.00)	-0.02 (0.00)	-0.11 (0.02)	-0.03 (0.01)	0.00E+00	4.80E-04	1.12E-02	1.17E-03
15	-	-	1	1	-	-	-0.02 (0.01)	1.08 (0.03)	-	-	1.03E+00	6.19E-03
18	0	0	-	-	-0.01 (0.00)	-0.01 (0.00)	-	-	3.34E-05	7.15E-05	-	-
19	0	0	-	-	-0.01 (0.00)	-0.01 (0.00)	-	-	5.61E-05	1.40E-04	-	-

Table C.110

OSLA Predictions: *Crossfire*												
Stype	**Ground Truth**				**Estimations**				**Accuracy Square Error**			
	CW	**CS**	**AW**	**AS**	**CW**	**CS**	**AW**	**AS**	**CW**	**CS**	**AW**	**AS**
0	0	0	-	-	-0.00 (0.00)	-0.00 (0.00)	-	-	3.73E-07	2.33E-08	-	-
4	-1	-1	-	-	-1.00 (0.00)	-1.00 (0.00)	-	-	0.00E+00	0.00E+00	-	-

Table C.111

OLETS Predictions: *Crossfire*												
Stype	**Ground Truth**				**Estimations**				**Accuracy Square Error**			
	CW	**CS**	**AW**	**AS**	**CW**	**CS**	**AW**	**AS**	**CW**	**CS**	**AW**	**AS**
0	0	0	-	-	-0.00 (0.00)	-0.00 (0.00)	-	-	5.66E-08	8.37E-10	-	-
4	-1	-1	-	-	-1.00 (0.00)	-1.03 (0.03)	-	-	0.00E+00	9.18E-04	-	-

Table C.112

OLMCTS Predictions: *Crossfire*												
Stype	**Ground Truth**				**Estimations**				**Accuracy Square Error**			
	CW	**CS**	**AW**	**AS**	**CW**	**CS**	**AW**	**AS**	**CW**	**CS**	**AW**	**AS**
0	0	0	-	-	-0.00 (0.00)	-0.00 (0.00)	-	-	4.09E-06	1.01E-06	-	-
4	-1	-1	-	-	-1.00 (0.00)	-1.01 (0.00)	-	-	0.00E+00	4.47E-05	-	-
6	1	5	-	-	1.00 (0.00)	5.00 (0.00)	-	-	0.00E+00	0.00E+00	-	-

Table C.113

RHEA Predictions: *Crossfire*												
Stype	**Ground Truth**				**Estimations**				**Accuracy Square Error**			
	CW	**CS**	**AW**	**AS**	**CW**	**CS**	**AW**	**AS**	**CW**	**CS**	**AW**	**AS**
0	0	0	-	-	-0.01 (0.00)	-0.01 (0.00)	-	-	7.66E-05	7.66E-05	-	-
4	-1	-1	-	-	-1.00 (0.00)	-1.01 (0.00)	-	-	0.00E+00	1.29E-04	-	-

Table C.114

RS Predictions: *Crossfire*												
Stype	**Ground Truth**				**Estimations**				**Accuracy Square Error**			
	CW	**CS**	**AW**	**AS**	**CW**	**CS**	**AW**	**AS**	**CW**	**CS**	**AW**	**AS**
0	0	0	-	-	-0.01 (0.00)	-0.00 (0.00)	-	-	2.98E-05	2.30E-05	-	-
4	-1	-1	-	-	-1.00 (0.00)	-1.01 (0.00)	-	-	0.00E+00	4.23E-05	-	-

Table C.115

OSLA Predictions: *Digdug*												
Stype	**Ground Truth**				**Estimations**				**Accuracy Square Error**			
	CW	**CS**	**AW**	**AS**	**CW**	**CS**	**AW**	**AS**	**CW**	**CS**	**AW**	**AS**
0	0	0	0	0	0.00 (0.00)	0.00 (0.00)	0.00 (0.00)	0.00 (0.00)	0.00E+00	0.00E+00	0.00E+00	0.00E+00
4	0	0	1	0	0.00 (0.00)	0.00 (0.00)	0.00 (0.00)	0.00 (0.00)	0.00E+00	0.00E+00	1.00E+00	0.00E+00
5	1	1	-	-	0.00 (0.00)	1.00 (0.00)	-	-	1.00E+00	0.00E+00	-	-
6	-	-	0	0	-	-	-0.01 (0.00)	-0.00 (0.00)	-	-	4.07E-05	1.76E-05
8	0	0	-	-	0.00 (0.00)	0.00 (0.00)	-	-	0.00E+00	0.00E+00	-	-
9	0	0	-	-	-0.00 (0.00)	0.00 (0.00)	-	-	1.86E-05	0.00E+00	-	-
11	-1	-1	-	-	-1.00 (0.00)	-1.00 (0.00)	-	-	0.00E+00	0.00E+00	-	-
12	0	0	-	-	-1.00 (0.00)	0.00 (0.00)	-	-	1.00E+00	0.00E+00	-	-

Table C.116

OLETS Predictions: *Digdug*												
Stype	**Ground Truth**				**Estimations**				**Accuracy Square Error**			
	CW	**CS**	**AW**	**AS**	**CW**	**CS**	**AW**	**AS**	**CW**	**CS**	**AW**	**AS**
0	0	0	0	0	-0.00 (0.00)	0.00 (0.00)	-0.00 (0.00)	0.00 (0.00)	5.00E-08	3.03E-07	2.62E-07	7.92E-08
4	0	0	1	0	-0.00 (0.00)	-0.00 (0.00)	-0.00 (0.00)	-0.00 (0.00)	4.20E-09	3.09E-10	1.00E+00	3.45E-08
5	1	1	-	-	-0.00 (0.00)	1.00 (0.00)	-	-	1.01E+00	1.05E-05	-	-
6	-	-	0	0	-	-	-0.00 (0.00)	0.01 (0.00)	-	-	8.68E-07	3.29E-05
8	0	0	-	-	0.00 (0.00)	0.03 (0.03)	-	-	0.00E+00	7.82E-04	-	-
9	0	0	-	-	-0.00 (0.00)	0.00 (0.00)	-	-	3.95E-08	3.30E-10	-	-
11	-1	-1	-	-	-1.00 (0.00)	-1.03 (0.02)	-	-	0.00E+00	8.57E-04	-	-
12	0	0	-	-	-1.00 (0.00)	-0.02 (0.01)	-	-	1.00E+00	4.55E-04	-	-

Table C.117

OLMCTS Predictions: *Digdug*												
Stype	**Ground Truth**				**Estimations**				**Accuracy Square Error**			
	CW	**CS**	**AW**	**AS**	**CW**	**CS**	**AW**	**AS**	**CW**	**CS**	**AW**	**AS**
0	0	0	0	0	-0.00 (0.00)	-0.00 (0.00)	-0.00 (0.00)	-0.00 (0.00)	4.50E-06	1.05E-06	2.58E-06	2.65E-07
4	0	0	1	0	-0.00 (0.00)	-0.00 (0.00)	-0.00 (0.00)	-0.00 (0.00)	8.45E-07	2.70E-07	1.00E+00	4.19E-07
5	1	1	-	-	-0.02 (0.00)	0.98 (0.00)	-	-	1.04E+00	3.63E-04	-	-
6	-	-	0	0	-	-	-0.00 (0.00)	0.00 (0.00)	-	-	1.27E-05	1.93E-06
8	0	0	-	-	-0.00 (0.00)	0.04 (0.02)	-	-	8.58E-06	2.02E-03	-	-
9	0	0	-	-	-0.00 (0.00)	0.00 (0.00)	-	-	1.59E-06	8.87E-06	-	-
11	-1	-1	-	-	-1.00 (0.00)	-0.99 (0.01)	-	-	0.00E+00	1.09E-04	-	-
12	0	0	-	-	-1.00 (0.00)	0.00 (0.00)	-	-	1.00E+00	4.06E-06	-	-

Table C.118

RHEA Predictions: *Digdug*												
Stype	**Ground Truth**				**Estimations**				**Accuracy Square Error**			
	CW	**CS**	**AW**	**AS**	**CW**	**CS**	**AW**	**AS**	**CW**	**CS**	**AW**	**AS**
0	0	0	0	0	-0.00 (0.00)	-0.00 (0.00)	-0.00 (0.00)	-0.00 (0.00)	9.77E-06	7.12E-06	1.39E-05	9.13E-06
4	0	0	1	0	-0.00 (0.00)	-0.00 (0.00)	-0.00 (0.00)	0.00 (0.00)	2.05E-05	7.54E-06	1.00E+00	1.71E-08
5	1	1	-	-	-0.04 (0.01)	0.96 (0.01)	-	-	1.09E+00	1.91E-03	-	-
6	-	-	0	0	-	-	-0.01 (0.00)	0.00 (0.00)	-	-	3.22E-05	4.79E-07
8	0	0	-	-	0.00 (0.00)	0.12 (0.08)	-	-	0.00E+00	1.45E-02	-	-
9	0	0	-	-	0.00 (0.00)	0.00 (0.00)	-	-	0.00E+00	1.13E-05	-	-
11	-1	-1	-	-	-1.00 (0.00)	-0.97 (0.03)	-	-	0.00E+00	1.20E-03	-	-
12	0	0	-	-	-1.00 (0.00)	0.00 (0.00)	-	-	1.00E+00	5.89E-08	-	-

Table C.119

RS Predictions: ***Digdug***												
Stype	**Ground Truth**				**Estimations**				**Accuracy Square Error**			
	CW	**CS**	**AW**	**AS**	**CW**	**CS**	**AW**	**AS**	**CW**	**CS**	**AW**	**AS**
0	0	0	0	0	-0.00 (0.00)	-0.00 (0.00)	-0.00 (0.00)	-0.00 (0.00)	1.08E-05	9.25E-07	1.30E-05	6.57E-07
4	0	0	1	0	-0.00 (0.00)	-0.00 (0.00)	-0.01 (0.00)	-0.01 (0.00)	2.00E-05	1.14E-05	1.01E+00	2.70E-05
5	1	1	-	-	-0.02 (0.01)	0.98 (0.00)	-	-	1.04E+00	3.75E-04	-	-
6	-	-	0	0	-	-	-0.01 (0.00)	-0.00 (0.00)	-	-	6.12E-05	2.12E-06
8	0	0	-	-	-0.00 (0.00)	0.04 (0.03)	-	-	5.60E-06	1.34E-03	-	-
9	0	0	-	-	-0.00 (0.00)	0.00 (0.00)	-	-	7.51E-06	2.05E-05	-	-
11	-1	-1	-	-	-1.00 (0.00)	-1.00 (0.01)	-	-	0.00E+00	9.37E-06	-	-
12	0	0	-	-	-1.00 (0.00)	-0.00 (0.00)	-	-	1.00E+00	1.56E-05	-	-

Table C.120

OSLA Predictions: *Escape*												
Stype	**Ground Truth**				**Estimations**				**Accuracy Square Error**			
	CW	**CS**	**AW**	**AS**	**CW**	**CS**	**AW**	**AS**	**CW**	**CS**	**AW**	**AS**
0	0	0	-	-	-0.00 (0.00)	0.00 (0.00)	-	-	7.50E-07	0.00E+00	-	-

Table C.121

OLETS Predictions: *Escape*												
Stype	**Ground Truth**				**Estimations**				**Accuracy Square Error**			
	CW	**CS**	**AW**	**AS**	**CW**	**CS**	**AW**	**AS**	**CW**	**CS**	**AW**	**AS**
0	0	0	-	-	-0.00 (0.00)	0.00 (0.00)	-	-	3.90E-07	0.00E+00	-	-
3	0	0	-	-	-0.00 (0.00)	0.00 (0.00)	-	-	1.46E-07	0.00E+00	-	-
4	1	1	-	-	1.00 (0.00)	1.00 (0.00)	-	-	0.00E+00	0.00E+00	-	-
5	-1	-1	-	-	-1.00 (0.00)	-1.00 (0.00)	-	-	0.00E+00	0.00E+00	-	-

Table C.122

OLMCTS Predictions: *Escape*												
Stype	**Ground Truth**				**Estimations**				**Accuracy Square Error**			
	CW	**CS**	**AW**	**AS**	**CW**	**CS**	**AW**	**AS**	**CW**	**CS**	**AW**	**AS**
0	0	0	-	-	-0.00 (0.00)	0.00 (0.00)	-	-	1.49E-06	0.00E+00	-	-
3	0	0	-	-	-0.00 (0.00)	0.00 (0.00)	-	-	7.12E-06	0.00E+00	-	-
5	-1	-1	-	-	-1.00 (0.00)	-1.00 (0.00)	-	-	0.00E+00	0.00E+00	-	-

Table C.123

RHEA Predictions: *Escape*												
Stype	**Ground Truth**				**Estimations**				**Accuracy Square Error**			
	CW	**CS**	**AW**	**AS**	**CW**	**CS**	**AW**	**AS**	**CW**	**CS**	**AW**	**AS**
0	0	0	-	-	0.00 (0.00)	0.00 (0.00)	-	-	0.00E+00	0.00E+00	-	-
3	0	0	-	-	0.00 (0.00)	0.00 (0.00)	-	-	0.00E+00	0.00E+00	-	-
4	1	1	-	-	1.00 (0.00)	1.00 (0.00)	-	-	0.00E+00	0.00E+00	-	-
5	-1	-1	-	-	-1.00 (0.00)	-1.00 (0.00)	-	-	0.00E+00	0.00E+00	-	-

Table C.124

RS Predictions: *Escape*												
Stype	**Ground Truth**				**Estimations**				**Accuracy Square Error**			
	CW	**CS**	**AW**	**AS**	**CW**	**CS**	**AW**	**AS**	**CW**	**CS**	**AW**	**AS**
0	0	0	-	-	-0.00 (0.00)	0.00 (0.00)	-	-	2.66E-06	0.00E+00	-	-
3	0	0	-	-	-0.00 (0.00)	0.00 (0.00)	-	-	1.99E-06	0.00E+00	-	-
4	1	1	-	-	1.00 (0.00)	1.00 (0.00)	-	-	0.00E+00	0.00E+00	-	-
5	-1	-1	-	-	-1.00 (0.00)	-1.00 (0.00)	-	-	0.00E+00	0.00E+00	-	-

Table C.125

OSLA Predictions: *Hungry Birds*												
Stype	**Ground Truth**				**Estimations**				**Accuracy Square Error**			
	CW	**CS**	**AW**	**AS**	**CW**	**CS**	**AW**	**AS**	**CW**	**CS**	**AW**	**AS**
0	0	0	-	-	-0.01 (0.00)	0.00 (0.00)	-	-	2.99E-05	0.00E+00	-	-

Table C.126

OLETS Predictions: *Hungry Birds*												
Stype	**Ground Truth**				**Estimations**				**Accuracy Square Error**			
	CW	**CS**	**AW**	**AS**	**CW**	**CS**	**AW**	**AS**	**CW**	**CS**	**AW**	**AS**
0	0	0	-	-	-0.00 (0.00)	0.00 (0.00)	-	-	5.61E-10	0.00E+00	-	-
5	0	0	-	-	0.00 (0.00)	0.00 (0.00)	-	-	0.00E+00	0.00E+00	-	-

Table C.127

OLMCTS Predictions: *Hungry Birds*												
Stype	**Ground Truth**				**Estimations**				**Accuracy Square Error**			
	CW	**CS**	**AW**	**AS**	**CW**	**CS**	**AW**	**AS**	**CW**	**CS**	**AW**	**AS**
0	0	0	-	-	-0.00 (0.00)	0.00 (0.00)	-	-	5.98E-10	0.00E+00	-	-
5	0	0	-	-	0.00 (0.00)	0.00 (0.00)	-	-	0.00E+00	0.00E+00	-	-

Table C.128

RHEA Predictions: *Hungry Birds*												
Stype	**Ground Truth**				**Estimations**				**Accuracy Square Error**			
	CW	**CS**	**AW**	**AS**	**CW**	**CS**	**AW**	**AS**	**CW**	**CS**	**AW**	**AS**
0	0	0	-	-	-0.00 (0.00)	0.00 (0.00)	-	-	1.33E-09	0.00E+00	-	-
5	0	0	-	-	0.00 (0.00)	0.00 (0.00)	-	-	0.00E+00	0.00E+00	-	-

Table C.129

RS Predictions: *Hungry Birds*												
Stype	**Ground Truth**				**Estimations**				**Accuracy Square Error**			
	CW	**CS**	**AW**	**AS**	**CW**	**CS**	**AW**	**AS**	**CW**	**CS**	**AW**	**AS**
0	0	0	-	-	-0.00 (0.00)	0.00 (0.00)	-	-	1.31E-09	0.00E+00	-	-
5	0	0	-	-	0.00 (0.00)	0.00 (0.00)	-	-	0.00E+00	0.00E+00	-	-

Table C.130

OSLA Predictions: *Infection*												
Stype	**Ground Truth**				**Estimations**				**Accuracy Square Error**			
	CW	**CS**	**AW**	**AS**	**CW**	**CS**	**AW**	**AS**	**CW**	**CS**	**AW**	**AS**
0	0	0	-	-	0.00 (0.00)	-0.00 (0.00)	-	-	1.07E-08	8.08E-08	-	-
10	1	2	-	-	0.00 (0.00)	2.00 (0.00)	-	-	1.00E+00	0.00E+00	-	-
11	0	0	-	-	0.00 (0.00)	0.00 (0.00)	-	-	0.00E+00	0.00E+00	-	-
12	0	-1	0	2	0.00 (0.00)	-0.75 (0.15)	0.00 (0.00)	1.81 (0.07)	0.00E+00	6.21E-02	0.00E+00	3.43E-02

Table C.131

OLETS Predictions: *Infection*												
Stype	**Ground Truth**				**Estimations**				**Accuracy Square Error**			
	CW	**CS**	**AW**	**AS**	**CW**	**CS**	**AW**	**AS**	**CW**	**CS**	**AW**	**AS**
0	0	0	-	-	0.00 (0.00)	0.00 (0.00)	-	-	5.36E-09	8.02E-09	-	-
10	1	2	-	-	0.00 (0.00)	2.03 (0.03)	-	-	1.00E+00	1.06E-03	-	-
11	0	0	-	-	0.00 (0.00)	0.02 (0.01)	-	-	0.00E+00	2.61E-04	-	-
12	0	-1	0	2	0.00 (0.00)	-0.86 (0.03)	0.00 (0.00)	1.90 (0.03)	0.00E+00	2.02E-02	0.00E+00	9.98E-03
5	0	0	-	-	0.00 (0.00)	0.00 (0.00)	-	-	0.00E+00	2.78E-06	-	-

Table C.132

OLMCTS Predictions: *Infection*												
Stype	**Ground Truth**				**Estimations**				**Accuracy Square Error**			
	CW	**CS**	**AW**	**AS**	**CW**	**CS**	**AW**	**AS**	**CW**	**CS**	**AW**	**AS**
0	0	0	-	-	0.00 (0.00)	0.00 (0.00)	-	-	1.71E-09	1.14E-07	-	-
10	1	2	-	-	0.03 (0.02)	1.99 (0.00)	-	-	9.43E-01	4.09E-05	-	-
11	0	0	-	-	0.00 (0.00)	0.02 (0.00)	-	-	0.00E+00	2.43E-04	-	-
12	0	-1	0	2	0.00 (0.00)	-0.82 (0.03)	0.00 (0.00)	1.84 (0.05)	0.00E+00	3.23E-02	2.39E-08	2.45E-02
5	0	0	-	-	0.00 (0.00)	0.00 (0.00)	-	-	0.00E+00	1.31E-05	-	-

Table C.133

RHEA Predictions: *Infection*												
Stype	**Ground Truth**				**Estimations**				**Accuracy Square Error**			
	CW	**CS**	**AW**	**AS**	**CW**	**CS**	**AW**	**AS**	**CW**	**CS**	**AW**	**AS**
0	0	0	-	-	0.00 (0.00)	0.00 (0.00)	-	-	6.47E-08	2.92E-07	-	-
10	1	2	-	-	0.00 (0.00)	2.03 (0.04)	-	-	9.98E-01	7.86E-04	-	-
11	0	0	-	-	0.00 (0.00)	0.03 (0.01)	-	-	6.99E-08	1.03E-03	-	-
12	0	-1	0	2	0.00 (0.00)	-0.87 (0.02)	0.00 (0.00)	1.80 (0.03)	1.42E-06	1.79E-02	1.97E-06	4.06E-02
5	0	0	-	-	0.00 (0.00)	0.02 (0.01)	-	-	0.00E+00	2.49E-04	-	-

Table C.134

RS Predictions: *Infection*												
Stype	**Ground Truth**				**Estimations**				**Accuracy Square Error**			
	CW	**CS**	**AW**	**AS**	**CW**	**CS**	**AW**	**AS**	**CW**	**CS**	**AW**	**AS**
0	0	0	-	-	0.00 (0.00)	0.00 (0.00)	-	-	1.97E-07	9.97E-07	-	-
10	1	2	-	-	0.00 (0.00)	1.97 (0.04)	-	-	1.00E+00	1.04E-03	-	-
11	0	0	-	-	0.00 (0.00)	0.04 (0.01)	-	-	0.00E+00	1.50E-03	-	-
12	0	-1	0	2	0.00 (0.00)	-0.79 (0.01)	0.00 (0.00)	1.80 (0.04)	0.00E+00	4.40E-02	7.97E-09	4.16E-02
5	0	0	-	-	0.00 (0.00)	0.01 (0.00)	-	-	0.00E+00	3.93E-05	-	-

Table C.135

OSLA Predictions: *Intersection*												
Stype	**Ground Truth**				**Estimations**				**Accuracy Square Error**			
	CW	**CS**	**AW**	**AS**	**CW**	**CS**	**AW**	**AS**	**CW**	**CS**	**AW**	**AS**
0	0	0	-	-	0.00 (0.00)	0.00 (0.00)	-	-	4.66E-07	0.00E+00	-	-
15	0	0	-	-	0.00 (0.00)	0.00 (0.00)	-	-	3.76E-07	0.00E+00	-	-

Table C.136

OLETS Predictions: *Intersection*												
Stype	**Ground Truth**				**Estimations**				**Accuracy Square Error**			
	CW	**CS**	**AW**	**AS**	**CW**	**CS**	**AW**	**AS**	**CW**	**CS**	**AW**	**AS**
0	0	0	-	-	0.00 (0.00)	0.00 (0.00)	-	-	2.11E-07	0.00E+00	-	-
10	-1	-5	-	-	-0.00 (0.00)	-5.00 (0.00)	-	-	9.99E-01	0.00E+00	-	-
12	-1	-5	-	-	-0.08 (0.07)	-5.00 (0.00)	-	-	8.52E-01	0.00E+00	-	-
6	0	0	-	-	-0.00 (0.00)	-5.00 (0.00)	-	-	3.70E-07	2.50E+01	-	-
15	0	0	-	-	0.06 (0.05)	0.00 (0.00)	-	-	3.11E-03	0.00E+00	-	-

Table C.137

OLMCTS Predictions: *Intersection*												
Stype	**Ground Truth**				**Estimations**				**Accuracy Square Error**			
	CW	**CS**	**AW**	**AS**	**CW**	**CS**	**AW**	**AS**	**CW**	**CS**	**AW**	**AS**
0	0	0	-	-	0.00 (0.00)	0.00 (0.00)	-	-	4.05E-06	0.00E+00	-	-
5	0	10	-	-	0.00 (0.00)	10.00 (0.00)	-	-	0.00E+00	0.00E+00	-	-
6	0	0	-	-	-0.28 (0.04)	-5.01 (0.01)	-	-	7.73E-02	2.51E+01	-	-
10	-1	-5	-	-	-0.32 (0.04)	-5.03 (0.01)	-	-	4.65E-01	7.86E-04	-	-
12	-1	-5	-	-	-0.39 (0.09)	-5.05 (0.02)	-	-	3.77E-01	2.04E-03	-	-
15	0	0	-	-	0.00 (0.00)	0.00 (0.00)	-	-	2.90E-06	0.00E+00	-	-

Table C.138

RHEA Predictions: *Intersection*												
Stype	**Ground Truth**				**Estimations**				**Accuracy Square Error**			
	CW	**CS**	**AW**	**AS**	**CW**	**CS**	**AW**	**AS**	**CW**	**CS**	**AW**	**AS**
0	0	0	-	-	0.00 (0.00)	0.00 (0.00)	-	-	0.00E+00	0.00E+00	-	-
5	0	10	-	-	0.00 (0.00)	10.00 (0.00)	-	-	0.00E+00	0.00E+00	-	-
6	0	0	-	-	-0.32 (0.04)	-5.01 (0.00)	-	-	1.02E-01	2.51E+01	-	-
10	-1	-5	-	-	-0.29 (0.05)	-5.02 (0.00)	-	-	4.98E-01	3.14E-04	-	-
12	-1	-5	-	-	-0.26 (0.07)	-5.13 (0.06)	-	-	5.50E-01	1.61E-02	-	-
15	0	0	-	-	0.00 (0.00)	0.00 (0.00)	-	-	0.00E+00	0.00E+00	-	-

Table C.139

RS Predictions: *Intersection*												
Stype	**Ground Truth**				**Estimations**				**Accuracy Square Error**			
	CW	**CS**	**AW**	**AS**	**CW**	**CS**	**AW**	**AS**	**CW**	**CS**	**AW**	**AS**
0	0	0	-	-	0.00 (0.00)	0.00 (0.00)	-	-	4.12E-06	0.00E+00	-	-
5	0	10	-	-	0.00 (0.00)	10.00 (0.00)	-	-	0.00E+00	0.00E+00	-	-
6	0	0	-	-	-0.63 (0.04)	-5.01 (0.00)	-	-	4.02E-01	2.51E+01	-	-
10	-1	-5	-	-	-0.66 (0.04)	-5.01 (0.00)	-	-	1.17E-01	1.79E-04	-	-
12	-1	-5	-	-	-0.38 (0.08)	-5.04 (0.01)	-	-	3.85E-01	1.79E-03	-	-
15	0	0	-	-	0.00 (0.00)	0.00 (0.00)	-	-	2.89E-06	0.00E+00	-	-

Table C.140

OSLA Predictions: *Lemmings*

Stype	Ground Truth				Estimations				Accuracy Square Error			
	CW	**CS**	**AW**	**AS**	**CW**	**CS**	**AW**	**AS**	**CW**	**CS**	**AW**	**AS**
0	0	0	0	-1	-0.00 (0.00)	0.00 (0.01)	0.00 (0.00)	-1.00 (0.00)	1.74E-07	2.27E-05	0.00E+00	2.08E-05
3	-1	-5	-	-	-1.00 (0.00)	-5.00 (0.00)	-	-	0.00E+00	2.08E-05	-	-

Table C.141

OLETS Predictions: *Lemmings*

Stype	Ground Truth				Estimations				Accuracy Square Error			
	CW	**CS**	**AW**	**AS**	**CW**	**CS**	**AW**	**AS**	**CW**	**CS**	**AW**	**AS**
0	0	0	0	-1	0.00 (0.00)	0.01 (0.00)	0.00 (0.00)	-0.99 (0.01)	3.49E-10	6.36E-05	3.17E-10	1.52E-04
3	-1	-5	-	-	-1.00 (0.00)	-4.98 (0.01)	-	-	0.00E+00	3.57E-04	-	-

Table C.142

OLMCTS Predictions: *Lemmings*

Stype	Ground Truth				Estimations				Accuracy Square Error			
	CW	**CS**	**AW**	**AS**	**CW**	**CS**	**AW**	**AS**	**CW**	**CS**	**AW**	**AS**
0	0	0	0	-1	0.00 (0.00)	-0.00 (0.00)	0.00 (0.00)	-1.00 (0.00)	1.12E-07	1.51E-06	9.39E-08	4.37E-09
3	-1	-5	-	-	-1.00 (0.00)	-4.99 (0.01)	-	-	0.00E+00	1.89E-04	-	-

Table C.143

RHEA Predictions: *Lemmings*												
Stype	**Ground Truth**				**Estimations**				**Accuracy Square Error**			
	CW	**CS**	**AW**	**AS**	**CW**	**CS**	**AW**	**AS**	**CW**	**CS**	**AW**	**AS**
0	0	0	0	-1	-0.00 (0.00)	-0.00 (0.00)	-0.00 (0.00)	-1.00 (0.00)	4.68E-08	5.31E-06	3.43E-08	1.66E-06
3	-1	-5	-	-	-1.00 (0.00)	-4.99 (0.01)	-	-	0.00E+00	7.15E-05	-	-

Table C.144

RS Predictions: *Lemmings*												
Stype	**Ground Truth**				**Estimations**				**Accuracy Square Error**			
	CW	**CS**	**AW**	**AS**	**CW**	**CS**	**AW**	**AS**	**CW**	**CS**	**AW**	**AS**
0	0	0	0	-1	0.00 (0.00)	0.01 (0.00)	0.00 (0.00)	-0.99 (0.00)	8.11E-08	4.96E-05	2.84E-08	2.72E-05
3	-1	-5	-	-	-1.00 (0.00)	-4.98 (0.01)	-	-	0.00E+00	2.83E-04	-	-

Table C.145

OSLA Predictions: *Missile Command*												
Stype	**Ground Truth**				**Estimations**				**Accuracy Square Error**			
	CW	**CS**	**AW**	**AS**	**CW**	**CS**	**AW**	**AS**	**CW**	**CS**	**AW**	**AS**
0	0	0	-	-	-0.02 (0.01)	-0.04 (0.01)	-	-	2.67E-04	1.36E-03	-	-
7	-	-	1	2	-	-	0.00 (0.00)	2.00 (0.00)	-	-	1.00E+00	0.00E+00

Table C.146

OLETS Predictions: *Missile Command*												
Stype	**Ground Truth**				**Estimations**				**Accuracy Square Error**			
	CW	**CS**	**AW**	**AS**	**CW**	**CS**	**AW**	**AS**	**CW**	**CS**	**AW**	**AS**
0	0	0	-	-	-0.00 (0.00)	-0.00 (0.00)	-	-	2.68E-07	2.42E-06	-	-

Table C.147

OLMCTS Predictions: *Missile Command*												
Stype	**Ground Truth**				**Estimations**				**Accuracy Square Error**			
	CW	**CS**	**AW**	**AS**	**CW**	**CS**	**AW**	**AS**	**CW**	**CS**	**AW**	**AS**
0	0	0	-	-	-0.01 (0.01)	-0.04 (0.01)	-	-	1.93E-04	1.72E-03	-	-
7	-	-	1	2	-	-	0.14 (0.06)	2.00 (0.00)	-	-	7.37E-01	0.00E+00

Table C.148

RHEA Predictions: *Missile Command*

Stype	Ground Truth				Estimations				Accuracy Square Error			
	CW	**CS**	**AW**	**AS**	**CW**	**CS**	**AW**	**AS**	**CW**	**CS**	**AW**	**AS**
0	0	0	-	-	0.01 (0.03)	-0.10 (0.06)	-	-	1.97E-04	9.27E-03	-	-
7	-	-	1	2	-	-	0.13 (0.05)	2.00 (0.00)	-	-	7.56E-01	2.20E-06

Table C.149

RS Predictions: *Missile Command*

Stype	Ground Truth				Estimations				Accuracy Square Error			
	CW	**CS**	**AW**	**AS**	**CW**	**CS**	**AW**	**AS**	**CW**	**CS**	**AW**	**AS**
0	0	0	-	-	-0.00 (0.01)	-0.04 (0.02)	-	-	1.49E-05	1.70E-03	-	-
7	-	-	1	2	-	-	0.18 (0.06)	1.94 (0.03)	-	-	6.75E-01	3.39E-03

Table C.150

OSLA Predictions: *Modality*												
Stype	**Ground Truth**				**Estimations**				**Accuracy Square Error**			
	CW	**CS**	**AW**	**AS**	**CW**	**CS**	**AW**	**AS**	**CW**	**CS**	**AW**	**AS**
0	0	0	-	-	-0.00 (0.00)	0.00 (0.00)	-	-	1.60E-07	0.00E+00	-	-
2	0	0	-	-	0.00 (0.00)	0.00 (0.00)	-	-	0.00E+00	0.00E+00	-	-

Table C.151

OLETS Predictions: *Modality*												
Stype	**Ground Truth**				**Estimations**				**Accuracy Square Error**			
	CW	**CS**	**AW**	**AS**	**CW**	**CS**	**AW**	**AS**	**CW**	**CS**	**AW**	**AS**
0	0	0	-	-	-0.00 (0.00)	0.00 (0.00)	-	-	3.36E-07	0.00E+00	-	-
2	0	0	-	-	-0.00 (0.00)	0.00 (0.00)	-	-	7.86E-09	0.00E+00	-	-
3	0	0	-	-	-0.00 (0.00)	0.00 (0.00)	-	-	1.56E-08	0.00E+00	-	-
4	0	0	-	-	-0.00 (0.00)	0.00 (0.00)	-	-	3.61E-08	0.00E+00	-	-
7	0	0	-	-	0.00 (0.00)	0.00 (0.00)	-	-	3.49E-07	5.77E-07	-	-

Table C.152

OLMCTS Predictions: *Modality*												
Stype	**Ground Truth**				**Estimations**				**Accuracy Square Error**			
	CW	**CS**	**AW**	**AS**	**CW**	**CS**	**AW**	**AS**	**CW**	**CS**	**AW**	**AS**
0	0	0	-	-	-0.00 (0.00)	0.00 (0.00)	-	-	2.11E-07	0.00E+00	-	-
2	0	0	-	-	-0.00 (0.00)	0.00 (0.00)	-	-	2.76E-07	0.00E+00	-	-
3	0	0	-	-	-0.00 (0.00)	0.00 (0.00)	-	-	3.01E-07	0.00E+00	-	-
4	0	0	-	-	-0.00 (0.00)	0.00 (0.00)	-	-	3.88E-07	0.00E+00	-	-
7	0	0	-	-	0.00 (0.00)	0.00 (0.00)	-	-	6.06E-06	7.54E-06	-	-

Table C.153

RHEA Predictions: *Modality*												
Stype	**Ground Truth**				**Estimations**				**Accuracy Square Error**			
	CW	**CS**	**AW**	**AS**	**CW**	**CS**	**AW**	**AS**	**CW**	**CS**	**AW**	**AS**
0	0	0	-	-	-0.00 (0.00)	0.00 (0.00)	-	-	8.71E-08	0.00E+00	-	-
2	0	0	-	-	-0.00 (0.00)	0.00 (0.00)	-	-	2.96E-07	0.00E+00	-	-
3	0	0	-	-	-0.00 (0.00)	0.00 (0.00)	-	-	2.42E-07	0.00E+00	-	-
4	0	0	-	-	-0.00 (0.00)	0.00 (0.00)	-	-	2.11E-07	0.00E+00	-	-
7	0	0	-	-	0.19 (0.04)	0.19 (0.04)	-	-	3.73E-02	3.75E-02	-	-

Table C.154

RS Predictions: *Modality*												
Stype	**Ground Truth**				**Estimations**				**Accuracy Square Error**			
	CW	**CS**	**AW**	**AS**	**CW**	**CS**	**AW**	**AS**	**CW**	**CS**	**AW**	**AS**
0	0	0	-	-	-0.00 (0.00)	0.00 (0.00)	-	-	2.08E-07	0.00E+00	-	-
2	0	0	-	-	-0.00 (0.00)	0.00 (0.00)	-	-	1.69E-07	0.00E+00	-	-
3	0	0	-	-	-0.00 (0.00)	0.00 (0.00)	-	-	1.43E-07	0.00E+00	-	-
4	0	0	-	-	-0.00 (0.00)	0.00 (0.00)	-	-	3.34E-07	0.00E+00	-	-
7	0	0	-	-	0.15 (0.01)	0.15 (0.01)	-	-	2.12E-02	2.13E-02	-	-

Table C.155

OSLA Predictions: *Plaque Attack*												
Stype	**Ground Truth**				**Estimations**				**Accuracy Square Error**			
	CW	**CS**	**AW**	**AS**	**CW**	**CS**	**AW**	**AS**	**CW**	**CS**	**AW**	**AS**
0	0	0	0	0	-0.00 (0.00)	-0.05 (0.01)	-0.00 (0.00)	-0.06 (0.01)	3.29E-06	2.85E-03	9.42E-06	3.49E-03
11	-	-	1	2	-	-	0.00 (0.00)	2.20 (0.12)	-	-	1.00E+00	4.07E-02
7	0	1	-	-	0.04 (0.01)	-2.04 (0.04)	-	-	1.52E-03	9.25E+00	-	-

Table C.156

OLETS Predictions: *Plaque Attack*												
Stype	**Ground Truth**				**Estimations**				**Accuracy Square Error**			
	CW	**CS**	**AW**	**AS**	**CW**	**CS**	**AW**	**AS**	**CW**	**CS**	**AW**	**AS**
0	0	0	0	0	-0.00 (0.00)	-0.04 (0.00)	-0.00 (0.00)	-0.03 (0.00)	2.56E-07	1.70E-03	2.40E-07	1.21E-03
11	-	-	1	2	-	-	-0.14 (0.10)	2.03 (0.12)	-	-	1.31E+00	6.80E-04
7	0	1	-	-	0.05 (0.03)	-1.09 (0.61)	-	-	3.00E-03	4.36E+00	-	-

Table C.157

OLMCTS Predictions: *Plaque Attack*												
Stype	**Ground Truth**				**Estimations**				**Accuracy Square Error**			
	CW	**CS**	**AW**	**AS**	**CW**	**CS**	**AW**	**AS**	**CW**	**CS**	**AW**	**AS**
0	0	0	0	0	0.00 (0.00)	-0.02 (0.01)	-0.00 (0.00)	-0.00 (0.01)	1.93E-06	3.38E-04	1.80E-06	1.72E-05
11	-	-	1	2	-	-	0.02 (0.01)	2.02 (0.04)	-	-	9.59E-01	2.41E-04
7	0	1	-	-	0.03 (0.03)	0.13 (0.19)	-	-	9.23E-04	7.60E-01	-	-

Table C.158

RHEA Predictions: ***Plaque Attack***												
Stype	**Ground Truth**				**Estimations**				**Accuracy Square Error**			
	CW	**CS**	**AW**	**AS**	**CW**	**CS**	**AW**	**AS**	**CW**	**CS**	**AW**	**AS**
0	0	0	0	0	-0.00 (0.00)	-0.02 (0.01)	-0.00 (0.00)	-0.02 (0.01)	1.35E-05	3.33E-04	8.18E-06	2.42E-04
11	-	-	1	2	-	-	0.01 (0.01)	1.99 (0.03)	-	-	9.79E-01	4.21E-05
7	0	1	-	-	0.01 (0.01)	-0.04 (0.21)	-	-	6.20E-05	1.08E+00	-	-

Table C.159

RS Predictions: ***Plaque Attack***												
Stype	**Ground Truth**				**Estimations**				**Accuracy Square Error**			
	CW	**CS**	**AW**	**AS**	**CW**	**CS**	**AW**	**AS**	**CW**	**CS**	**AW**	**AS**
0	0	0	0	0	-0.00 (0.00)	-0.02 (0.00)	-0.00 (0.00)	-0.01 (0.01)	8.90E-08	3.61E-04	1.10E-07	5.29E-05
11	-	-	1	2	-	-	0.04 (0.01)	1.98 (0.05)	-	-	9.13E-01	2.48E-04
7	0	1	-	-	0.12 (0.05)	-0.18 (0.16)	-	-	1.36E-02	1.40E+00	-	-

Table C.160

OSLA Predictions: *Roguelike*												
Stype	**Ground Truth**				**Estimations**				**Accuracy Square Error**			
	CW	**CS**	**AW**	**AS**	**CW**	**CS**	**AW**	**AS**	**CW**	**CS**	**AW**	**AS**
0	0	0	-	-	-0.00 (0.00)	0.00 (0.00)	-	-	5.55E-07	0.00E+00	-	-
13	-1	0	0	2	-1.00 (0.00)	0.00 (0.00)	-	-	0.00E+00	0.00E+00	-	-
14	-1	0	0	1	-1.00 (0.00)	0.00 (0.00)	-	-	0.00E+00	0.00E+00	-	-

Table C.161

OLETS Predictions: *Roguelike*												
Stype	**Ground Truth**				**Estimations**				**Accuracy Square Error**			
	CW	**CS**	**AW**	**AS**	**CW**	**CS**	**AW**	**AS**	**CW**	**CS**	**AW**	**AS**
0	0	0	-	-	-0.00 (0.00)	0.00 (0.00)	-	-	3.84E-08	0.00E+00	-	-
13	-1	0	0	2	-1.00 (0.00)	0.00 (0.00)	-	-	0.00E+00	0.00E+00	-	-
14	-1	0	0	1	-1.00 (0.00)	0.00 (0.00)	-	-	0.00E+00	0.00E+00	-	-

Table C.162

OLMCTS Predictions: *Roguelike*												
Stype	**Ground Truth**				**Estimations**				**Accuracy Square Error**			
	CW	**CS**	**AW**	**AS**	**CW**	**CS**	**AW**	**AS**	**CW**	**CS**	**AW**	**AS**
0	0	0	-	-	-0.00 (0.00)	0.00 (0.00)	-	-	1.53E-06	0.00E+00	-	-
8	0	1	-	-	0.00 (0.00)	1.00 (0.00)	-	-	0.00E+00	0.00E+00	-	-
10	0	0	-	-	-0.25 (0.22)	0.00 (0.00)	-	-	6.42E-02	0.00E+00	-	-
13	-1	0	0	2	-1.00 (0.00)	0.00 (0.00)	-	-	0.00E+00	0.00E+00	-	-
14	-1	0	0	1	-1.00 (0.00)	0.00 (0.00)	-	-	0.00E+00	0.00E+00	-	-

Table C.163

RHEA Predictions: *Roguelike*												
Stype	**Ground Truth**				**Estimations**				**Accuracy Square Error**			
	CW	**CS**	**AW**	**AS**	**CW**	**CS**	**AW**	**AS**	**CW**	**CS**	**AW**	**AS**
0	0	0	-	-	-0.00 (0.00)	0.00 (0.00)	-	-	2.67E-06	0.00E+00	-	-
8	0	1	-	-	-0.02 (0.02)	1.00 (0.00)	-	-	3.31E-04	0.00E+00	-	-
10	0	0	-	-	-0.30 (0.14)	0.00 (0.00)	-	-	9.14E-02	0.00E+00	-	-
13	-1	0	0	2	-1.00 (0.00)	0.00 (0.00)	-	-	0.00E+00	1.54E-05	-	-
14	-1	0	0	1	-1.00 (0.00)	0.00 (0.00)	-	-	0.00E+00	0.00E+00	-	-

Table C.164

RS Predictions: *Roguelike*												
Stype	**Ground Truth**				**Estimations**				**Accuracy Square Error**			
	CW	**CS**	**AW**	**AS**	**CW**	**CS**	**AW**	**AS**	**CW**	**CS**	**AW**	**AS**
0	0	0	-	-	-0.00 (0.00)	0.00 (0.00)	-	-	5.45E-06	0.00E+00	-	-
6	0	0	-	-	-0.54 (0.32)	0.00 (0.00)	-	-	2.93E-01	0.00E+00	-	-
8	0	1	-	-	0.00 (0.00)	1.00 (0.00)	-	-	0.00E+00	0.00E+00	-	-
10	0	0	-	-	-0.38 (0.17)	0.00 (0.00)	-	-	1.45E-01	0.00E+00	-	-
13	-1	0	0	2	-1.00 (0.00)	0.00 (0.00)	-	-	0.00E+00	0.00E+00	-	-
14	-1	0	0	1	-0.96 (0.04)	0.00 (0.00)	-	-	1.78E-03	0.00E+00	-	-

Table C.165

OSLA Predictions: *Seaquest*												
Stype	**Ground Truth**				**Estimations**				**Accuracy Square Error**			
	CW	**CS**	**AW**	**AS**	**CW**	**CS**	**AW**	**AS**	**CW**	**CS**	**AW**	**AS**
16	-1	0	0	1	-1.00 (0.00)	0.06 (0.05)	-0.00 (0.00)	1.32 (0.14)	0.00E+00	3.09E-03	7.30E-06	1.01E-01
17	0	0	0	1	0.00 (0.00)	0.00 (0.00)	-	-	0.00E+00	0.00E+00	-	-
2	0	0	-	-	0.00 (0.00)	0.53 (0.18)	-	-	1.26E-06	2.77E-01	-	-
14	-1	0	0	1	-1.00 (0.00)	0.02 (0.02)	-0.05 (0.05)	1.00 (0.00)	0.00E+00	2.78E-04	2.77E-03	0.00E+00
15	-1	0	0	1	-1.00 (0.00)	0.00 (0.00)	-0.02 (0.02)	1.24 (0.09)	0.00E+00	0.00E+00	3.43E-04	5.57E-02

Table C.166

OLETS Predictions: *Seaquest*												
Stype	**Ground Truth**				**Estimations**				**Accuracy Square Error**			
	CW	**CS**	**AW**	**AS**	**CW**	**CS**	**AW**	**AS**	**CW**	**CS**	**AW**	**AS**
16	-1	0	0	1	-1.00 (0.00)	0.04 (0.02)	-0.00 (0.00)	1.35 (0.08)	0.00E+00	1.74E-03	2.75E-07	1.21E-01
17	0	0	0	1	-0.00 (0.00)	1.43 (0.75)	-	-	1.01E-07	2.03E+00	-	-
2	0	0	-	-	0.00 (0.00)	0.56 (0.12)	-	-	8.01E-07	3.16E-01	-	-
14	-1	0	0	1	-1.00 (0.00)	0.04 (0.01)	-0.00 (0.00)	1.03 (0.01)	0.00E+00	2.01E-03	6.13E-07	7.77E-04
15	-1	0	0	1	-1.00 (0.00)	0.03 (0.02)	-0.00 (0.00)	1.46 (0.31)	0.00E+00	1.04E-03	1.68E-07	2.13E-01

Table C.167

OLMCTS Predictions: ***Seaquest***												
Stype	**Ground Truth**				**Estimations**				**Accuracy Square Error**			
	CW	**CS**	**AW**	**AS**	**CW**	**CS**	**AW**	**AS**	**CW**	**CS**	**AW**	**AS**
16	-1	0	0	1	-1.00 (0.00)	0.06 (0.01)	-0.00 (0.00)	1.38 (0.06)	0.00E+00	4.04E-03	1.55E-06	1.48E-01
17	0	0	0	1	-0.00 (0.00)	0.81 (0.44)	-	-	9.63E-07	6.58E-01	-	-
2	0	0	-	-	0.00 (0.00)	0.81 (0.17)	-	-	2.38E-06	6.57E-01	-	-
14	-1	0	0	1	-1.00 (0.00)	0.12 (0.01)	-0.01 (0.00)	1.78 (0.46)	0.00E+00	1.34E-02	3.48E-05	6.15E-01
15	-1	0	0	1	-1.00 (0.00)	0.03 (0.01)	-0.00 (0.00)	3.07 (1.58)	0.00E+00	1.21E-03	9.45E-07	4.29E+00

Table C.168

RHEA Predictions: ***Seaquest***												
Stype	**Ground Truth**				**Estimations**				**Accuracy Square Error**			
	CW	**CS**	**AW**	**AS**	**CW**	**CS**	**AW**	**AS**	**CW**	**CS**	**AW**	**AS**
16	-1	0	0	1	-1.00 (0.00)	0.11 (0.02)	-0.03 (0.01)	1.51 (0.20)	0.00E+00	1.29E-02	6.70E-04	2.57E-01
17	0	0	0	1	-0.00 (0.00)	0.29 (0.26)	-	-	2.42E-06	8.67E-02	-	-
2	0	0	-	-	0.00 (0.00)	0.40 (0.10)	-	-	4.15E-07	1.63E-01	-	-
14	-1	0	0	1	-1.00 (0.00)	0.12 (0.01)	-0.04 (0.02)	1.22 (0.08)	0.00E+00	1.38E-02	1.97E-03	4.91E-02
15	-1	0	0	1	-1.00 (0.00)	0.01 (0.01)	-0.00 (0.00)	1.33 (0.10)	0.00E+00	2.24E-04	2.51E-06	1.10E-01

Table C.169

RS Predictions: *Seaquest*

Stype	**Ground Truth**				**Estimations**				**Accuracy Square Error**			
	CW	**CS**	**AW**	**AS**	**CW**	**CS**	**AW**	**AS**	**CW**	**CS**	**AW**	**AS**
16	-1	0	0	1	-1.00 (0.00)	0.16 (0.02)	-0.01 (0.00)	1.34 (0.07)	0.00E+00	2.45E-02	4.62E-05	1.16E-01
17	0	0	0	1	-0.00 (0.00)	0.29 (0.18)	-	-	1.22E-05	8.24E-02	-	-
2	0	0	-	-	0.00 (0.00)	0.60 (0.11)	-	-	1.84E-06	3.60E-01	-	-
14	-1	0	0	1	-1.00 (0.00)	0.13 (0.01)	-0.01 (0.00)	1.04 (0.01)	0.00E+00	1.62E-02	1.53E-04	1.35E-03
15	-1	0	0	1	-1.00 (0.00)	0.10 (0.04)	-0.01 (0.00)	1.39 (0.11)	0.00E+00	1.02E-02	6.20E-05	1.54E-01

Table C.170

OSLA Predictions: *Survive Zombies*												
Stype	**Ground Truth**				**Estimations**				**Accuracy Square Error**			
	CW	**CS**	**AW**	**AS**	**CW**	**CS**	**AW**	**AS**	**CW**	**CS**	**AW**	**AS**
0	0	0	-	-	-0.00 (0.00)	-0.00 (0.00)	-	-	3.78E-08	4.26E-07	-	-
10	-1	-1	-	-	-0.27 (0.09)	-1.05 (0.03)	-	-	5.32E-01	2.44E-03	-	-
7	0	1	-	-	-0.03 (0.03)	0.96 (0.06)	-	-	8.16E-04	1.39E-03	-	-

Table C.171

OLETS Predictions: *Survive Zombies*												
Stype	**Ground Truth**				**Estimations**				**Accuracy Square Error**			
	CW	**CS**	**AW**	**AS**	**CW**	**CS**	**AW**	**AS**	**CW**	**CS**	**AW**	**AS**
0	0	0	-	-	0.00 (0.00)	-0.00 (0.00)	-	-	1.26E-09	3.94E-07	-	-
10	-1	-1	-	-	-0.14 (0.04)	-1.09 (0.04)	-	-	7.40E-01	7.96E-03	-	-
6	-1	-1	-	-	-0.99 (0.00)	-0.43 (0.39)	-	-	5.72E-05	3.26E-01	-	-
7	0	1	-	-	-0.02 (0.01)	0.88 (0.05)	-	-	3.62E-04	1.36E-02	-	-

Table C.172

OLMCTS Predictions: *Survive Zombies*												
Stype	**Ground Truth**				**Estimations**				**Accuracy Square Error**			
	CW	**CS**	**AW**	**AS**	**CW**	**CS**	**AW**	**AS**	**CW**	**CS**	**AW**	**AS**
0	0	0	-	-	0.00 (0.00)	-0.00 (0.00)	-	-	1.82E-06	1.99E-06	-	-
10	-1	-1	-	-	-0.16 (0.03)	-1.18 (0.05)	-	-	7.12E-01	3.18E-02	-	-
6	-1	-1	-	-	-1.00 (0.00)	-0.94 (0.30)	-	-	3.85E-06	4.09E-03	-	-
7	0	1	-	-	-0.04 (0.01)	0.72 (0.07)	-	-	1.69E-03	7.84E-02	-	-

Table C.173

RHEA Predictions: *Survive Zombies*												
Stype	**Ground Truth**				**Estimations**				**Accuracy Square Error**			
	CW	**CS**	**AW**	**AS**	**CW**	**CS**	**AW**	**AS**	**CW**	**CS**	**AW**	**AS**
0	0	0	-	-	-0.00 (0.00)	-0.00 (0.00)	-	-	6.31E-09	1.54E-06	-	-
10	-1	-1	-	-	-0.22 (0.03)	-1.15 (0.07)	-	-	6.03E-01	2.14E-02	-	-
6	-1	-1	-	-	-1.00 (0.00)	-1.06 (0.22)	-	-	0.00E+00	3.27E-03	-	-
7	0	1	-	-	-0.11 (0.03)	0.70 (0.06)	-	-	1.15E-02	9.05E-02	-	-

Table C.174

RS Predictions: *Survive Zombies*												
Stype	**Ground Truth**				**Estimations**				**Accuracy Square Error**			
	CW	**CS**	**AW**	**AS**	**CW**	**CS**	**AW**	**AS**	**CW**	**CS**	**AW**	**AS**
0	0	0	-	-	0.00 (0.00)	-0.00 (0.00)	-	-	1.15E-07	1.26E-05	-	-
10	-1	-1	-	-	-0.39 (0.04)	-1.44 (0.11)	-	-	3.71E-01	1.90E-01	-	-
6	-1	-1	-	-	-1.00 (0.00)	-0.43 (0.30)	-	-	1.36E-05	3.29E-01	-	-
7	0	1	-	-	-0.24 (0.04)	0.57 (0.09)	-	-	5.71E-02	1.86E-01	-	-

Table C.175

OSLA Predictions: *Wait for Breakfast*												
Stype	**Ground Truth**				**Estimations**				**Accuracy Square Error**			
	CW	**CS**	**AW**	**AS**	**CW**	**CS**	**AW**	**AS**	**CW**	**CS**	**AW**	**AS**
0	0	0	-	-	-0.55 (0.11)	0.00 (0.00)	-	-	3.04E-01	0.00E+00	-	-
18	0	0	-	-	0.00 (0.00)	0.00 (0.00)	-	-	0.00E+00	0.00E+00	-	-
4	-1	0	-	-	-1.00 (0.00)	0.00 (0.00)	-	-	0.00E+00	0.00E+00	-	-
2	-1	0	-	-	-0.19 (0.04)	0.00 (0.00)	-	-	6.62E-01	0.00E+00	-	-
5	0	0	-	-	-0.00 (0.00)	0.00 (0.00)	-	-	1.21E-07	0.00E+00	-	-

Table C.176

OLETS Predictions: *Wait for Breakfast*												
Stype	**Ground Truth**				**Estimations**				**Accuracy Square Error**			
	CW	**CS**	**AW**	**AS**	**CW**	**CS**	**AW**	**AS**	**CW**	**CS**	**AW**	**AS**
0	0	0	-	-	-0.85 (0.08)	0.00 (0.00)	-	-	7.23E-01	0.00E+00	-	-
2	-1	0	-	-	-0.02 (0.00)	0.01 (0.00)	-	-	9.61E-01	1.06E-04	-	-
4	-1	0	-	-	-1.00 (0.00)	0.00 (0.00)	-	-	0.00E+00	0.00E+00	-	-
5	0	0	-	-	-0.00 (0.00)	0.00 (0.00)	-	-	1.08E-07	0.00E+00	-	-
7	0	0	-	-	0.00 (0.00)	0.00 (0.00)	-	-	0.00E+00	0.00E+00	-	-
8	1	1	-	-	1.00 (0.00)	1.00 (0.00)	-	-	0.00E+00	0.00E+00	-	-
18	0	0	-	-	0.00 (0.00)	0.00 (0.00)	-	-	0.00E+00	0.00E+00	-	-

Table C.177

OLMCTS Predictions: *Wait for Breakfast*												
Stype	**Ground Truth**				**Estimations**				**Accuracy Square Error**			
	CW	**CS**	**AW**	**AS**	**CW**	**CS**	**AW**	**AS**	**CW**	**CS**	**AW**	**AS**
0	0	0	-	-	-0.40 (0.11)	0.00 (0.00)	-	-	1.59E-01	0.00E+00	-	-
2	-1	0	-	-	-0.37 (0.05)	0.00 (0.00)	-	-	4.03E-01	6.78E-06	-	-
4	-1	0	-	-	-1.00 (0.00)	0.00 (0.00)	-	-	0.00E+00	0.00E+00	-	-
5	0	0	-	-	-0.00 (0.00)	0.00 (0.00)	-	-	6.78E-07	0.00E+00	-	-
7	0	0	-	-	0.00 (0.00)	0.00 (0.00)	-	-	0.00E+00	0.00E+00	-	-
8	1	1	-	-	1.00 (0.00)	1.00 (0.00)	-	-	0.00E+00	0.00E+00	-	-
18	0	0	-	-	0.00 (0.00)	0.00 (0.00)	-	-	0.00E+00	0.00E+00	-	-

Table C.178

RHEA Predictions: *Wait for Breakfast*												
Stype	**Ground Truth**				**Estimations**				**Accuracy Square Error**			
	CW	**CS**	**AW**	**AS**	**CW**	**CS**	**AW**	**AS**	**CW**	**CS**	**AW**	**AS**
0	0	0	-	-	-0.79 (0.07)	0.00 (0.00)	-	-	6.22E-01	0.00E+00	-	-
2	-1	0	-	-	-0.23 (0.04)	0.02 (0.01)	-	-	5.95E-01	4.96E-04	-	-
4	-1	0	-	-	-1.00 (0.00)	0.00 (0.00)	-	-	0.00E+00	0.00E+00	-	-
5	0	0	-	-	0.00 (0.00)	0.00 (0.00)	-	-	0.00E+00	0.00E+00	-	-
7	0	0	-	-	0.00 (0.00)	0.00 (0.00)	-	-	0.00E+00	0.00E+00	-	-
8	1	1	-	-	0.83 (0.16)	0.83 (0.16)	-	-	3.06E-02	3.06E-02	-	-
18	0	0	-	-	0.00 (0.00)	0.00 (0.00)	-	-	0.00E+00	0.00E+00	-	-

Table C.179

RS Predictions: *Wait for Breakfast*												
Stype	**Ground Truth**				**Estimations**				**Accuracy Square Error**			
	CW	**CS**	**AW**	**AS**	**CW**	**CS**	**AW**	**AS**	**CW**	**CS**	**AW**	**AS**
0	0	0	-	-	-0.74 (0.08)	0.00 (0.00)	-	-	5.42E-01	0.00E+00	-	-
2	-1	0	-	-	-0.10 (0.02)	0.03 (0.00)	-	-	8.14E-01	6.61E-04	-	-
4	-1	0	-	-	-1.00 (0.00)	0.00 (0.00)	-	-	0.00E+00	0.00E+00	-	-
5	0	0	-	-	-0.00 (0.00)	0.00 (0.00)	-	-	4.41E-07	0.00E+00	-	-
7	0	0	-	-	0.00 (0.00)	0.00 (0.00)	-	-	0.00E+00	0.00E+00	-	-
8	1	1	-	-	1.00 (0.00)	1.00 (0.00)	-	-	0.00E+00	0.00E+00	-	-
18	0	0	-	-	0.00 (0.00)	0.00 (0.00)	-	-	0.00E+00	0.00E+00	-	-

Table C.180

Appendix D

Full Results: Team of Agents Generated per Game

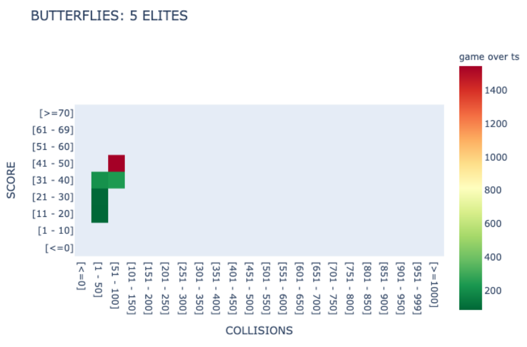
This appendix contains the graphs that constitute the resulting *team* for each of the games and set of enabled heuristics from Chapter 6. The graphs included are organised by game, as follows:

- *Team* generated for *Butterflies*, displaying B2 and B3 results together for comparison (D.1).
- *Team* generated for *Zelda*, Z5 (D.2).
- *Team* generated for *Digdug*, D5 (D.3).
- *Team* generated for *Sheriff*, S4 (D.4).

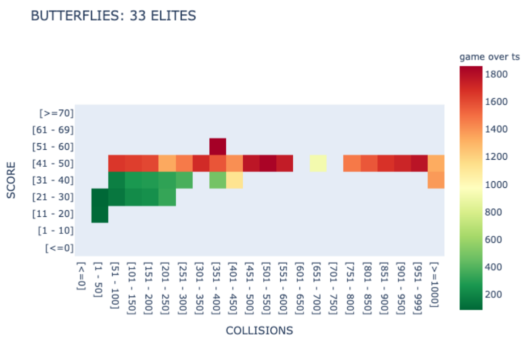
D.1 *Butterflies Team* Assemble (B2 and B3 Comparison)

Each figure contains the MAP-Elites generated for the same pair of features for *Butterflies* for two different experiments, one corresponding to B2 and the other to B3. Their results have been included side to side for easy comparison. Each graph contains the following information:

- Title showing the game and number of elites (agents) generated.
- Features (resulting stats) constituting the axis of the map.
- Heatmap representing the MAP-Elites.
- The legend represents the final average of game-ticks to EoG of each agent.

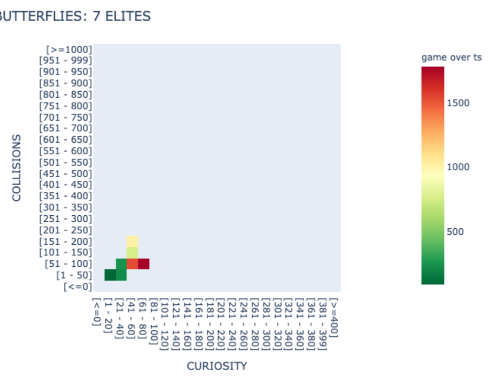


(a) B2

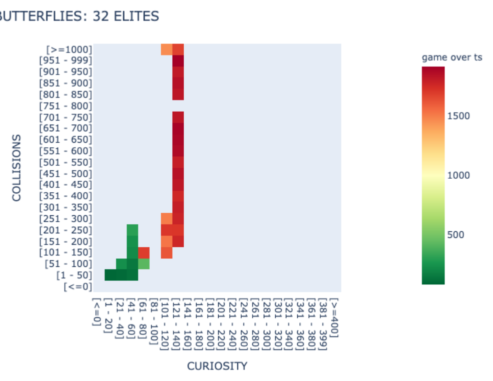


(b) B3

Figure D.1: Features: *Collisions* and *Score*.

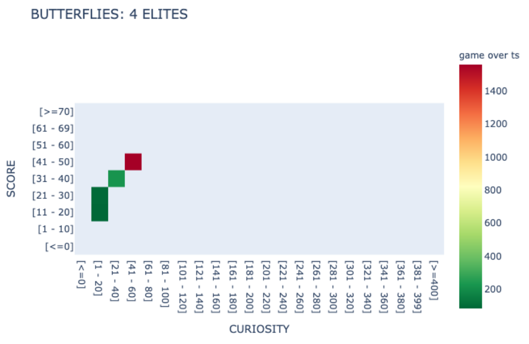


(a) B2



(b) B3

Figure D.2: Features: *Curiosity* and *Collisions*.


(a) B2

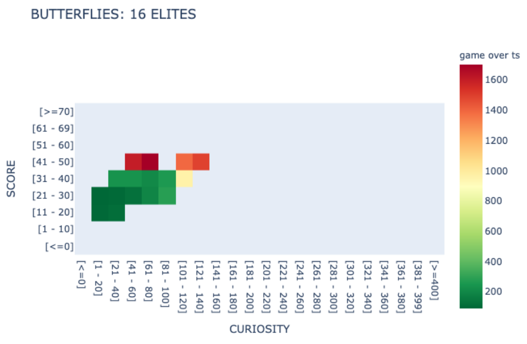
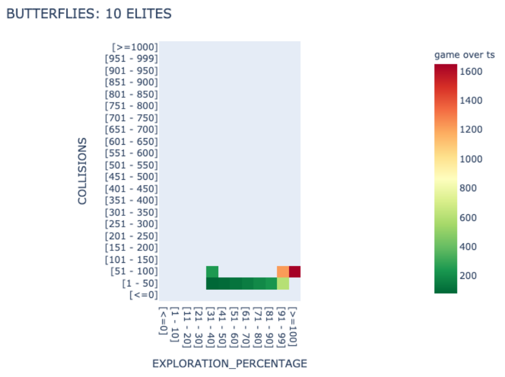

(b) B3

Figure D.3: Features: *Curiosity* and *Score*.


(a) B2

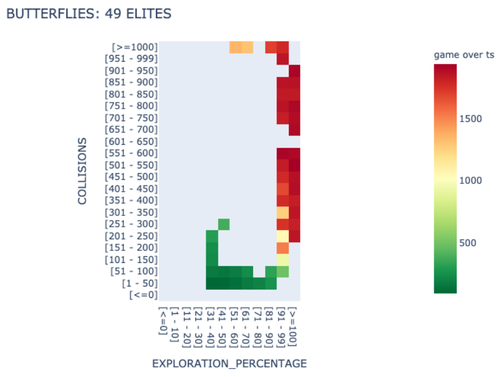
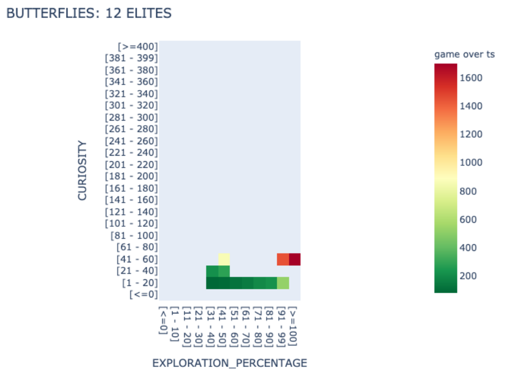

(b) B3

Figure D.4: Features: *Exploration percentage* and *Collisions*.


(a) B2

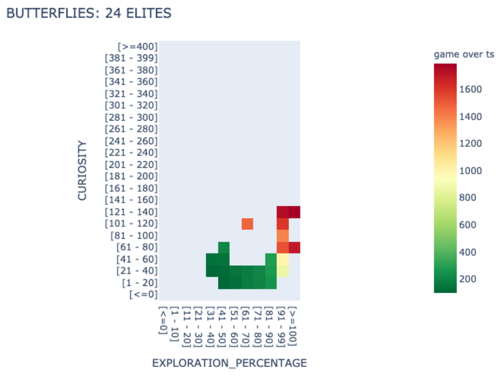
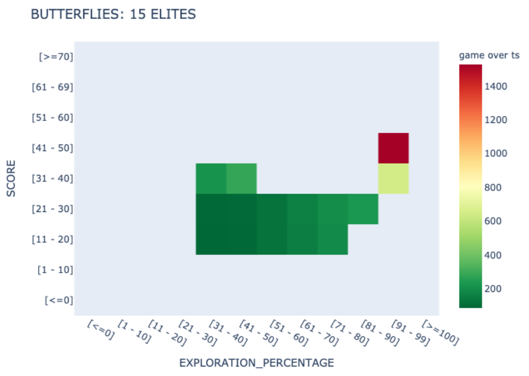
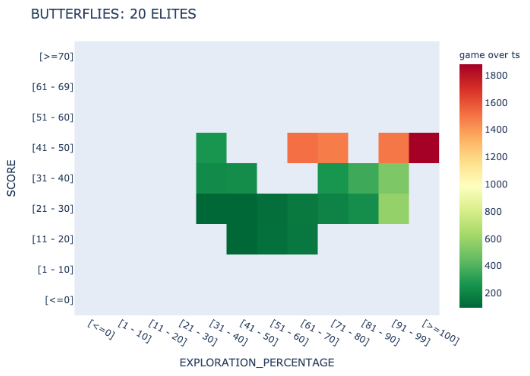

(b) B3

Figure D.5: Features: *Exploration percentage* and *Curiosity*.

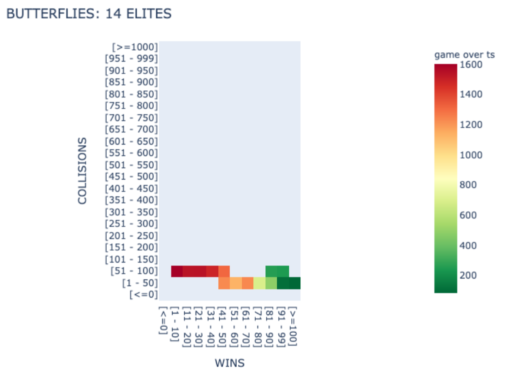


(a) B2

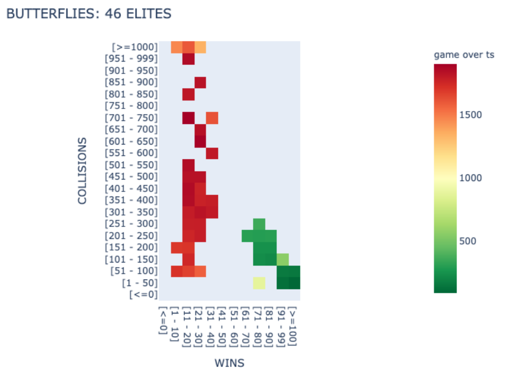


(b) B3

Figure D.6: Features: *Exploration percentage* and *Score*.

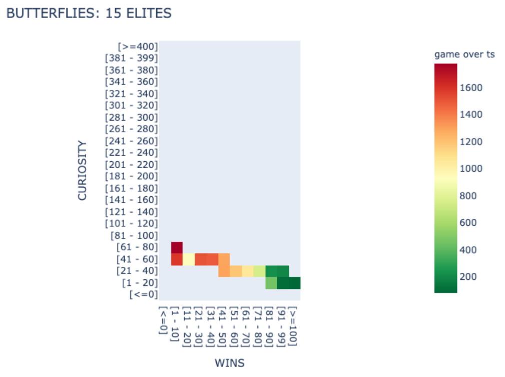


(a) B2

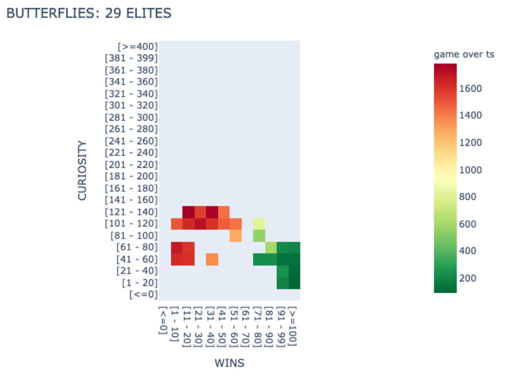


(b) B3

Figure D.7: Features: *Wins* and *Collisions*.

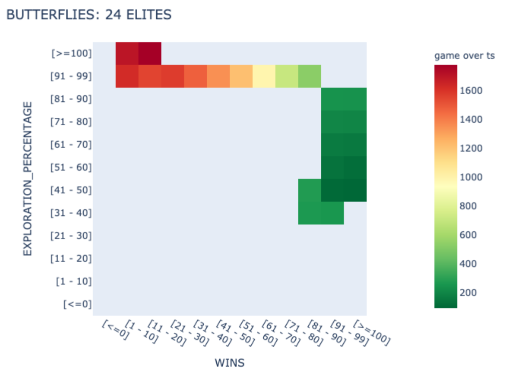


(a) B2

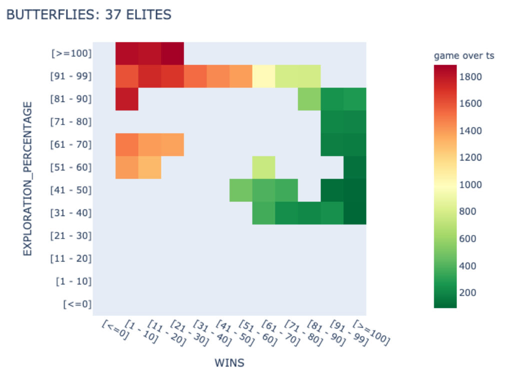


(b) B3

Figure D.8: Features: *Wins* and *Curiosity*.

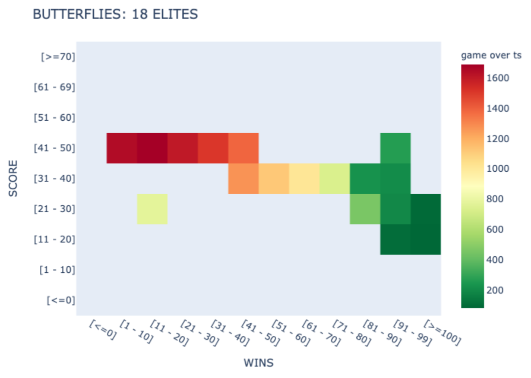


(a) B2

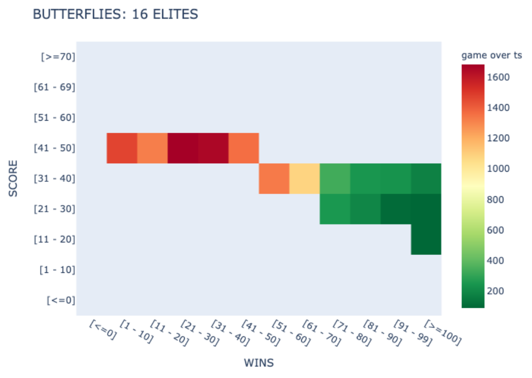


(b) B3

Figure D.9: Features: *Wins* and *Exploration percentage*.



(a) B2



(b) B3

Figure D.10: Features: *Wins* and *Score*.

D.2 *Zelda Team* Assemble

Each figure contains the MAP-Elites generated for a pair of features for *Zelda*. Each graph contains the following information:

- Title showing the game and number of elites (agents) generated.
- Features (resulting stats) constituting the axis of the map.
- Heatmap representing the MAP-Elites.
- The legend represents the final average of game-ticks to EoG of each agent.

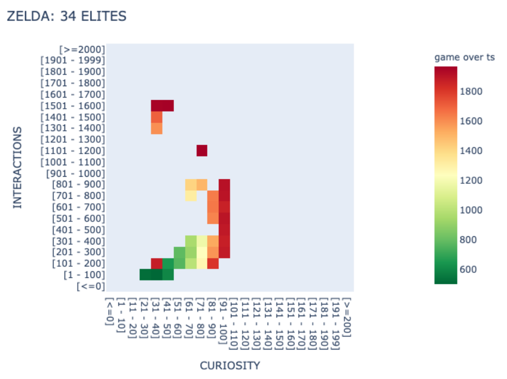


Figure D.11: Features: *Curiosity* and *Interactions.*

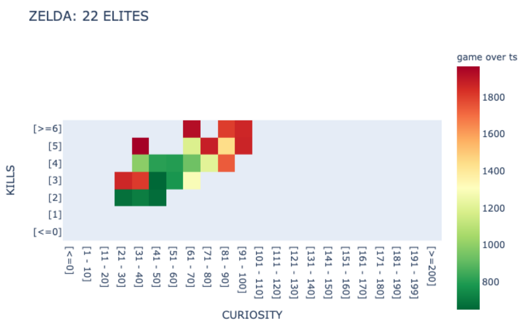


Figure D.12: Features: *Curiosity* and *Kills.*

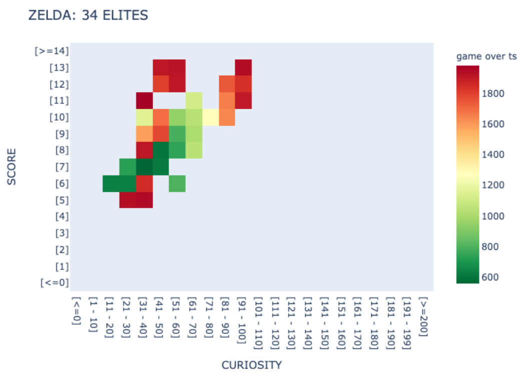


Figure D.13: Features: *Curiosity* and *Score.*

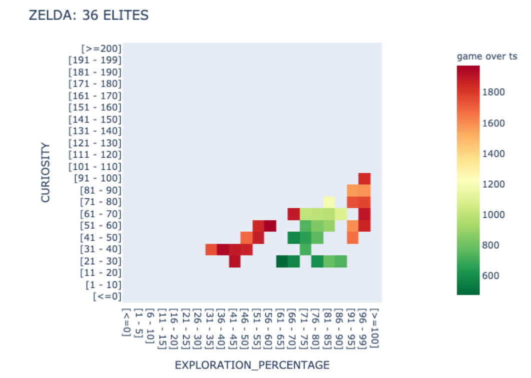


Figure D.14: Features: *Exploration percentage* and *Curiosity.*

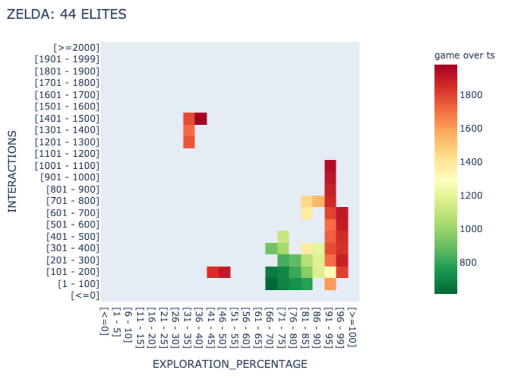


Figure D.15: Features: *Exploration percentage* and *Interactions*.

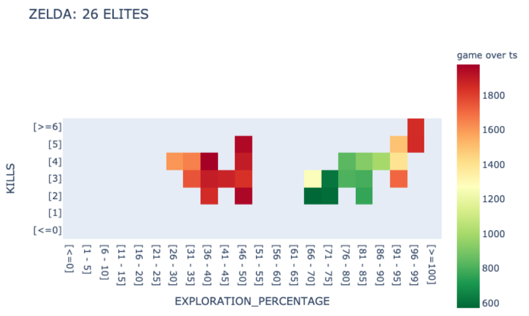


Figure D.16: Features: *Exploration percentage* and *Kills*.

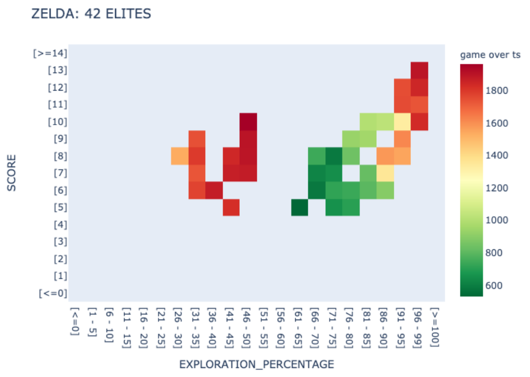


Figure D.17: Features: *Exploration percentage* and *Score*.

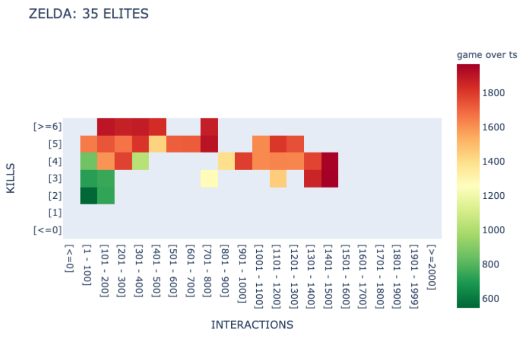


Figure D.18: Features: *Interactions* and *Kills*.

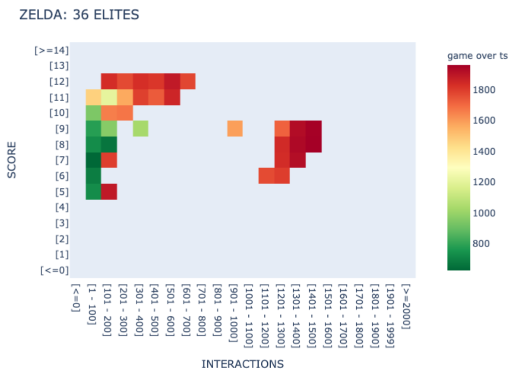


Figure D.19: Features: *Interactions* and *Score*.

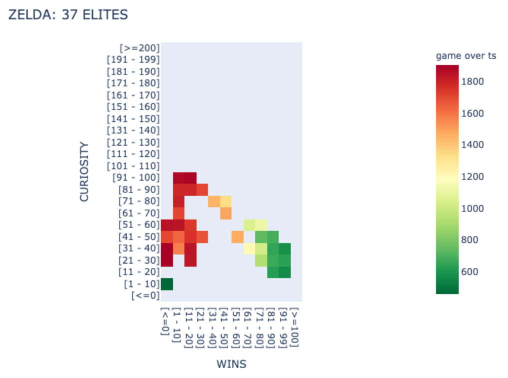


Figure D.20: Features: *Wins* and *Curiosity*.

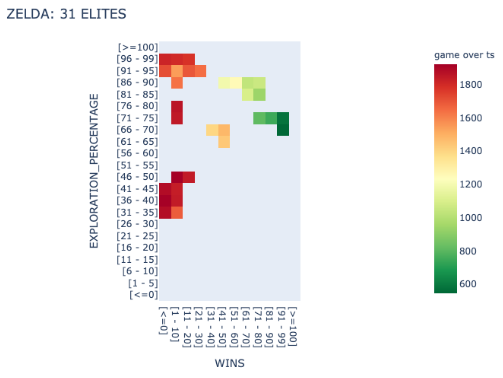


Figure D.21: Features: *Wins* and *Exploration percentage*.

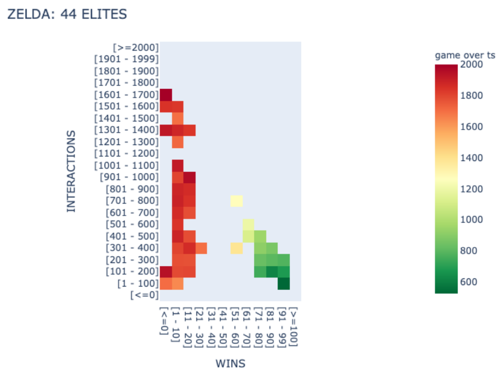


Figure D.22: Features: *Wins* and *Interactions*.

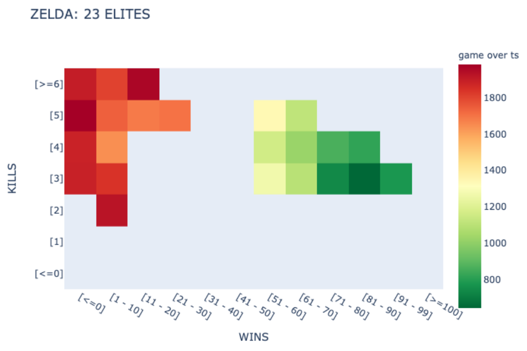


Figure D.23: Features: *Wins* and *Kills*.

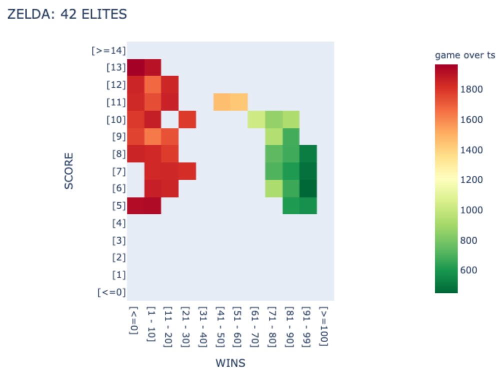


Figure D.24: Features: *Wins* and *Score*.

D.3 *Digdug Team* Assemble

Each figure contains the MAP-Elites generated for a pair of features for *Digdug*. Each graph contains the following information:

- Title showing the game and number of elites (agents) generated.
- Features (resulting stats) constituting the axis of the map.
- Heatmap representing the MAP-Elites.
- The legend represents the final average of game-ticks to EoG of each agent.

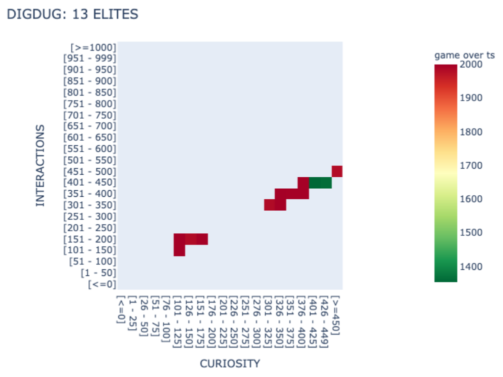


Figure D.25: Features: *Curiosity* x *Interactions*.

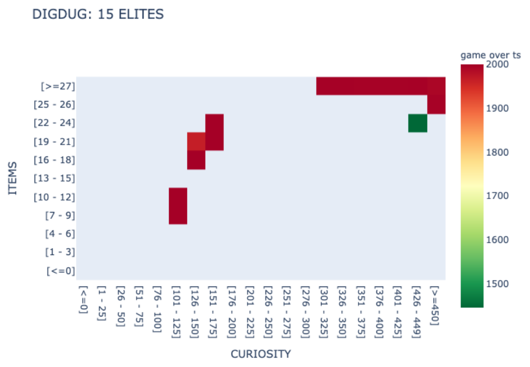


Figure D.26: Features: *Curiosity* x *Items*.

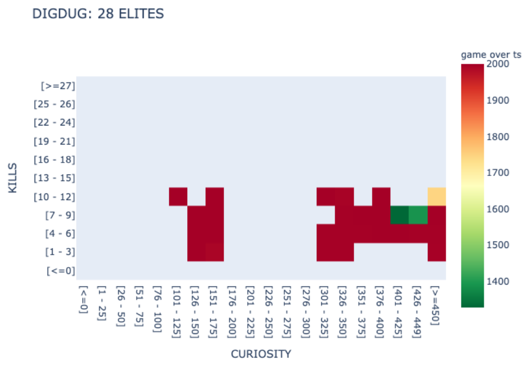


Figure D.27: Features: *Curiosity* x *Kills*.

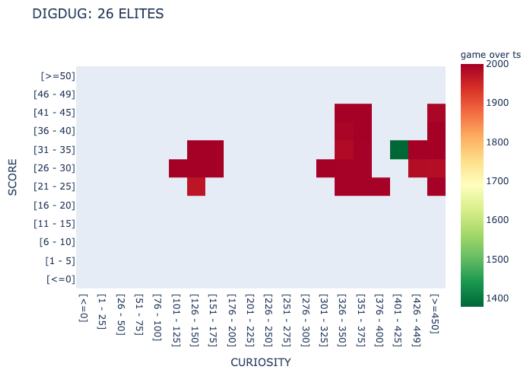


Figure D.28: Features: *Curiosity* x *Score*.

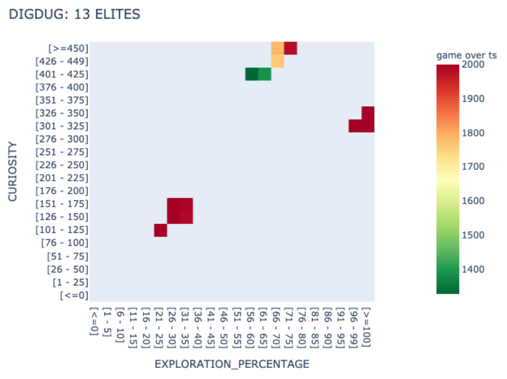


Figure D.29: Features: *Exploration percentage* x *Curiosity*.

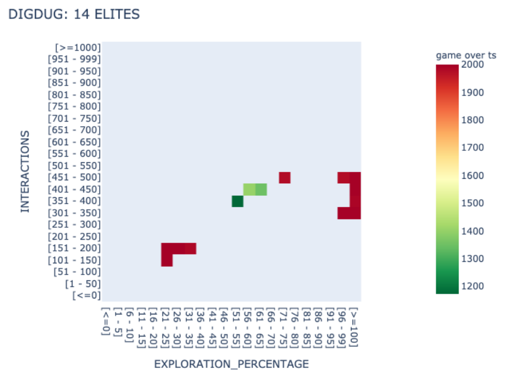


Figure D.30: Features: *Exploration percentage* x *Interactions*.

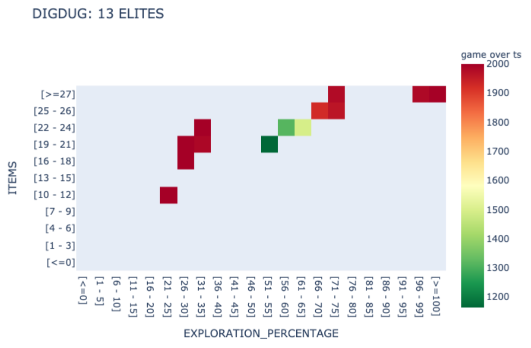


Figure D.31: Features: *Exploration percentage* x *Items*.

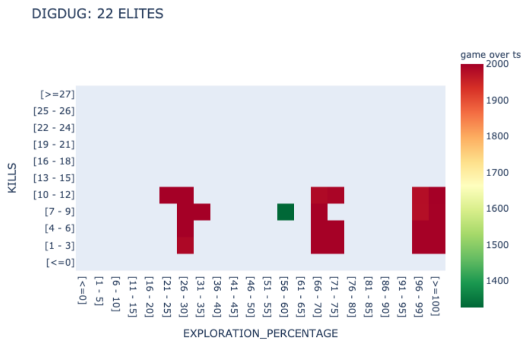


Figure D.32: Features: *Exploration percentage* x *Kills*.

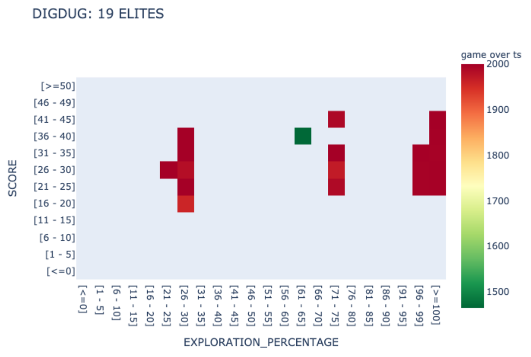


Figure D.33: Features: *Exploration percentage* x *Score*.

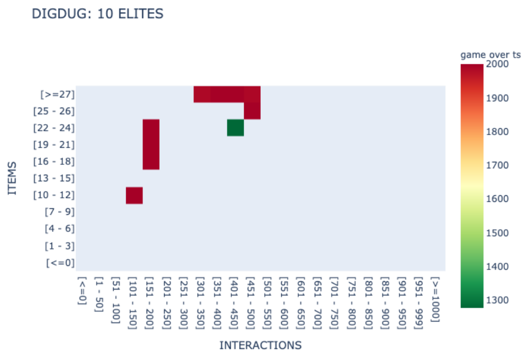


Figure D.34: Features: *Interactions* x *Items*.

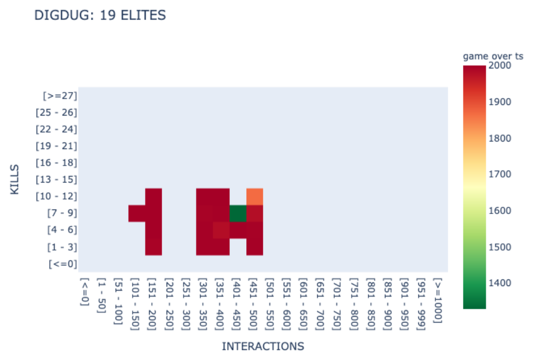


Figure D.35: Features: *Interactions* x *Kills*.

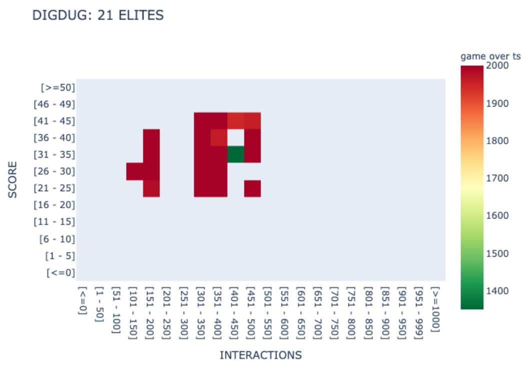


Figure D.36: Features: *Interactions* x *Score*.

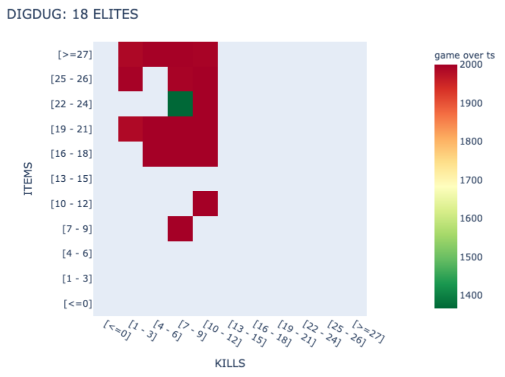


Figure D.37: Features: *Kills* x *Items*.

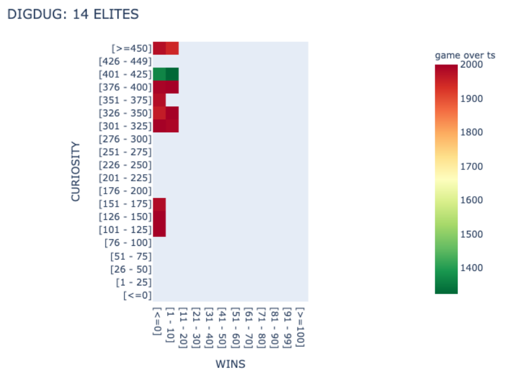


Figure D.38: Features: *Wins* x *Curiosity*.

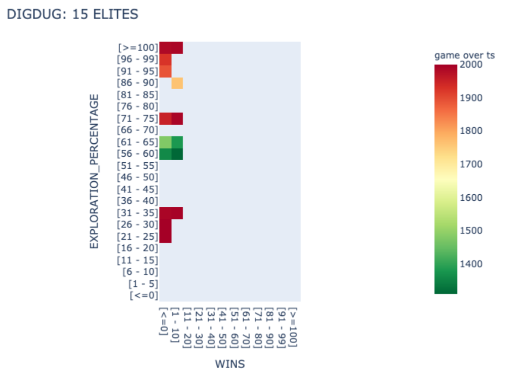


Figure D.39: Features: *Wins* x *Exploration percentage*.

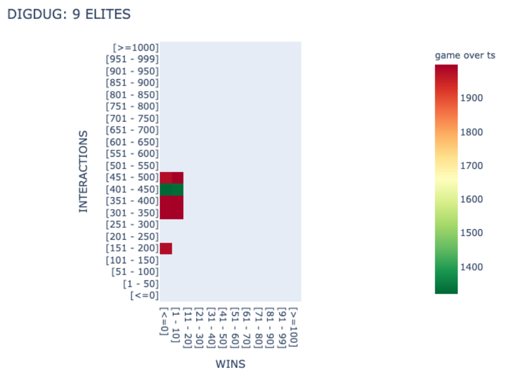


Figure D.40: Features: *Wins* x *Interactions*.

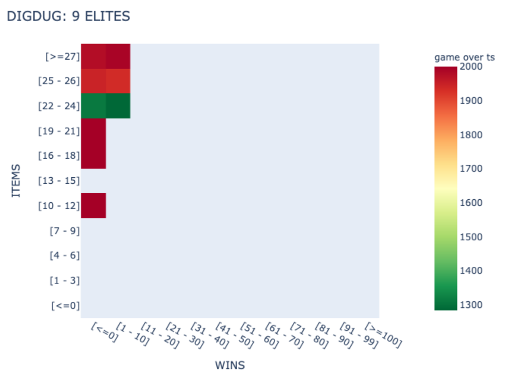


Figure D.41: Features: *Wins* x *Items*.

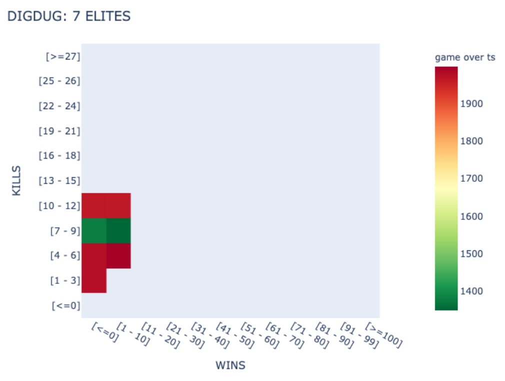


Figure D.42: Features: *Wins* x *Kills*.

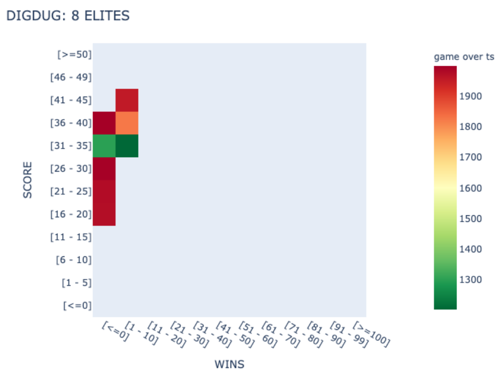


Figure D.43: Features: *Wins* x *Score*.

D.4 *Sheriff Team* Assemble

Each figure contains the MAP-Elites generated for a pair of features for *Sheriff*. Each graph contains the following information:

- Title showing the game and number of elites (agents) generated.
- Features (resulting stats) constituting the axis of the map.
- Heatmap representing the MAP-Elites.
- The legend represents the final average of game-ticks to EoG of each agent.

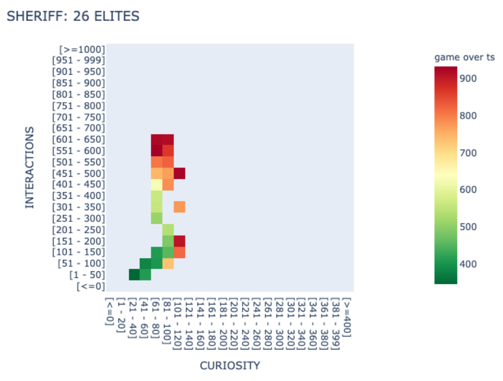


Figure D.44: Features: *Curiosity* x *Interactions.*

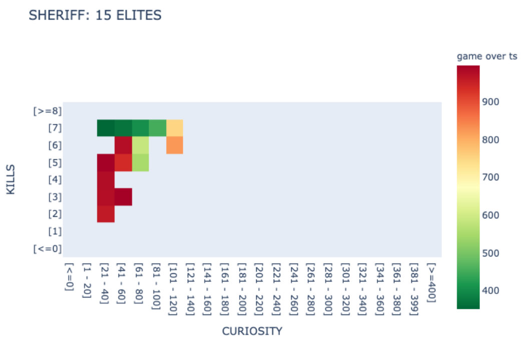


Figure D.45: Features: *Curiosity* x *Kills.*

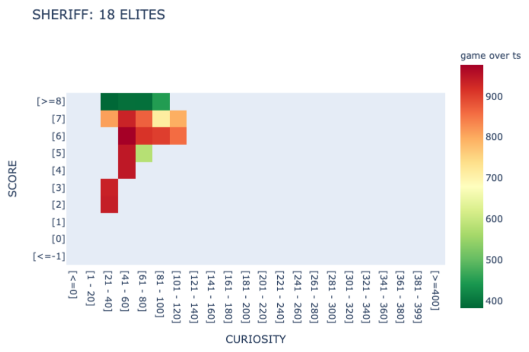


Figure D.46: Features: *Curiosity* x *Score.*

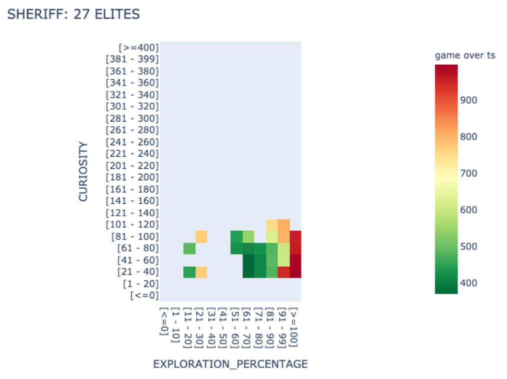


Figure D.47: Features: *Exploration percentage* x *Curiosity.*

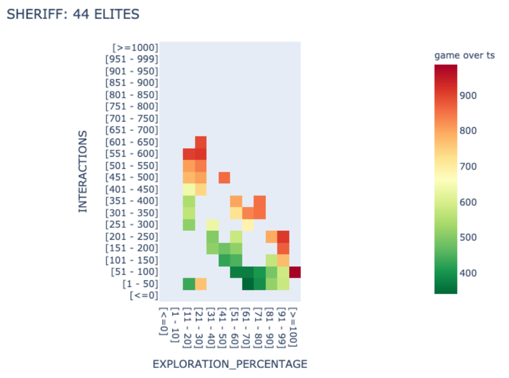


Figure D.48: Features: *Exploration percentage* x *Interactions.*

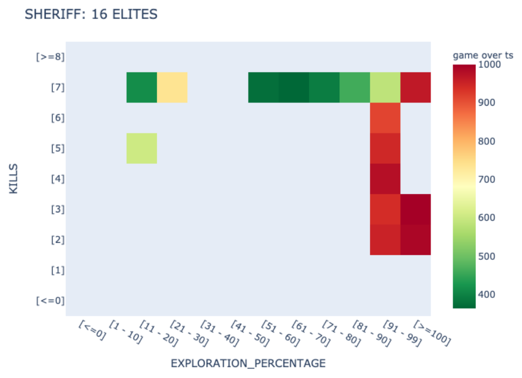


Figure D.49: Features: *Exploration percentage* x *Kills.*

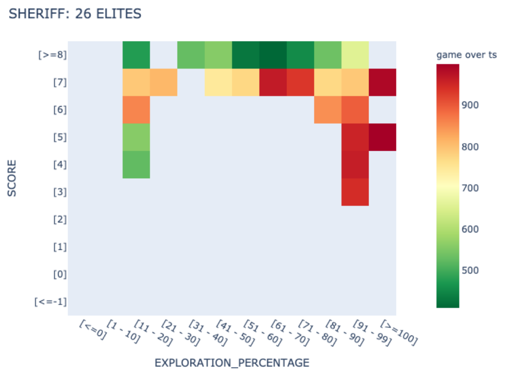


Figure D.50: Features: *Exploration percentage* x *Score.*

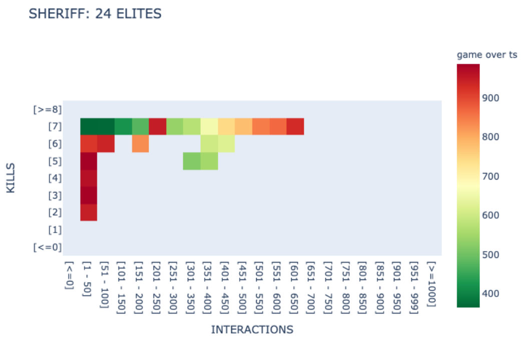


Figure D.51: Features: *Interactions* x *Kills.*

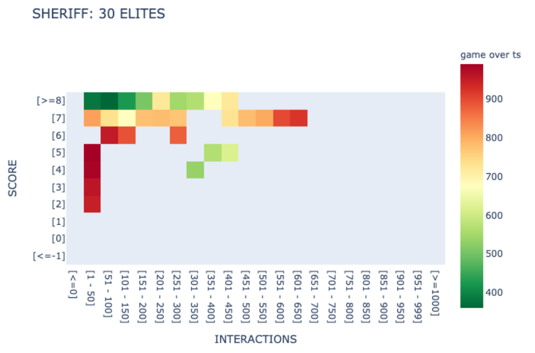


Figure D.52: Features: *Interactions* x *Score.*

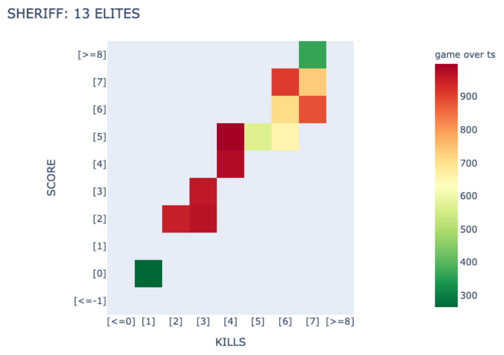


Figure D.53: Features: *Kills* x *Score.*

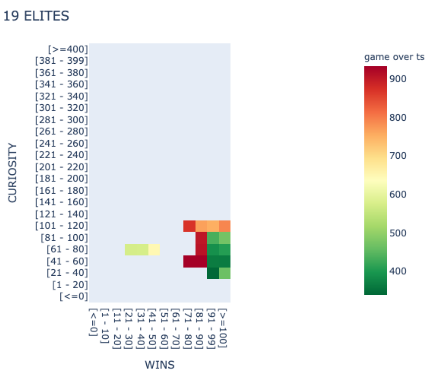


Figure D.54: Features: *Wins* x *Curiosity*.

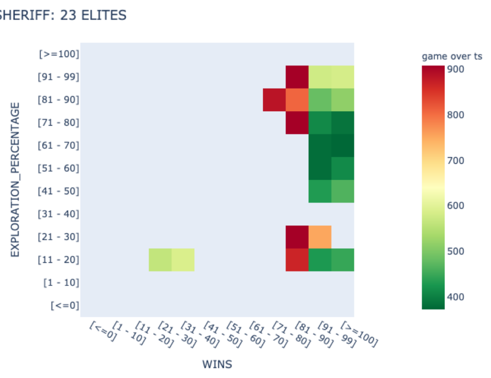


Figure D.55: Features: *Wins* x *Exploration percentage*.

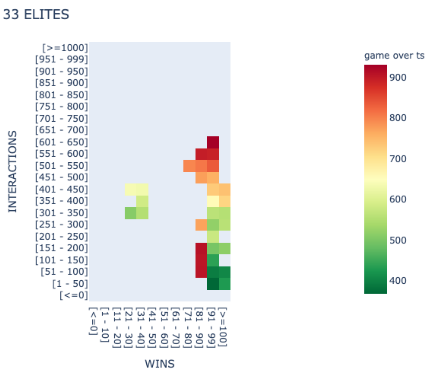


Figure D.56: Features: *Wins* x *Interactions*.

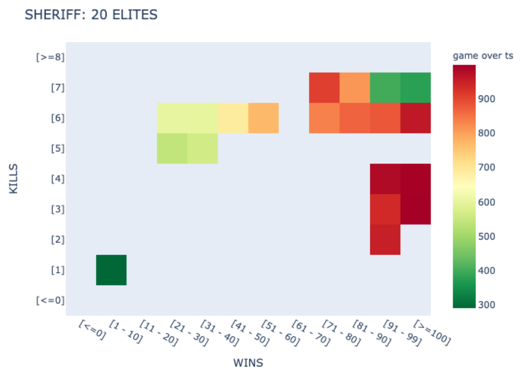


Figure D.57: Features: *Wins* x *Kills*.

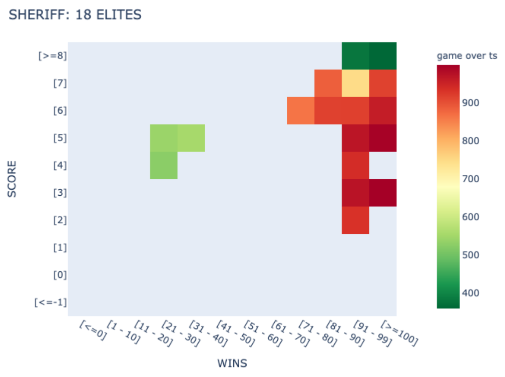


Figure D.58: Features: *Wins* x *Score*.

Appendix E

Full Results: Team Portability to New Levels

This appendix contains the results from the portability experiments described in Chapter 7, gathered by each of the games used as follows:

- *Butterflies* (E.1).
- *Zelda* (E.2).
- *Digdug* (E.3).
- *Sheriff* (E.4).

For each game, I include two types of tables. One contains the resulting stats of each *behaviour-type* agent across all levels, and the other, the collective resulting stats of every agent in each level. The stats related to the competence of the agent are highlighted.

E.1 Portability Results in *Butterflies*

Two types of tables are included. First, each table contains the final stats of a particular agent across all the levels used for *Butterflies*. Last, each table gathers the stats by level, displaying the results of the different *behaviour-type* agents in it.

In the tables, the stats related to the competence of the corresponding *behaviour-type* are highlighted. Also, information about relevant maximum values for each of the levels is included for reference.

E1 (*Low scorer*)					
Features	**lvl 7.2a**	**lvl 7.19a**	**lvl 7.19b**	**lvl 7.19c**	**lvl 7.19d**
Level Info (Max. Values)					
EoG	2000	2000	2000	2000	2000
Score	64	38	58	38	18
Gameplay Results					
EoG	95.41	102.64	120.84	81.03	78.68
Win rate	100%	100%	100%	100%	85.00%
Score	20.44	29.58	43.56	21.2	12.64
Exploration %	42.04%	44.26%	50.70%	33.77%	32.25%
Unique interactions	1.34	1.2	1.36	1.13	1.48
Curiosity	7.84	13.19	17.59	9.25	5.89
Collisions	7.89	13.33	17.84	9.29	6.25
Interactions	7.89	13.33	17.84	9.29	6.25

Table E.1: *Butterflies*: Per-level portability results for E1 (*Low scorer*).

E2 (*High scorer*)					
Features	**lvl 7.2a**	**lvl 7.19a**	**lvl 7.19b**	**lvl 7.19c**	**lvl 7.19d**
Level Info (Max. Values)					
EoG	2000	2000	2000	2000	2000
Score	64	38	58	38	18
Gameplay Results					
EoG	1866.53	1927.34	1753.88	1988.18	959.46
Win rate	33%	32%	25%	28.99%	9%
Score	47.76	30.78	51.1	22.58	11.86
Exploration %	98.45%	99.20%	99.29%	100%	78.05%
Unique interactions	2	2	2	2	1.94
Curiosity	125.97	104.28	127.64	85.56	74.79
Collisions	380.85	315.50	312.38	265	254.69
Interactions	380.85	315.50	312.38	265	254.69

Table E.2: *Butterflies*: Per-level portability results for E2 (*High scorer*).

E3 (*High curiosity*)					
Features	**lvl 7.2a**	**lvl 7.19a**	**lvl 7.19b**	**lvl 7.19c**	**lvl 7.19d**
Level Info (Max. Values)					
EoG	2000	2000	2000	2000	2000
Walls (trees)	102	93	91	83	99
Butterflies	6	12	12	10	7
Gameplay Results					
EoG	1704.40	1981.48	1611.16	1987.38	862.94
Win rate	6%	5%	0%	2%	4%
Score	46.08	30.98	51.7	22.16	11.22
Exploration %	68.05%	60.77%	64.66%	45.66%	42.82%
Unique interactions	2	2	2	2	1.91
Curiosity	116.53	98.99	116.38	82.13	76.38
Collisions	1436.82	1754.85	1362.33	1780.46	715.56
Interactions	1436.82	1754.85	1362.33	1780.46	715.56

Table E.3: *Butterflies*: Per-level portability results for E3 (*High curiosity*).

E4 (*Speed-runner*)					
Features	**lvl 7.2a**	**lvl 7.19a**	**lvl 7.19b**	**lvl 7.19c**	**lvl 7.19d**
Level Info (Max. Values)					
EoG	2000	2000	2000	2000	2000
Gameplay Results					
EoG	102.15	103.49	114.38	73.61	61.13
Win rate	100%	100%	99%	100%	89%
Score	21.1	29.68	46.44	21.32	13.28
Exploration %	41.17%	42.08%	45.85%	29.02%	25.60%
Unique interactions	1.96	1.91	1.97	1.68	1.8
Curiosity	11.79	14.91	20.95	10.2	6.66
Collisions	12.2	15.26	21.77	10.35	6.97
Interactions	12.2	15.26	21.77	10.35	6.97

Table E.4: *Butterflies*: Per-level portability results for E4 (*Speed-runner*).

E5 (*High explorer*)					
Features	**lvl 7.2a**	**lvl 7.19a**	**lvl 7.19b**	**lvl 7.19c**	**lvl 7.19d**
Level Info (Max. Values)					
EoG	2000	2000	2000	2000	2000
Open locations	206	215	217	225	209
Gameplay Results					
EoG	1886.98	1951.63	1753.26	1967.24	793.78
Win rate	21%	27%	6.99%	28.99%	8%
Score	46.96	31.26	52.24	22.62	12.54
Exploration %	99.29%	99.30%	99.33%	99.77%	74.56%
Unique interactions	2	2	2	2	1.97
Curiosity	128.95	106.08	129.88	84.93	73.98
Collisions	524.31	442.26	423.59	362.27	258.09
Interactions	524.31	442.26	423.59	362.27	258.09

Table E.5: *Butterflies*: Per-level portability results for E5 (*High explorer*).

E6 (*Walls interaction*)					
Features	**lvl 7.2a**	**lvl 7.19a**	**lvl 7.19b**	**lvl 7.19c**	**lvl 7.19d**
Level Info (Max. Values)					
EoG	2000	2000	2000	2000	2000
Walls (trees)	102	93	91	83	99
Butterflies	6	12	12	10	7
Open locations	206	215	217	225	209
Gameplay Results					
EoG	1557.9	1924.17	1612.95	1990.3	860.37
Win rate	10%	12%	3%	3%	5%
Score	44.54	31.12	51.78	22.34	11.24
Exploration %	67.43%	60.28%	68.15%	44.60%	43.25%
Unique interactions	2	2	2	2	1.96
Curiosity	114.59	96.53	120.46	79.86	75.16
Collisions	1296.68	1683.06	1351.35	1801.41	708.53
Interactions	1296.68	1683.06	1351.35	1801.41	708.53

Table E.6: *Butterflies*: Per-level portability results for E6 (*Walls interaction*).

Level 7.2a							
Info							
Butterflies	6						
Cocoons	27						
Open locations	206						
Walls (trees)	102						
Gameplay Results							
Features	**Max**	**E1**	**E2**	**E3**	**E4**	**E5**	**E6**
EoG	**2000**	95.41	1866.53	1704.40	102.15	1886.98	1557.9
Win rate	**-**	100%	33%	6%	100%	21%	10%
Score	**64**	20.44	47.76	46.08	21.1	46.96	44.54
Exploration %	**-**	42.04%	98.45%	68.05%	41.17%	99.29%	67.43%
Unique interactions	**-**	1.34	2	2	1.96	2	2
Curiosity	**-**	7.84	125.97	116.53	11.79	128.95	114.59
Collisions	**-**	7.89	380.85	1436.82	12.2	524.31	1296.68
Interactions	**-**	7.89	380.85	1436.82	12.2	524.31	1296.68

Table E.7: *Butterflies*: Per-agent portability results for Level 7.2a (original).

Level 7.19a							
Info							
Butterflies	12						
Cocoons	8						
Open locations	215						
Walls (trees)	93						
Gameplay Results							
Features	**Max**	**E1**	**E2**	**E3**	**E4**	**E5**	**E6**
EoG	**2000**	102.64	1927.34	1981.48	103.49	1951.63	1924.17
Win rate	**-**	100%	32%	5%	100%	27%	12%
Score	**38**	29.58	30.78	30.98	29.68	31.269	31.12
Exploration %	**-**	44.26%	99.20%	60.77%	42.08%	99.30%	60.28%
Unique interactions	**-**	1.2	2	2	1.91	2	2
Curiosity	**-**	13.19	104.28	98.99	14.91	106.08	96.53
Collisions	**-**	13.33	315.50	1754.85	15.26	442.26	1683.06
Interactions	**-**	13.33	315.50	1754.85	15.26	442.26	1683.06

Table E.8: *Butterflies*: Per-agent portability results for Level 7.19a.

Level 7.19b							
Info							
Butterflies	12						
Cocoons	18						
Open locations	217						
Walls (trees)	91						
Gameplay Results							
Features	**Max**	**E1**	**E2**	**E3**	**E4**	**E5**	**E6**
EoG	**2000**	120.84	1753.88	1611.16	114.38	1753.26	1612.95
Win rate	**-**	100%	25%	0%	99%	6.99%	3%
Score	**58**	43.56	51.1	51.7	46.44	52.24	51.78
Exploration %	**-**	50.70%	99.29%	64.66%	45.85%	99.33%	68.15%
Unique interactions	**-**	1.36	2	2	1.97	2	2
Curiosity	**-**	17.59	127.64	116.38	20.95	129.88	120.46
Collisions	**-**	17.84	312.38	1362.33	21.77	423.59	1351.35
Interactions	**-**	17.84	312.38	1362.33	21.77	423.59	1351.35

Table E.9: *Butterflies*: Per-agent portability results for Level 7.19b.

Level 7.19c							
Info							
Butterflies	10						
Cocoons	10						
Open locations	225						
Walls (trees)	83						
Gameplay Results							
Features	**Max**	**E1**	**E2**	**E3**	**E4**	**E5**	**E6**
EoG	**2000**	81.03	1988.18	1987.38	73.61	1967.24	1990.3
Win rate	**-**	100%	28.99%	2%	100%	28.99%	3%
Score	**38**	21.2	22.58	22.16	21.32	22.62	22.34
Exploration %	**-**	33.77%	100%	45.66%	29.02%	99.77%	44.6%
Unique interactions	**-**	1.13	2	2	1.68	2	2
Curiosity	**-**	9.25	85.56	82.13	10.2	84.93	79.86
Collisions	**-**	9.29	265	1780.46	10.35	362.27	1801.41
Interactions	**-**	9.29	265	1780.46	10.35	362.27	1801.41

Table E.10: *Butterflies*: Per-agent portability results for Level 7.19c.

Level 7.19d	
Info	
Butterflies	7
Cocoons	3
Open locations	209
Walls (trees)	99

Gameplay Results							
Features	**Max**	**E1**	**E2**	**E3**	**E4**	**E5**	**E6**
EoG	**2000**	78.68	959.46	862.94	61.13	793.78	860.37
Win rate	**-**	85%	9%	4%	89%	8%	5%
Score	**18**	12.64	11.86	11.22	13.28	12.54	11.24
Exploration %	**-**	32.25%	78.05%	42.82%	25.60%	74.56%	43.25%
Unique interactions	**-**	1.48	1.94	1.91	1.8	1.97	1.96
Curiosity	**-**	5.89	74.79	76.38	6.66	73.98	75.16
Collisions	**-**	6.25	254.69	715.56	6.97	258.09	708.53
Interactions	**-**	6.25	254.69	715.56	6.97	258.09	708.53

Table E.11: *Butterflies*: Per-agent portability results for Level 7.19d.

E.2 Portability results in *Zelda*

Two types of tables are included. First, each table contains the final stats of a particular agent across all the levels used for *Zelda*. Last, each table gathers the stats by level, displaying the results of the different *behaviour-type* agents in it.

In the tables, the stats related to the competence of the corresponding *behaviour-type* are highlighted. Also, information about relevant maximum values for each of the levels is included for reference.

E1 (*High scorer*)					
Features	**lvl 7.2b**	**lvl 7.28a**	**lvl 7.28b**	**lvl 7.28c**	**lvl 7.28d**
Level Info (Max. Values)					
EoG	2000	2000	2000	2000	2000
Score	14	14	14	18	14
Gameplay Results					
EoG	1892.14	1871.92	1794.14	1942.7	1741.66
Win rate	0%	0%	0%	1%	0%
Score	12.25	12.11	11.70	16.44	11.55
Exploration %	96.27%	96.12%	93.87%	97.35%	91.62%
Unique interactions	5.07	5	4.8	5.22	3.89
Curiosity	57.75	51.06	63.43	48.66	50.53
Collisions	90.87	74.09	106.94	62.07	83.28
Hits	5.65	5.61	5.45	7.72	5.31
Interactions	96.52	79.7	112.39	69.79	88.59
Kills	5.65	5.61	5.45	7.72	5.31
Items	0.97	0.95	0.90	0.98	0.96

Table E.12: *Zelda*: Per-level portability results for E1 (*High scorer*).

E2 (*Speed-runner*)					
Features	**lvl 7.2b**	**lvl 7.28a**	**lvl 7.28b**	**lvl 7.28c**	**lvl 7.28d**
Level Info (Max. Values)					
EoG	2000	2000	2000	2000	2000
Gameplay Results					
EoG	617.18	1187.98	1242.26	689.03	450.86
Win rate	90%	76%	65%	89%	96%
Score	6.52	9.17	9.68	8.74	5.95
Exploration %	72.65%	94.17%	92.42%	76.76%	58.175%
Unique interactions	4.02	5.19	5.13	4.64	3.53
Curiosity	42.74	69.57	75.72	41.74	27.72
Collisions	87.05	178.73	223.01	76.32	57.02
Hits	2.38	3.79	4.04	3.51	2.05
Interactions	89.43	182.52	227.04	79.82	59.06
Kills	2.38	3.79	4.04	3.51	2.05
Items	0.96	0.96	0.97	0.95	0.98

Table E.13: *Zelda*: Per-level portability results for E2 (*Speed-runner*).

E3 (*High explorer*)					
Features	**lvl 7.2b**	**lvl 7.28a**	**lvl 7.28b**	**lvl 7.28c**	**lvl 7.28d**
Level Info (Max. Values)					
EoG	2000	2000	2000	2000	2000
Open locations	126	130	114	138	120
Gameplay Results					
EoG	1905.62	1845.61	1872.29	1941.68	1808.31
Win rate	10%	3%	3%	11%	8%
Score	11.39	11.13	11.25	15.38	11.39
Exploration %	98.03%	95.36%	95.25%	98.31%	93.33%
Unique interactions	4.9	5.05	5.07	5.34	3.93
Curiosity	64.12	57.31	72.87	57.97	59
Collisions	102.91	84.35	130.35	81.57	101.84
Hits	5.18	5.12	5.18	7.13	5.2
Interactions	108.09	89.46	135.53	88.80	107.04
Kills	5.18	5.12	5.18	7.13	5.2
Items	0.99	0.95	0.95	0.99	0.97

Table E.14: *Zelda*: Per-level portability results for E3 (*High explorer*).

E4 (*Low killer*)					
Features	**lvl 7.2b**	**lvl 7.28a**	**lvl 7.28b**	**lvl 7.28c**	**lvl 7.28d**
Level Info (Max. Values)					
EoG	2000	2000	2000	2000	2000
NPCs (monsters)	6	6	6	8	6
Gameplay Results					
EoG	548.03	1277.5	1162.81	985.01	445.86
Win rate	94%	73%	60%	79%	98%
Score	5.06	8.15	7.32	8.59	5.47
Exploration %	65.06%	94.11%	86.64%	83.23%	58.30%
Unique interactions	3.19	4.61	4.59	4.21	3.21
Curiosity	24.78	52.02	55.54	36.63	18.34
Collisions	34.73	80.79	99.53	54.18	26.9
Hits	1.62	3.25	2.97	3.53	1.79
Interactions	36.35	84.04	102.5	57.71	28.68
Kills	1.62	3.25	2.97	3.53	1.79
Items	0.97	0.97	0.91	0.89	1

Table E.15: *Zelda*: Per-level portability results for E4 (*Low killer*).

E5 (*High killer + high explorer*)					
Features	**lvl 7.2b**	**lvl 7.28a**	**lvl 7.28b**	**lvl 7.28c**	**lvl 7.28d**
Level Info (Max. Values)					
EoG	2000	2000	2000	2000	2000
NPCs (monsters)	6	6	6	8	6
Open locations	126	130	114	138	120
Gameplay Results					
EoG	1943.6	1920.91	1829.95	1964.87	1849.12
Win rate	1%	1%	4%	13%	12%
Score	12.35	12.26	11.75	16.53	11.89
Exploration %	97.19%	97.93%	94.71%	98.61%	94%
Unique interactions	5.06	5.50	5.26	5.37	3.95
Curiosity	81.21	76.27	86.48	72.96	69.74
Collisions	132.98	116.94	160.69	103.38	131.33
Hits	5.68	5.66	5.44	7.71	5.41
Interactions	138.65	122.60	166.13	111.09	136.74
Kills	5.68	5.66	5.44	7.71	5.41
Items	0.99	0.99	0.94	0.99	1

Table E.16: *Zelda*: Per-level portability results for E5 (*High killer + high explorer*).

E6 (*High killer + low explorer*)					
Features	**lvl 7.2b**	**lvl 7.28a**	**lvl 7.28b**	**lvl 7.28c**	**lvl 7.28d**
Level Info (Max. Values)					
EoG	2000	2000	2000	2000	2000
NPCs (monsters)	6	6	6	8	6
Open locations	126	130	114	138	120
Gameplay Results					
EoG	2000	2000	1949.72	1904.54	1979.58
Win rate	0%	0%	0%	0%	0%
Score	9.94	10.43	8.51	14.74	8.11
Exploration %	48.38%	50.66%	42.25%	54.79%	35.73%
Unique interactions	3.94	4.61	3.87	4.34	3.44
Curiosity	33.34	32.13	32.90	34.85	27.44
Collisions	124.97	108	145.86	100.17	128.42
Hits	4.78	5.17	4.21	7.26	3.77
Interactions	129.75	113.17	150.07	107.43	132.19
Kills	4.78	5.17	4.21	7.26	3.77
Items	0.35	0.07	0.08	0.22	0.58

Table E.17: *Zelda*: Per-level portability results for E6 (*High killer + low explorer*).

Level 7.2b							
Info							
Open locations	126						
Walls	90						
Gameplay Results							
Features	**Max**	**E1**	**E2**	**E3**	**E4**	**E5**	**E6**
EoG	**2000**	1892.14	617.18	1905.62	548.03	1943.6	2000
Win rate	**-**	0%	90%	10%	94%	1%	0%
Score	**14**	12.25	6.52	11.39	5.06	12.35	9.94
Exploration %	**-**	96.27%	72.65%	98.03%	65.06%	97.19%	48.38%
Unique interactions	**-**	5.07	4.02	4.9	3.19	5.06	3.94
Curiosity	**-**	57.75	42.74	64.12	24.78	81.21	33.34
Collisions	**-**	90.87	87.05	102.91	34.73	132.98	124.97
Hits	**-**	5.65	2.38	5.18	1.62	5.68	4.78
Interactions	**-**	96.52	89.43	108.09	36.35	138.65	129.75
Kills	**6**	5.65	2.38	5.18	1.62	5.68	4.78
Items	**1**	0.97	0.96	0.99	0.97	0.99	0.35

Table E.18: *Zelda*: Per-agent portability results for Level 7.2b (original).

Level 7.28a							
Info							
Open locations	130						
Walls	86						
Gameplay Results							
Features	**Max**	**E1**	**E2**	**E3**	**E4**	**E5**	**E6**
EoG	**2000**	1871.92	1187.98	1845.61	1277.5	1920.91	2000
Win rate	**-**	0%	76%	3%	73%	1%	0%
Score	**14**	12.11	9.17	11.13	8.15	12.26	10.43
Exploration %	**-**	96.12%	94.17%	95.36%	94.11%	97.93%	50.66%
Unique interactions	**-**	5	5.19	5.05	4.61	5.50	4.61
Curiosity	**-**	51.06	69.57	57.31	52.02	76.27	32.13
Collisions	**-**	74.09	178.73	84.35	80.79	116.94	108
Hits	**-**	5.61	3.79	5.12	3.25	5.66	5.17
Interactions	**-**	79.7	182.52	89.46	84.04	122.60	113.17
Kills	**6**	5.61	3.79	5.12	3.25	5.66	5.17
Items	**1**	0.95	0.96	0.95	0.97	0.99	0.07

Table E.19: *Zelda*: Per-agent portability results for Level 7.28a.

Level 7.28b							
Info							
Open locations	114						
Walls	102						
Gameplay Results							
Features	**Max**	**E1**	**E2**	**E3**	**E4**	**E5**	**E6**
EoG	**2000**	1794.14	1242.26	1872.29	1162.81	1829.95	1949.72
Win rate	**-**	0%	65%	3%	60%	4%	0%
Score	**14**	11.67	9.68	11.25	7.32	11.75	8.51
Exploration %	**-**	93.87%	92.42%	95.25%	86.64%	94.71%	42.25%
Unique interactions	**-**	4.8	5.13	5.07	4.59	5.26	3.87
Curiosity	**-**	63.43	75.72	72.87	55.54	86.48	32.90
Collisions	**-**	106.94	223.01	130.35	99.53	160.69	145.86
Hits	**-**	5.45	4.04	5.18	2.97	5.44	4.21
Interactions	**-**	112.39	227.04	135.53	102.5	166.13	150.07
Kills	**6**	5.45	4.04	5.18	2.97	5.44	4.21
Items	**1**	0.90	0.97	0.95	0.91	0.94	0.08

Table E.20: *Zelda*: Per-agent portability results for Level 7.28b.

Level 7.28c							
Info							
Open locations	138						
Walls	78						
Gameplay Results							
Features	**Max**	**E1**	**E2**	**E3**	**E4**	**E5**	**E6**
EoG	**2000**	1942.7	689.03	1941.68	985.01	1964.87	1904.54
Win rate	**-**	1%	89%	11%	79%	13%	0%
Score	**18**	16.44	8.74	15.38	8.59	16.53	14.74
Exploration %	**-**	97.35%	76.76%	98.31%	83.23%	98.61%	54.79%
Unique interactions	**-**	5.22	4.64	5.34	4.21	5.37	4.34
Curiosity	**-**	48.66	41.74	57.97	36.63	72.96	34.85
Collisions	**-**	62.07	76.32	81.57	54.18	103.38	100.17
Hits	**-**	7.72	3.51	7.13	3.53	7.71	7.26
Interactions	**-**	69.79	79.82	88.70	57.71	111.09	107.43
Kills	**8**	7.72	3.51	7.13	3.53	7.71	7.26
Items	**1**	0.98	0.95	0.99	0.89	0.99	0.22

Table E.21: *Zelda*: Per-agent portability results for Level 7.28c.

Level 7.28d							
Info							
Open locations	120						
Walls	96						
Gameplay Results							
Features	**Max**	**E1**	**E2**	**E3**	**E4**	**E5**	**E6**
EoG	**2000**	1741.66	450.86	1808.31	445.86	1849.12	1979.58
Win rate	**-**	0%	96%	8%	98%	12%	0%
Score	**14**	11.55	5.95	11.39	5.47	11.89	8.11
Exploration %	**-**	91.62%	58.175%	93.33%	58.30%	94%	35.73%
Unique interactions	**-**	3.89	3.53	3.93	3.21	3.95	3.44
Curiosity	**-**	50.53	27.72	59	18.34	69.74	27.44
Collisions	**-**	83.28	57.02	101.84	26.9	131.33	128.42
Hits	**-**	5.31	2.05	5.2	1.79	5.41	3.77
Interactions	**-**	88.59	59.06	107.04	28.68	136.74	132.19
Kills	**6**	5.31	2.05	5.2	1.79	5.41	3.77
Items	**1**	0.96	0.98	0.97	1	1	0.58

Table E.22: *Zelda*: Per-agent portability results for Level 7.28d.

E.3 Portability results in *Digdug*

Two types of tables are included. First, each table contains the final stats of a particular agent across all the levels used for *Digdug*. Last, each table gathers the stats by level, displaying the results of the different *behaviour-type* agents in it.

In the tables, the stats related to the competence of the corresponding *behaviour-type* are highlighted. Also, information about relevant maximum values for each of the levels is included for reference.

E1 (*High collector + high killer*)					
Features	**lvl 7.2c**	**lvl 7.36a**	**lvl 7.36b**	**lvl 7.36c**	**lvl 7.36d**
Level Info (Max. Values)					
EoG	2000	2000	2000	2000	2000
NPCs (monsters)	12	16	13	5	14
Items	27	19	41	8	50
Gameplay Results					
EoG	1999.619	1999.73	1999.77	1999.94	1999.57
Win rate	6.99%	1%	2%	2%	6%
Score	41.57	44.05	58.2	11.83	51.02
Exploration %	73.19%	74.00%	74.03%	67.25%	57.01%
Unique interactions	7	6.94	7	6.74	7
Curiosity	476.84	494.16	454.66	367.8	282.05
Collisions	208.37	206.12	206.63	169.57	146.84
Hits	275.42	297.7	255.12	200.11	138.69
Interactions	483.78	503.82	461.75	369.67	285.53
Kills	10.64	13.74	11.56	3.88	12.68
Items	26.92	18.91	40.76	7.95	49.90

Table E.23: *Digdug*: Per-level portability results for E1 (*High collector + high killer*).

E2 (*High collector + low killer*)					
Features	**lvl 7.2c**	**lvl 7.36a**	**lvl 7.36b**	**lvl 7.36c**	**lvl 7.36d**
Level Info (Max. Values)					
EoG	2000	2000	2000	2000	2000
NPCs (monsters)	12	16	13	5	14
Items	27	19	41	8	50
Gameplay Results					
EoG	2000	2000	2000	2000	2000
Win rate	0%	0%	0%	0%	0%
Score	25.73	22.60	41.2	5.22	34.95
Exploration %	99.93%	99.84%	99.87%	99.98%	99.97%
Unique interactions	6.07	6.02	6.02	5.46	6.29
Curiosity	315.38	332.08	307.97	212.44	186.03
Collisions	39.78	36.19	53.22	12.21	53.83
Hits	276.56	297.60	255.55	200.54	132.57
Interactions	316.34	333.79	308.76	212.75	186.40
Kills	2.81	3.19	3.05	0.60	4.69
Items	27	19	40.99	8	50

Table E.24: *Digdug*: Per-level portability results for E2 (*High collector + low killer*).

E3 (*Low collector + high killer*)					
Features	**lvl 7.2c**	**lvl 7.36a**	**lvl 7.36b**	**lvl 7.36c**	**lvl 7.36d**
Level Info (Max. Values)					
EoG	2000	2000	2000	2000	2000
NPCs (monsters)	12	16	13	5	14
Items	27	19	41	8	50
Gameplay Results					
EoG	2000	2000	2000	2000	2000
Win rate	0%	0%	0%	0%	0%
Score	27.38	30.97	35.33	10.37	32.37
Exploration %	21.49%	21.54%	24.63%	23.83%	29.68%
Unique interactions	6	4.90	6.26	5.65	6.24
Curiosity	114.98	118.7	118.99	88.94	74.37
Collisions	70.01	71.90	74.21	54.27	43.96
Hits	74.67	82	77.94	59.09	47.45
Interactions	144.68	153.89	152.14	113.35	91.40
Kills	9.64	12.62	12.53	4.7	13.31
Items	9.14	5.05	11.36	1.72	9.74

Table E.25: *Digdug*: Per-level portability results for E3 (*Low collector + high killer*).

E4 (*Walls breaker*)					
Features	**lvl 7.2c**	**lvl 7.36a**	**lvl 7.36b**	**lvl 7.36c**	**lvl 7.36d**
Level Info (Max. Values)					
EoG	2000	2000	2000	2000	2000
Breakable walls	267	292	247	196	103
NPCs (monsters)	12	16	13	5	14
Items	27	19	41	8	50
Gameplay Results					
EoG	1916.47	1814.02	1819.31	1947.28	1598.63
Win rate	1%	1%	0%	3%	2%
Score	40.32	41.91	53.95	12.03	42.13
Exploration %	70.12%	67.66%	67.70%	64.73%	46.98%
Unique interactions	6.97	6.88	7	6.98	6.77
Curiosity	464.6	463.77	434.05	358.54	249.87
Collisions	200.09	187.56	193.87	162.54	130.35
Hits	269.65	282.32	245.53	197.70	120.67
Interactions	469.74	469.88	439.4	360.24	251.01
Kills	10.66	13.57	10.97	4.1	10.67
Items	25.68	17.17	37.30	7.7	41.40

Table E.26: *Digdug*: Per-level portability results for E4 (*Walls breaker*).

E5 (*High explorer*)					
Features	**lvl 7.2c**	**lvl 7.36a**	**lvl 7.36b**	**lvl 7.36c**	**lvl 7.36d**
Level Info (Max. Values)					
EoG	2000	2000	2000	2000	2000
Open locations	405	405	405	405	405
Gameplay Results					
EoG	1998.90	1998.53	1998.78	1999.66	1962.19
Win rate	0%	0%	0%	0%	0%
Score	31.54	28.78	47.35	9.12	39.65
Exploration %	99.98%	100%	99.99%	100%	98.51%
Unique interactions	7	7	7	7	6.97
Curiosity	328.28	345.97	321.81	220.82	191.72
Collisions	49.54	45.78	63.42	18.47	58.9
Hits	279.75	301.40	259.16	202.56	133.36
Interactions	329.29	347.17	322.58	221.02	192.26
Kills	5.82	6.40	6.19	2.56	7.43
Items	27	19	41	8	49.02

Table E.27: *Digdug*: Per-level portability results for E5 (*High explorer*).

E6 (*Low explorer + high scorer*)					
Features	**lvl 7.2c**	**lvl 7.36a**	**lvl 7.36b**	**lvl 7.36c**	**lvl 7.36d**
Level Info (Max. Values)					
EoG	2000	2000	2000	2000	2000
Score	44	48	61	14	53
Open locations	405	405	405	405	405
Gameplay Results					
EoG	2000	2000	2000	1999.90	1999.88
Win rate	0%	0%	0%	2%	6.99%
Score	35.49	35.46	51.3	11.02	47.71
Exploration %	30.92%	26.49%	36.32%	26.96%	37.98%
Unique interactions	6.37	5.56	6.35	5.85	6.44
Curiosity	155.65	137.1	165.06	98.46	123
Collisions	94.77	84.11	104.51	60.04	73.60
Hits	95.38	89.59	95.86	63.84	62.54
Interactions	190.14	173.7	200.36	123.88	136.14
Kills	9.64	12.01	11.64	4.36	12.18
Items	21.15	11.92	39	4.51	44.99

Table E.28: *Digdug*: Per-level portability results for E6 (*Low explorer + high scorer*).

Level 7.2c							
Info							
Open locations	405						
Breakable walls	267						
Gameplay Results							
Features	**Max**	**E1**	**E2**	**E3**	**E4**	**E5**	**E6**
EoG	**2000**	1999.61	2000	2000	1916.47	1998.90	2000
Win rate	**-**	6.99%	0%	0%	1%	0%	0%
Score	**44**	41.57	25.73	27.38	40.32	31.54	35.49
Exploration %	**-**	73.19%	99.93%	21.49%	70.12%	99.98%	30.92%
Unique interactions	**-**	7	6.07	6	6.97	7	6.37
Curiosity	**-**	476.84	315.38	114.98	464.6	328.28	155.65
Collisions	**-**	208.37	39.78	70.019	200.09	49.54	94.77
Hits	**-**	275.42	276.56	74.67	269.65	279.75	95.38
Interactions	**-**	483.78	316.34	144.68	469.74	329.29	190.14
Kills	**12**	10.64	2.81	9.64	10.66	5.82	9.64
Items	**27**	26.92	27	9.14	25.68	27	21.15

Table E.29: *Digdug*: Per-agent portability results for Level 7.2c (original).

Level 7.36a							
Info							
Open locations	405						
Breakable walls	292						
Gameplay Results							
Features	**Max**	**E1**	**E2**	**E3**	**E4**	**E5**	**E6**
EoG	**2000**	1999.73	2000	2000	1814.02	1998.53	2000
Win rate	**-**	1%	0%	0%	1%	0%	0%
Score	**48**	44.05	22.60	30.97	41.91	28.78	35.46
Exploration %	**-**	74.00%	99.84%	21.54%	67.66%	100%	26.49%
Unique interactions	**-**	6.94	6.02	4.90	6.88	7	5.56
Curiosity	**-**	494.16	332.08	118.7	463.77	345.97	137.1
Collisions	**-**	206.12	36.19	71.90	187.56	45.78	84.11
Hits	**-**	297.7	297.60	82	282.32	301.40	89.59
Interactions	**-**	503.82	333.79	153.89	469.88	347.17	173.7
Kills	**16**	13.74	3.19	12.62	13.57	6.40	12.01
Items	**19**	18.91	19	5.05	17.17	19	11.92

Table E.30: *Digdug*: Per-agent portability results for Level 7.36a.

Level 7.36b							
Info							
Open locations	405						
Breakable walls	247						
Gameplay Results							
Features	**Max**	**E1**	**E2**	**E3**	**E4**	**E5**	**E6**
EoG	**2000**	1999.77	2000	2000	1819.31	1998.80	2000
Win rate	**-**	2%	0%	0%	0%	0%	0%
Score	**61**	58.2	41.2	35.33	53.95	47.35	51.3
Exploration %	**-**	74.03%	99.87%	24.63%	67.70%	99.99%	36.32%
Unique interactions	**-**	7	6.02	6.26	7	7	6.35
Curiosity	**-**	454.66	307.97	118.99	434.05	321.81	165.06
Collisions	**-**	206.63	53.22	74.21	193.87	63.42	104.51
Hits	**-**	255.12	255.55	77.94	245.53	259.16	95.86
Interactions	**-**	461.75	308.76	152.14	439.4	322.58	200.36
Kills	**13**	11.56	3.05	12.53	10.97	6.19	11.64
Items	**41**	40.76	40.99	11.36	37.30	41	39

Table E.31: *Digdug*: Per-agent portability results for Level 7.36b.

Level 7.36c							
Info							
Open locations	405						
Breakable walls	196						
Gameplay Results							
Features	**Max**	**E1**	**E2**	**E3**	**E4**	**E5**	**E6**
EoG	**2000**	1999.94	2000	2000	1947.28	1999.66	1999.90
Win rate	**-**	2%	0%	0%	3%	0%	2%
Score	**14**	11.83	5.22	10.37	12.03	9.12	11.02
Exploration %	**-**	67.25%	99.98%	23.83%	64.73%	100%	26.96%
Unique interactions	**-**	6.74	5.46	5.65	6.98	7	5.85
Curiosity	**-**	367.8	212.44	88.94	358.54	220.82	98.46
Collisions	**-**	169.57	12.21	54.27	162.54	18.47	60.04
Hits	**-**	200.11	200.54	59.09	197.70	202.56	63.84
Interactions	**-**	369.67	212.75	113.35	360.24	221.02	123.88
Kills	**5**	3.88	0.60	4.7	4.1	2.56	4.36
Items	**8**	7.95	8	1.72	7.7	8	4.51

Table E.32: *Digdug*: Per-agent portability results for Level 7.36c.

Level 7.36d							
Info							
Open locations	405						
Breakable walls	103						
Gameplay Results							
Features	**Max**	**E1**	**E2**	**E3**	**E4**	**E5**	**E6**
EoG	**2000**	1999.57	2000	2000	1598.63	1962.19	1999.88
Win rate	**-**	6%	0%	0%	2%	0%	6.99%
Score	**53**	51.02	34.95	32.37	42.13	39.65	47.71
Exploration %	**-**	57.01%	99.97%	29.68%	46.98%	98.51%	37.98%
Unique interactions	**-**	7	6.29	6.24	6.77	6.97	6.44
Curiosity	**-**	282.05	186.03	74.37	249.87	191.72	123
Collisions	**-**	146.84	53.83	43.96	130.35	58.9	73.60
Hits	**-**	138.69	132.57	47.45	120.67	133.36	62.54
Interactions	**-**	285.53	186.40	91.40	251.01	192.26	136.14
Kills	**14**	12.68	4.69	13.31	10.67	7.43	12.18
Items	**50**	49.90	50	9.74	41.40	49.02	44.99

Table E.33: *Digdug*: Per-agent portability results for Level 7.36d.

E.4 Portability results in *Sheriff*

Two types of tables are included. First, each table contains the final stats of a particular agent across all the levels used for *Sheriff*. Last, each table gathers the stats by level, displaying the results of the different *behaviour-type* agents in it.

In the tables, the stats related to the competence of the corresponding *behaviour-type* are highlighted. Also, information about relevant maximum values for each of the levels is included for reference.

E1 (*Survivor + low killer*)					
Features	**lvl 7.2d**	**lvl 7.43a**	**lvl 7.43b**	**lvl 7.43c**	**lvl 7.43d**
Level Info (Max. Values)					
EoG	1000	1000	1000	1000	1000
NPCs (bandits)	8	9	9	11	11
Gameplay Results					
EoG	992.58	990.32	1000	991.82	1000
Win rate	98%	99%	100%	99%	100%
Score	3.27	4.03	3.91	4.77	4.87
Exploration %	99.11%	98.55%	99.67%	98.87%	99.41%
Unique interactions	5.26	5.44	5.2	5.84	5.67
Curiosity	39.73	45.74	23.82	40.94	31.69
Collisions	11.64	11.71	10.02	12.29	11.11
Hits	30.07	36.06	15.79	30.94	22.87
Interactions	41.71	47.76	25.81	43.23	33.97
Kills	3.28	4.02	3.89	4.78	4.87

Table E.34: *Sheriff*: Per-level portability results for E1 (*Survivor + low killer*).

E2 (*High killer + high explorer*)					
Features	**lvl 7.2d**	**lvl 7.43a**	**lvl 7.43b**	**lvl 7.43c**	**lvl 7.43d**
Level Info (Max. Values)					
EoG	1000	1000	1000	1000	1000
NPCs (bandits)	8	9	9	11	11
Open locations	280	280	280	280	280
Gameplay Results					
EoG	952.62	977.38	969.09	972.01	980.83
Win rate	97%	99%	98%	97%	100%
Score	7.07	8.09	8.03	9.93	10.12
Exploration %	97.89%	98.97%	99.01%	98.2%	99.55%
Unique interactions	6.02	5.85	5.64	6.42	6.17
Curiosity	53.3	63.69	32.58	62.36	47.94
Collisions	8.63	9.60	6.88	10.22	9.31
Hits	45.98	55.85	27.16	53.89	40.16
Interactions	54.61	65.44	34.04	64.11	49.47
Kills	6.96	7.95	7.96	9.91	10

Table E.35: *Sheriff*: Per-level portability results for E2 (*High killer + high explorer*).

E3 (*High killer + low explorer*)					
Features	**lvl 7.2d**	**lvl 7.43a**	**lvl 7.43b**	**lvl 7.43c**	**lvl 7.43d**
Level Info (Max. Values)					
EoG	1000	1000	1000	1000	1000
NPCs (bandits)	8	9	9	11	11
Open locations	280	280	280	280	280
Gameplay Results					
EoG	449.45	503.97	430.43	556.24	479.03
Win rate	98%	98%	100%	99%	99%
Score	7.92	8.88	8.99	10.89	10.98
Exploration %	15.33%	16.36%	15.26%	17.81%	15.96%
Unique interactions	5.09	5.25	4.41	5.40	5.4
Curiosity	24.11	28.27	14.52	32.22	25.07
Collisions	9.18	11.72	6.84	12.27	9.95
Hits	20.90	24.34	13.06	28.72	22.48
Interactions	30.07	36.06	19.9	40.98	32.43
Kills	6.95	7.93	8	9.95	10

Table E.36: *Sheriff*: Per-level portability results for E3 (*High killer + low explorer*).

E4 (*Speed-runner*)					
Features	**lvl 7.2d**	**lvl 7.43a**	**lvl 7.43b**	**lvl 7.43c**	**lvl 7.43d**
Level Info (Max. Values)					
EoG	1000	1000	1000	1000	1000
Gameplay Results					
EoG	364.75	360.53	363.59	417.71	418.7
Win rate	100%	97%	98%	99%	100%
Score	7.99	8.78	8.89	10.88	10.99
Exploration %	65.90%	62.46%	63.58%	70.28%	68.63%
Unique interactions	5.91	5.93	5.98	6.29	6.16
Curiosity	44.06	46.51	29.85	53.43	42.42
Collisions	11.11	9.81	11.46	13.28	12.26
Hits	34.81	38.9	21.14	42.87	32.53
Interactions	45.92	48.71	32.6	56.15	44.79
Kills	7	7.84	7.93	9.91	10

Table E.37: *Sheriff*: Per-level portability results for E4 (*Speed-runner*).

E5 (*Barrels shooter*)					
Features	**lvl 7.2d**	**lvl 7.43a**	**lvl 7.43b**	**lvl 7.43c**	**lvl 7.43d**
Level Info (Max. Values)					
EoG	1000	1000	1000	1000	1000
Barrels	55	65	27	71	51
NPCs (bandits)	8	9	9	11	11
Gameplay Results					
EoG	668.73	571.65	534.52	597.22	498.91
Win rate	44%	33%	31%	32%	26%
Score	5.34	5.29	5.59	6.84	6
Exploration %	17.10%	15.22%	14.10%	16.14%	14.66%
Unique interactions	5.49	5.30	5.06	5.53	5.5
Curiosity	73.17	70.85	45.54	72.52	57.08
Collisions	403.46	334.44	327.25	345.68	281.61
Hits	35.96	38.19	17.49	39.86	29.28
Interactions	439.41	372.63	344.74	385.54	310.88
Kills	5.83	5.89	6.22	7.46	6.65

Table E.38: *Sheriff*: Per-level portability results for E5 (*Barrels shooter*).

E6 (*High curiosity + low interactions*)					
Features	**lvl 7.2d**	**lvl 7.43a**	**lvl 7.43b**	**lvl 7.43c**	**lvl 7.43d**
Level Info (Max. Values)					
EoG	1000	1000	1000	1000	1000
Gameplay Results					
EoG	837.65	817.12	807.13	813.78	864.55
Win rate	94%	94%	92%	92%	97%
Score	7.07	8.04	8.24	9.79	10.12
Exploration %	91.44%	91.06%	91.81%	89.63%	93.46%
Unique interactions	6.11	6.01	6.17	6.03	6.25
Curiosity	95.77	101.81	73.46	99.92	89.47
Collisions	75.38	73.13	79.42	72.89	76.31
Hits	47.14	54.33	26.86	53.04	40.67
Interactions	122.51	127.46	106.28	125.93	116.98
Kills	6.8	7.71	7.89	9.49	9.81

Table E.39: *Sheriff*: Per-level portability results for E6 (*High curiosity + low interactions*).

Level 7.2d							
Info							
Barrels	55						
Open locations	280						
Walls	88						
Gameplay Results							
Features	**Max**	**E1**	**E2**	**E3**	**E4**	**E5**	**E6**
EoG	**1000**	992.58	952.62	449.45	364.75	668.73	837.65
Win rate	**-**	98%	97%	98%	100%	44%	94%
Score	**8**	3.27	7.07	7.92	7.99	5.34	7.07
Exploration %	**-**	99.11%	97.89%	15.33%	65.90%	17.10%	91.44%
Unique interactions	**-**	5.26	6.02	5.09	5.91	5.49	6.11
Curiosity	**-**	39.73	53.3	24.11	44.06	73.17	95.77
Collisions	**-**	11.64	8.63	9.18	11.11	403.46	75.38
Hits	**-**	30.07	45.98	20.90	34.81	35.96	47.14
Interactions	**-**	41.71	54.61	30.07	45.92	439.41	122.51
Kills	**8**	3.28	6.96	6.95	7	5.83	6.8

Table E.40: *Sheriff*: Per-agent portability results for Level 7.2d.

Level 7.43a							
Info							
Barrels	65						
Open locations	280						
Walls	88						
Gameplay Results							
Features	**Max**	**E1**	**E2**	**E3**	**E4**	**E5**	**E6**
EoG	**1000**	990.32	977.38	503.97	360.53	571.65	817.12
Win rate	**-**	99%	99%	98%	97%	33%	94%
Score	**9**	4.03	8.09	8.88	8.78	5.29	8.04
Exploration %	**-**	98.55%	98.97%	16.36%	62.46%	15.22%	91.06%
Unique interactions	**-**	5.44	5.85	5.25	5.93	5.30	6.01
Curiosity	**-**	45.74	63.69	28.27	46.51	70.85	101.81
Collisions	**-**	11.71	9.60	11.72	9.81	334.44	73.13
Hits	**-**	36.06	55.85	24.34	38.9	38.19	54.33
Interactions	**-**	47.76	65.44	36.06	48.71	372.63	127.46
Kills	**9**	4.02	7.95	7.93	7.84	5.89	7.71

Table E.41: *Sheriff*: Per-agent portability results for Level 7.43a.

Level 7.43b							
Info							
Barrels	27						
Open locations	280						
Walls	88						
Gameplay Results							
Features	**Max**	**E1**	**E2**	**E3**	**E4**	**E5**	**E6**
EoG	**1000**	1000	969.09	430.43	363.59	534.52	807.13
Win rate	**-**	100%	98%	100%	98%	31%	92%
Score	**9**	3.91	8.03	8.99	8.89	5.59	8.24
Exploration %	**-**	99.67%	99.01%	15.26%	63.58%	14.10%	91.81%
Unique interactions	**-**	5.2	5.64	4.47	5.98	5.06	6.17
Curiosity	**-**	23.82	32.58	14.52	29.85	45.54	73.46
Collisions	**-**	10.02	6.88	6.84	11.46	327.25	79.42
Hits	**-**	15.79	27.16	13.06	21.14	17.49	26.86
Interactions	**-**	25.81	34.04	19.9	32.6	344.74	106.28
Kills	**9**	3.89	7.96	8	7.93	6.22	7.89

Table E.42: *Sheriff*: Per-agent portability results for Level 7.43b.

Level 7.43c							
Info							
Barrels	71						
Open locations	280						
Walls	88						
Gameplay Results							
Features	**Max**	**E1**	**E2**	**E3**	**E4**	**E5**	**E6**
EoG	**1000**	991.82	972.01	556.24	417.71	597.22	813.78
Win rate	**-**	99%	97%	99%	99%	32%	92%
Score	**11**	4.77	9.93	10.89	10.88	6.84	9.79
Exploration %	**-**	98.87%	98.2%	17.81%	70.28%	16.14%	89.63%
Unique interactions	**-**	5.84	6.42	5.40	6.29	5.53	6.03
Curiosity	**-**	40.94	62.36	32.22	53.43	72.52	99.92
Collisions	**-**	12.29	10.22	12.27	13.28	345.68	72.89
Hits	**-**	30.94	53.89	28.72	42.87	39.86	53.04
Interactions	**-**	43.23	64.11	40.98	56.15	385.54	125.93
Kills	**11**	4.78	9.91	9.95	9.91	7.46	9.49

Table E.43: *Sheriff*: Per-agent portability results for Level 7.43c.

Level 7.43d							
Info							
Barrels	51						
Open locations	280						
Walls	88						
Gameplay Results							
Features	**Max**	**E1**	**E2**	**E3**	**E4**	**E5**	**E6**
EoG	**1000**	1000	980.83	479.03	418.7	498.91	864.55
Win rate	**-**	100%	100%	99%	100%	26%	97%
Score	**11**	4.87	10.12	10.98	10.99	6	10.12
Exploration %	**-**	99.41%	99.55%	15.96%	68.63%	14.66%	93.46%
Unique interactions	**-**	5.70	6.17	5.4	6.16	5.5	6.25
Curiosity	**-**	31.69	47.94	25.07	42.42	57.08	89.47
Collisions	**-**	11.11	9.31	9.95	12.26	281.61	76.31
Hits	**-**	22.87	40.16	22.48	32.53	29.28	40.67
Interactions	**-**	33.97	49.47	32.43	44.79	310.88	116.98
Kills	**11**	4.87	10	10	10	6.65	9.81

Table E.44: *Sheriff*: Per-agent portability results for Level 7.43d.

Appendix F

Case Study: Material

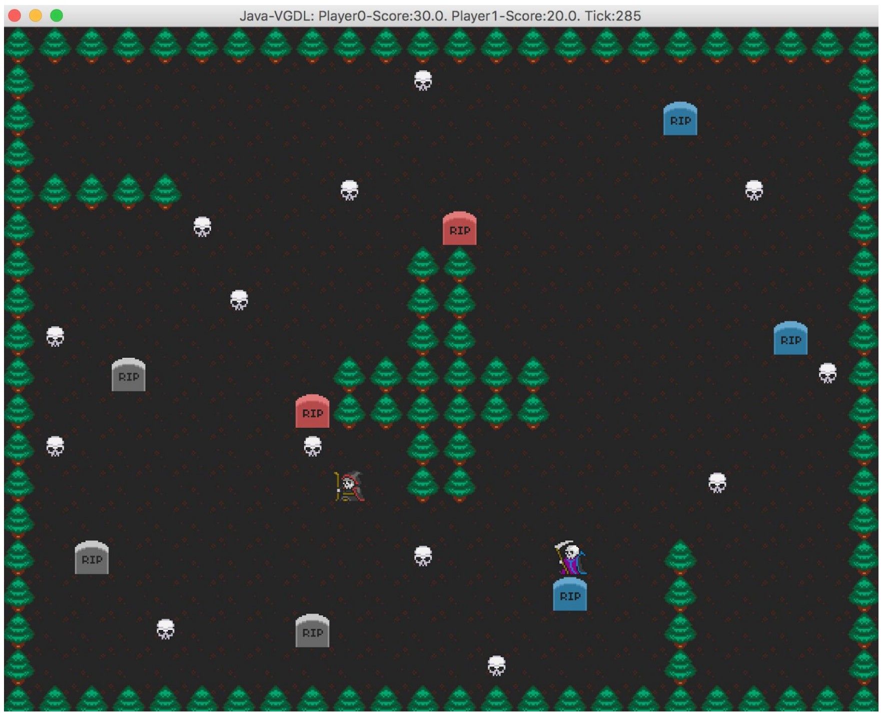
For the Case Study described in Chapter 8 I include the Information Sheet and Consent Form provided to the participants; as well as the list of questions used, taken from the IMI and PENS questionnaires. PENS is not open source, so I merely mention the number of the question used, while in IMI I include the questions used and their adaptation to our questionnaire.

F.1 Information Sheet

Information Sheet

Overview
In this study, you will be asked to play a game and then to fill 2 questionnaires about your experience of playing the game. The game is a basic player versus agent capture game. You are not being assessed on your performance in the game but do try to play as best you can!
You will be playing the game 3 times and each session will be 30 seconds long. The whole experiment should not take more than 15 minutes

The game
Skulls and Tombstones is a competitive game. You collect skulls and bring them to the tombstones, one by one.
Your avatar is on the right and he turns the tombstone BLUE when you bring a skull to a tombstone, indicating that you have captured it.
Your opponent turns the tombstones RED, but you can turn them back to BLUE by bringing a skull to it.
You win if the number of blue tombstones is higher than the red tombstones when the game ends.



Player controllers
The game is played with the arrows of the keyboard. Only four actions are allowed: ↑, ↓, ← and →.

Questionnaire
After playing, you will be asked to fill out a questionnaire about your playing experience. You are not required to answer all the questions but please do so for the study to be robust.

Questions
If you have any questions about the game or the study please ask them now but once the study has started, please keep your questions until the the end.

Withdrawing
You are free to withdraw from this study at any point without giving a reason□. There are no incentives for doing this activity so your participation should be voluntary.

Data
Your ingame score data and the data that you are filling in the questionnaire is being gathered. The data gathered will be visible to the researchers conducting the study. This will be in the form of a spreadsheet but you will not be individually identifiable from the data. The data may be used later for further study or publication. However, you will not be identifiable in any way for either reason.

F.2 Consent Form

Consent to Participate in a Research Study

Goldsmiths, University of London
10th July 2017 - Onwards

Title of study: Skulls and tombstones experiment
Investigators: Sokol Murturi (smurt001@gold.ac.uk), Andrew Martin (andrew.martin@gold.ac.uk), Cristina Guerrero-Romero (cris.guerrero@essex.ac.uk), Shringi Kumari (sk1382@york.ac.uk)

Please read carefully, tick all the boxes and sign the form to give your consent to participate in the study.

- I confirm that I am at least 18 years old. ☐
- I confirm that I have read and understood the information sheet provided. ☐
- I confirm that I have had the opportunity to consider the information, ask questions and have had any queries answered satisfactorily. ☐
- I understand that my participation is voluntary and that I am free to withdraw at any time without giving any reason and without there being any negative consequences. ☐
- I agree for the data collected from me to be anonymously used by the investigators in the ongoing experiment. ☐
- I give my permission to participate in this study. ☐

Name of participant	Date	Signature
Name of researcher	Date	Signature

1

ID	Component	Item	Questionnaire item	Original item
1	Competence	PENS1		
2		PENS2		
3		PENS3		
4	Interest/Enjoyment	IMI1	I enjoyed playing this game very much	I enjoyed doing this activity very much
5		IMI2	This game was fun to play	This activity was fun to do.
6		IMI3	I thought this was a boring game	I thought this was a boring activity. (R)
7		IMI4	This game did not hold my attention at all	This activity did not hold my attention at all. (R)
8		IMI5	I would describe this game as very interesting	I would describe this activity as very interesting.
9		IMI6	I thought this game was quite enjoyable	I thought this activity was quite enjoyable.
10		IMI7	While I was playing the game, I was thinking about how much I enjoyed it	While I was doing this activity, I was thinking about how much I enjoyed it.
11	Pressure/Tension	IMI19	I did not feel nervous at all while playing this game	I did not feel nervous at all while doing this. (R)
12		IMI20	I felt very tense while playing this game	I felt very tense while doing this activity.
13		IMI21	I was very relaxed while playing this	I was very relaxed in doing these. (R)
14		IMI22	I was anxious while playing this game	I was anxious while working on this task.
15		IMI23	I felt pressured while playing this game	I felt pressured while doing these.
16	Perceived Competence	IMI8	I think I am pretty good at this game	I think I am pretty good at this activity.
17		IMI9	I think I did pretty well at this game, compared to others players	I think I did pretty well at this activity, compared to other students.
18		IMI10	After playing this game for a while, I felt pretty competent	After working at this activity for awhile, I felt pretty competent.
19		IMI11	I am satisfied with my performance in the game	I am satisfied with my performance at this task.
20		IMI12	I was pretty skilled at this game	I was pretty skilled at this activity.
21		IMI13	This was a game I could not play very well	This was an activity that I couldn't do very well.

Table F.1: The PENS and IMI items used in the questionnaire for the Case Study. PENS is not open source so questions are referenced but not included. For IMI, it includes the original and paraphrased questions used in the study.

Bibliography

Activision. *Call of Duty* (series). Activision, Oct 2003.

Aghyad Mohammad Albaghajati and Moataz Aly Kamaleldin Ahmed. Video Game Automated Testing Approaches: An Assessment Framework. *IEEE Transactions on Games*, 2020.

Alberto Alvarez, Steve Dahlskog, Jose Font, and Julian Togelius. Empowering Quality Diversity in Dungeon Design with Interactive Constrained MAP-Elites. In *2019 IEEE Conference on Games (CoG)*, pages 1–8. IEEE, 2019.

Damien Anderson, Cristina Guerrero-Romero, Philip Rodgers, John Levine, and Diego Perez-Liebana. Ensemble Decision Systems for General Video Game Playing. In *2019 IEEE Conference on Games (CoG)*, pages 1–8. IEEE, 2019.

Marc André. *Splendor*. Space Cowboys, 2014.

Sinan Ariyurek, Aysu Betin-Can, and Elif Surer. Automated Video Game Testing Using Synthetic and Human-Like Agents. *IEEE Transactions on Games*, 2019.

Sinan Ariyurek, Elif Surer, and Aysu Betin-Can. Playtesting: What is Beyond Personas. *arXiv preprint arXiv:2107.11965*, 2021.

Martin Balla, Adrián Barahona-Rıos, Adam Katona, et al. Illuminating Game Space Using MAP-Elites for Assisting Video Game Design. In *11th AISB Symposium on AI & Games (AI&G)*, pages 1–6, 2021.

Nolan Bard, Jakob N Foerster, Sarath Chandar, Neil Burch, Marc Lanctot, H Francis Song, Emilio Parisotto, Vincent Dumoulin, Subhodeep Moitra, Edward Hughes, et al. The Hanabi Challenge: A New Frontier for AI Research. *Artificial Intelligence*, 280: 103216, 2020.

Richard Bartle. Hearts, Clubs, Diamonds, Spades: Players who Suit MUDs. *Journal of MUD research*, 1(1):19, 1996.

Richard Bartle. Virtual Worlds: Why People Play. *Massively multiplayer game development*, 2(1):3–18, 2005.

Richard Bartle. Player Type Theory: Uses and Abuses, Feb 2012. URL `https://www.youtube.com/watch?v=ZIzLbE-93nc`. Casual Connect Europe.

Antoine Bauza. *Hanabi*. R&R Games, Cocktail Games, Abacus Spiele, 2014.

Marc Bellemare, Sriram Srinivasan, Georg Ostrovski, Tom Schaul, David Saxton, and Remi Munos. Unifying Count-Based Exploration and Intrinsic Motivation. *Advances in neural information processing systems*, 29:1471–1479, 2016.

Marc G Bellemare, Yavar Naddaf, Joel Veness, and Michael Bowling. The Arcade Learning Environment: An Evaluation Platform for General Agents. *Journal of Artificial Intelligence Research*, 47:253–279, 2013.

Hans J Berliner. Experiences in Evaluation with BKG-A Program that Plays Backgammon. In *IJCAI*, volume 5, pages 428–433. Citeseer, 1977.

Regina Bernhaupt. User Experience Evaluation Methods in the Games Development Life Cycle. In *Game User Experience Evaluation*, pages 1–8. Springer, 2015.

Darse Billings, Denis Papp, Jonathan Schaeffer, and Duane Szafron. Poker as a Testbed for AI Research. In *Proceedings of AI'98, Conference of the Canadian Society for Computational Studies of Intelligence*, pages 228–238. Springer, 1998.

BioWare. *Mass Effect*. Microsoft Game Studios, Nov 2007.

Blizzard Entertainment. *Starcraft*. Blizzard Entertainment, Mar 1998.

Blizzard Entertainment. *Hearthstone*. Blizzard Entertainment, Mar 2014.

Ivan Bravi and Simon Lucas. Rinascimento: Searching the Behaviour Space of Splendor. *arXiv preprint arXiv:2106.08371*, 2021.

Greg Brockman, Vicki Cheung, Ludwig Pettersson, Jonas Schneider, John Schulman, Jie Tang, and Wojciech Zaremba. OpenAI Gym. *arXiv preprint arXiv:1606.01540*, 2016.

Noam Brown and Tuomas Sandholm. Superhuman AI for Multiplayer Poker. *Science*, 365(6456):885–890, 2019.

Cameron Browne and Frederic Maire. Evolutionary Game Design. *IEEE Transactions on Computational Intelligence and AI in Games*, 2(1):1–16, 2010.

Cameron B Browne, Edward Powley, Daniel Whitehouse, Simon M Lucas, Peter I Cowling, Philipp Rohlfshagen, Stephen Tavener, Diego Perez, Spyridon Samothrakis, and Simon Colton. A Survey of Monte Carlo Tree Search Methods. *IEEE Transactions on Computational Intelligence and AI in games*, 4(1):1–43, 2012.

Michael Buro. The Othello Match of the Year: Takeshi Murakami vs. Logistello. *ICGA Journal*, 20(3):189–193, 1997.

Murray Campbell, A Joseph Hoane Jr, and Feng-hsiung Hsu. Deep Blue. *Artificial Intelligence*, 134(1-2):57–83, 2002.

Rodrigo Canaan, Julian Togelius, Andy Nealen, and Stefan Menzel. Diverse Agents for Ad-Hoc Cooperation in Hanabi. In *2019 IEEE Conference on Games (CoG)*, pages 1–8. IEEE, 2019.

Alessandro Canossa and Anders Drachen. Play-personas: Behaviours and Belief Systems in User-Centred Game Design. In *IFIP Conference on Human-Computer Interaction*, pages 510–523. Springer, 2009.

Loïc Caroux, Katherine Isbister, Ludovic Le Bigot, and Nicolas Vibert. Player-Video Game Interaction: A Systematic Review of Current Concepts. *Computers in Human Behavior*, 48:366–381, 2015.

Tristan Cazenave, Yen-Chi Chen, Guan-Wei Chen, Shi-Yu Chen, Xian-Dong Chiu, Julien Dehos, Maria Elsa, Qucheng Gong, Hengyuan Hu, Vasil Khalidov, et al. Polygames: Improved Zero Learning. *ICGA Journal*, 42(4):244–256, 2020.

Jenova Chen. Designing Journey, 2013. URL `https://www.gdcvault.com/play/1017700/Designing`. GDC.

Chun Yin Chu, Tomohiro Harada, and Ruck Thawonmas. Biasing Monte-Carlo Rollouts with Potential Field in General Video Game Playing. In *IPSJ Kansai-Branch Convention*, pages 1–6, 2015a.

Chun Yin Chu, Hisaaki Hashizume, Zikun Guo, Tomohiro Harada, and Ruck Thawonmas. Combining Pathfinding Algorithm with Knowledge-based Monte-Carlo Tree Search in General Video Game Playing. In *2015 IEEE Conference on Computational Intelligence and Games (CIG)*, pages 523–529. IEEE, 2015b.

Alan Cooper. *The Inmates are Running the Asylum*. Sams, Apr 1999.

Creative Assembly. *Total War: Rome II*. SEGA, Sep 2013.

Crystal Dynamics. *Tomb Raider: Underworld*. Eidos Interactive, Nov 2008.

Daedalic Entertainment. *Deponia*. Daedalic Entertainment, Jan 2012.

Pierre Le Pelletier de Woillemont, Rémi Labory, and Vincent Corruble. Configurable Agent With Reward As Input: A Play-Style Continuum Generation. In *2021 IEEE Conference on Games (CoG)*. IEEE, 2021.

Edward L Deci and Richard M Ryan. Intrinsic Motivation Inventory. *Self-Determination Theory*, 267, 2003.

Anders Drachen, Alessandro Canossa, and Georgios N Yannakakis. Player Modeling Using Self-Organization in Tomb Raider: Underworld. In *Computational Intelligence and Games (CIG). IEEE Symposium on*, pages 1–8. IEEE, 2009.

Anders Drachen, Pejman Mirza-Babaei, and Lennart E Nacke. *Games User Research*. Oxford University Press, 2018.

Marc Ebner, John Levine, Simon M. Lucas, Tom Schaul, Tommy Thompson, and Julian Togelius. Towards a Video Game Description Language. In *Artificial and Computational Intelligence in Games*, volume 6 of *Dagstuhl Follow-Ups*, pages 85–100. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, Dagstuhl, Germany, 2013. ISBN 978-3-939897-62-0. doi: 10.4230/DFU.Vol6.12191.85. URL `http://drops.dagstuhl.de/opus/volltexte/2013/4338`.

Agoston E Eiben, James E Smith, et al. *Introduction to Evolutionary Computing*, volume 53. Springer, 2003.

Mark Ferguson, Sam Devlin, Daniel Kudenko, and James Alfred Walker. Player Style Clustering without Game Variables. In *International Conference on the Foundations of Digital Games*, pages 1–4. ACM, 2020.

Matthew C Fontaine, Scott Lee, Lisa B Soros, Fernando de Mesentier Silva, Julian Togelius, and Amy K Hoover. Mapping Hearthstone Deck Spaces through Map-Elites with Sliding Boundaries. In *Proceedings of The Genetic and Evolutionary Computation Conference*, pages 161–169, 2019.

Raluca D Gaina, Diego Pérez-Liébana, and Simon M Lucas. General Video Game for 2 Players: Framework and Competition. In *2016 8th Computer Science and Electronic Engineering (CEEC)*, pages 186–191. IEEE, 2016.

Raluca D Gaina, Jialin Liu, Simon M Lucas, and Diego Pérez-Liébana. Analysis of Vanilla Rolling Horizon Evolution Parameters in General Video Game Playing. In *European Conference on the Applications of Evolutionary Computation*, pages 418–434. Springer, 2017.

Pablo García-Sánchez, Alberto Tonda, Antonio M Mora, Giovanni Squillero, and Juan Julián Merelo. Automated Playtesting in Collectible Card Games Using Evolutionary Algorithms: A Case Study in Hearthstone. *Knowledge-Based Systems*, 153: 133–146, 2018.

Michael Genesereth and Yngvi Björnsson. The International General Game Playing Competition. *AI Magazine*, 34(2):107–107, 2013.

Michael Genesereth, Nathaniel Love, and Barney Pell. General Game Playing: Overview of the AAAI Competition. *AI magazine*, 26(2):62–62, 2005.

Christian Guckelsberger, Christoph Salge, Jeremy Gow, and Paul Cairns. Predicting Player Experience without the Player. an Exploratory Study. In *Proceedings of the Annual Symposium on Computer-Human Interaction in Play*, pages 305–315, 2017.

Stefan Freyr Gudmundsson, Philipp Eisen, Erik Poromaa, Alex Nodet, Sami Purmonen, Bartlomiej Kozakowski, Richard Meurling, and Lele Cao. Human-Like Playtesting with Deep Learning. In *2018 IEEE Conference on Computational Intelligence and Games (CIG)*, pages 1–8. IEEE, 2018.

Cristina Guerrero Romero. GVGAI experiments. Github, 2017. URL `https://github.com/kisenshi/gvgai-experiments`.

Cristina Guerrero Romero. Demo - Diversifying Heuristics for GVGAI. Youtube, 2018. URL `https://www.youtube.com/watch?v=aLgPm9kbfY8`.

Cristina Guerrero Romero. Beyond Playing to Win: Agent Behaviour Research (in GVGAI). Github, 2021a. URL `https://github.com/kisenshi/gvgai-agent-behaviour-research`.

Cristina Guerrero Romero. Beyond Playing to Win: Agent Behaviour Research. Results Processing. Github, 2021b. URL `https://github.com/kisenshi/experiments-automated-gameplay-results-processing`.

Cristina Guerrero Romero. Configuration and Results: Generalisation Experiments for the Generated Team of Agents with Diverse Behaviours, Oct 2021c. URL `https://osf.io/kmgz4`.

Cristina Guerrero Romero. Experiments and Results: MAP-Elites to Generate a Team of Agents that Elicits Diverse Automated Gameplay. OSF, Apr 2021d. URL `https://osf.io/whxm8`.

Cristina Guerrero Romero. Visualize Diverse Gameplays Based on Agent Behaviour, 2021e. URL `https://demo-visualize-diverse-gameplay-xqjmp.ondigitalocean.app/`.

Cristina Guerrero Romero and Shringi Kumari. Material of Studying General Agents in Video Games from the Perspective of Player Experience. OSF, Aug 2020. URL `https://osf.io/tmc6x`.

Cristina Guerrero-Romero and Diego Perez-Liebana. MAP-Elites to Generate a Team of Agents that Elicits Diverse Automated Gameplay. In *2021 IEEE Conference on Games (CoG)*. IEEE, 2021.

Cristina Guerrero-Romero, Annie Louis, and Diego Perez-Liebana. Beyond Playing to Win: Diversifying Heuristics for GVGAI. In *2017 IEEE Conference on Computational Intelligence and Games (CIG)*, pages 118–125. IEEE, 2017.

Cristina Guerrero-Romero, Simon M Lucas, and Diego Perez-Liebana. Using a Team of General AI Algorithms to Assist Game Design and Testing. In *2018 IEEE Conference on Computational Intelligence and Games (CIG)*, pages 1–8. IEEE, 2018.

Cristina Guerrero-Romero, Shringi Kumari, Diego Perez-Liebana, and Sebastian Deterding. Studying General Agents in Video Games from the Perspective of Player Experience. *Proceedings of the AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment*, 16(1):217–223, Oct. 2020.

Christoffer Holmgård, Antonios Liapis, Julian Togelius, and Georgios N Yannakakis. Evolving Personas for Player Decision Modeling. In *2014 IEEE Conference on Computational Intelligence and Games*, pages 1–8. IEEE, 2014a.

Christoffer Holmgård, Antonios Liapis, Julian Togelius, and Georgios N Yannakakis. Generative Agents for player Decision Modeling in Games. In *9th International Conference on the Foundations of Digital Games*, 2014b.

Christoffer Holmgård, Julian Togelius, Antonios Liapis, and Georgios N Yannakakis. MiniDungeons 2: An Experimental Game for Capturing and Modeling Player Decisions. In *10th Conference on the Foundations of Digital Games*, 2015.

Christoffer Holmgård, Michael Cerny Green, Antonios Liapis, and Julian Togelius. Automated Playtesting with Procedural Personas through MCTS with Evolved Heuristics. *IEEE Transactions on Games*, 11(4):352–362, 2018.

Andrew Hoyt, Matthew Guzdial, Yalini Kumar, Gillian Smith, and Mark O Riedl. Integrating Automated Play in Level Co-Creation. *arXiv preprint arXiv:1911.09219*, 2019.

id Software. *Doom*. id Software, 1993.

Sidra Iftikhar, Muhammad Zohaib Iqbal, Muhammad Uzair Khan, and Wardah Mahmood. An Automated Model Based Testing Approach for Platform Games. In *2015 ACM/IEEE 18th International Conference on Model Driven Engineering Languages and Systems (MODELS)*, pages 426–435. IEEE, 2015.

IO Interactive. *Hitman: Blood Money*. Eidos Interactive, May 2006.

Matthew Johnson, Katja Hofmann, Tim Hutton, and David Bignell. The Malmo Platform for Artificial Intelligence Experimentation. In *IJCAI*, pages 4246–4247. Citeseer, 2016.

Arthur Juliani, Vincent-Pierre Berges, Ervin Teng, Andrew Cohen, Jonathan Harper, Chris Elion, Chris Goy, Yuan Gao, Hunter Henry, Marwan Mattar, et al. Unity: A General Platform for Intelligent Agents. *arXiv preprint arXiv:1809.02627*, 2018.

Arthur Juliani, Ahmed Khalifa, Vincent-Pierre Berges, Jonathan Harper, Ervin Teng, Hunter Henry, Adam Crespi, Julian Togelius, and Danny Lange. Obstacle Tower: A Generalization Challenge in Vision, Control, and Planning. *arXiv preprint arXiv:1902.01378*, 2019.

Ahmed Khalifa, Diego Perez-Liebana, Simon M Lucas, and Julian Togelius. General Video Game Level Generation. In *Proceedings of the Genetic and Evolutionary Computation Conference 2016*, pages 253–259. ACM, 2016.

Ahmed Khalifa, Michael Cerny Green, Diego Perez-Liebana, and Julian Togelius. General Video Game Rule Generation. In *2017 IEEE Conference on Computational Intelligence and Games (CIG)*, pages 170–177. IEEE, 2017.

Ahmed Khalifa, Scott Lee, Andy Nealen, and Julian Togelius. Talakat: Bullet Hell Generation through Constrained MAP-Elites. In *Proceedings of The Genetic and Evolutionary Computation Conference*, pages 1047–1054, 2018.

Krams Design. *Anna's Quest*. Daedalic Entertainment, Jul 2015.

Marc Lanctot, Edward Lockhart, Jean-Baptiste Lespiau, Vinicius Zambaldi, Satyaki Upadhyay, Julien Pérolat, Sriram Srinivasan, Finbarr Timbers, Karl Tuyls, Shayegan Omidshafiei, et al. OpenSpiel: A Framework for Reinforcement Learning in Games. *arXiv preprint arXiv:1908.09453*, 2019.

John Levine, Clare Bates Congdon, Marc Ebner, et al. General Video Game Playing. In *Artificial and Computational Intelligence in Games*, Dagstuhl Follow-Ups, page 8. Dagstuhl Publishing, Nov 2013.

Daniele Loiacono, Pier Luca Lanzi, Julian Togelius, Enrique Onieva, David A Pelta, Martin V Butz, Thies D Lönneker, Luigi Cardamone, Diego Perez, Yago Sáez, et al. The 2009 simulated car racing championship. *IEEE Transactions on Computational Intelligence and AI in Games*, 2(2):131–147, 2010.

Simon M Lucas. Ms Pac-Man Competition. *ACM SIGEVOlution*, 2(4):37–38, 2007.

Simon M Lucas, Jialin Liu, and Diego Perez-Liebana. The N-Tuple Bandit Evolutionary Algorithm for Game Agent Optimisation. In *2018 IEEE Congress on Evolutionary Computation (CEC)*, pages 1–9. IEEE, 2018.

Marlos C Machado, Marc G Bellemare, Erik Talvitie, Joel Veness, Matthew Hausknecht, and Michael Bowling. Revisiting the Arcade Learning Environment: Evaluation Protocols and Open Problems for General Agents. *Journal of Artificial Intelligence Research*, 61:523–562, 2018.

T Machado, A Nealen, and J Togelius. CICERO: Computationally Intelligent Collaborative EnviROnment for game and level design. In *3rd workshop on Computational Creativity and Games (CCGW) at the 8th International Conference on Computational Creativity (ICCC'17)*, pages 1–8, 2017a.

Tiago Machado. SeekWhence, Nov 2018. URL `https://www.youtube.com/watch?v=MhjBv2ZPpek`.

Tiago Machado, Andy Nealen, and Julian Togelius. Seekwhence a Retrospective Analysis Tool for General Game Design. In *Proceedings of the 12th International Conference on the Foundations of Digital Games*, pages 1–6, 2017b.

Cyril Marlin. Automated Testing: Building A Flexible Game Solver, Oct 2011. URL `https://www.gamedeveloper.com/programming/automated-testing-building-a-flexible-game-solver`. Game Developer.

Robert Masella. Automated Testing of Gameplay Features in 'Sea of Thieves', 2019. URL `https://www.youtube.com/watch?v=X673tOi8pU8`. GDC.

Massive Entertainment. *Tom Clancy's The Division*. Ubisoft, Mar 2016.

John McCarthy. What is Artificial Intelligence? *Computer Science Department, Stanford University*, 1998. Available online at `http://jmc.stanford.edu/articles/whatisai/whatisai.pdf`.

Elisa D Mekler, Julia Ayumi Bopp, Alexandre N Tuch, and Klaus Opwis. A Systematic Review of Quantitative Studies on the Enjoyment of Digital Entertainment Games. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pages 927–936, 2014.

David Melhart, Ahmad Azadvar, Alessandro Canossa, Antonios Liapis, and Georgios N Yannakakis. Your Gameplay Says it All: Modelling Motivation in Tom Clancy's the Division. In *2019 IEEE Conference on Games (CoG)*, pages 1–8. IEEE, 2019.

Andre Mendes, Julian Togelius, and Andy Nealen. Hyper-Heuristic General Video Game Playing. In *2016 IEEE Conference on Computational Intelligence and Games (CIG)*, pages 1–8. IEEE, 2016.

General Computer Corporation Midway. *Ms. Pac-man*. Midway, Jan 1982.

Ian Millington. *AI for Games*. CRC Press, 3rd edition, Dec 2020.

Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, et al. Human-Level Control Through Deep Reinforcement Learning. *Nature*, 518 (7540):529, 2015.

Jean-Baptiste Mouret and Jeff Clune. Illuminating Search Spaces by Mapping Elites. *arXiv preprint arXiv:1504.04909*, 2015.

Luvneesh Mugrai, Fernando Silva, Christoffer Holmgård, and Julian Togelius. Automated Playtesting of Matching Tile Games. In *2019 IEEE Conference on Games (CoG)*, pages 1–7. IEEE, 2019.

Mark J Nelson. Game Metrics without Players: Strategies for Understanding Game Artifacts. In *Workshops at the Seventh Artificial Intelligence and Interactive Digital Entertainment Conference*, 2011.

Mark J Nelson. Investigating Vanilla MCTS Scaling on the GVG-AI Game Corpus. In *2016 IEEE Conference on Computational Intelligence and Games (CIG)*, pages 1–7. IEEE, 2016.

Thorbjørn S Nielsen, Gabriella AB Barros, Julian Togelius, and Mark J Nelson. General Video Game Evaluation Using Relative Algorithm Performance Profiles. In *European Conference on the Applications of Evolutionary Computation*, pages 369–380. Springer, 2015.

Nils J Nilsson. *Artificial Intelligence: A New Synthesis*. Morgan Kaufmann, Apr 1998.

Nintendo. *Super Mario Bros.* Nintendo, Sep 1985.

Santiago Ontanón, Gabriel Synnaeve, Alberto Uriarte, Florian Richoux, David Churchill, and Mike Preuss. A Survey of Real-Time Strategy Game AI Research and Competition in StarCraft. *IEEE Transactions on Computational Intelligence and AI in games*, 5 (4):293–311, 2013.

Hyunsoo Park and Kyung-Joong Kim. MCTS with Influence Map for General Video Game Playing. In *2015 IEEE Conference on Computational Intelligence and Games (CIG)*, pages 534–535. IEEE, 2015.

Judea Pearl. *Heuristics: Intelligent Search Strategies for Computer Problem Solving.* Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 1984. ISBN 0-201-05594-5.

Barney Pell. METAGAME in Symmetric Chess-Like Games. *University of Cambridge Computer Laboratory*, 1992.

Barney Pell. A Strategic Metagame Player for General Chess-like Games. *Computational Intelligence*, 12(1):177–198, 1996.

Diego Perez, Spyridon Samothrakis, Simon Lucas, and Philipp Rohlfshagen. Rolling Horizon Evolution Versus Tree Search for Navigation in Single-Player Real-Time Games. In *Proceedings of the 15th annual conference on Genetic and evolutionary computation*, pages 351–358, 2013.

Diego Perez, Spyridon Samothrakis, and Simon Lucas. Knowledge-Based Fast Evolutionary MCTS for General Video Game Playing. In *2014 IEEE Conference on Computational Intelligence and Games*, pages 1–8. IEEE, 2014.

Diego Perez Liebana, Jens Dieskau, Martin Hunermund, Sanaz Mostaghim, and Simon Lucas. Open loop Search for General Video Game Playing. In *Proceedings of the 2015 Annual Conference on Genetic and Evolutionary Computation*, pages 337–344, 2015.

Diego Perez-Liebana, Spyridon Samothrakis, Julian Togelius, Tom Schaul, Simon M Lucas, Adrien Couëtoux, Jerry Lee, Chong-U Lim, and Tommy Thompson. The 2014 General Video Game Playing Competition. *IEEE Transactions on Computational Intelligence and AI in Games*, 8(3):229–243, 2015.

Diego Perez-Liebana, Sanaz Mostaghim, and Simon M Lucas. Multi-Objective Tree Search Approaches for General Video Game Playing. In *2016 IEEE Congress on Evolutionary Computation (CEC)*, pages 624–631. IEEE, 2016.

Diego Pérez-Liébana, Spyridon Samothrakis, Julian Togelius, Tom Schaul, and Simon M Lucas. Analyzing the Robustness of General Video Game Playing Agents. In *2016 IEEE Conference on Computational Intelligence and Games (CIG)*, pages 1–8. IEEE, 2016.

Diego Perez-Liebana, Katja Hofmann, Sharada Prasanna Mohanty, Noburu Kuno, Andre Kramer, Sam Devlin, Raluca D Gaina, and Daniel Ionita. The Multi-Agent Reinforcement Learning in MalmÖ (MARLÖ) Competition. *arXiv preprint arXiv:1901.08129*, 2019a.

Diego Perez-Liebana, Jialin Liu, Ahmed Khalifa, Raluca D Gaina, Julian Togelius, and Simon M Lucas. General Video Game AI: A Multitrack Framework for Evaluating agents, Games, and Content Generation Algorithms. *IEEE Transactions on Games*, 11(3):195–214, 2019b.

Diego Perez-Liebana, Simon M. Lucas, Raluca D. Gaina, Julian Togelius, Ahmed Khalifa, and Jialin Liu. *General Video Game Artificial Intelligence*, volume 3. Morgan & Claypool Publishers, 2019c. `https://gaigresearch.github.io/gvgaibook/`.

Diego Perez-Liebana, Cristina Guerrero-Romero, Alexander Dockhorn, Dominik Jeurissen, and Linjie Xu. Generating Diverse and Competitive Play-Styles for Strategy Games. In *2021 IEEE Conference on Games (CoG)*. IEEE, 2021.

Johannes Pfau, Jan David Smeddinck, and Rainer Malaka. Automated Game Testing with ICARUS: Intelligent Completion of Adventure Riddles via Unsupervised Solving. In *Extended abstracts publication of the annual symposium on computer-human interaction in play*, pages 153–164, 2017.

Jacques Pitrat. Realization of a General Game-Playing Program. In *IFIP congress*, pages 1570–1574, 1968.

Steven Rabin. *Game AI Pro: Collected Wisdom of Game AI Professionals.* CRC Press, 2013. ISBN 1466565969.

Steven Rabin. *Game AI Pro 2: Collected Wisdom of Game AI Professionals.* CRC Press, 2015. ISBN 1482254794.

Rare. *Sea of Thieves*. Microsoft Studios, Mar 2018.

Shaghayegh Roohi, Jari Takatalo, Christian Guckelsberger, and Perttu Hämäläinen. Review of Intrinsic Motivation in Simulation-Based Game Testing. In *Proceedings of the 2018 chi conference on human factors in computing systems*, pages 1–13, 2018.

Stuart Russell and Peter Norvig. *Artificial Intelligence: A Modern Approach, Global Edition.* Pearson, 4th edition, May 2021.

Richard M Ryan and Edward L Deci. *Self-Determination Theory: Basic Psychological Needs in Motivation, Development, and Wellness.* Guilford Publications, 2017.

Richard M Ryan, C Scott Rigby, and Andrew Przybylski. The Motivational Pull of Video Games: A Self-Determination Theory Approach. *Motivation and Emotion*, 30 (4):344–360, 2006.

J Schaeffer. One Jump Ahead: Challenging Human Supremacy in Checkers. *ICGA Journal*, 20(2):93–93, 1997.

Markus Schatten, Igor Tomičić, Bogdan Okreša Đurić, and Nikola Ivković. Towards an Agent-Based Automated Testing Environment for Massively Multi-Player Role Playing Games. In *2017 40th International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO)*, pages 1149–1154. IEEE, 2017.

Tom Schaul. A Video Game Description Language for Model-Based or Interactive Learning. In *2013 IEEE Conference on Computational Inteligence in Games (CIG)*, pages 1–8. IEEE, 2013.

Magy Seif El-Nasr, Simon Niedenthal, Igor Kenz, Priya Almeida, and Joseph Zupko. Dynamic Lighting for Tension in Games. *Game Studies Journal*, 7(1), 2007.

Noor Shaker, Mohammad Shaker, and Julian Togelius. Ropossum: An Authoring Tool for Designing, Optimizing and Solving Cut the Rope Levels. In *Ninth Artificial Intelligence and Interactive Digital Entertainment Conference*, 2013.

Claude E Shannon. Programming a Computer for Playing Chess. *Philosophical Magazine*, 7(41):256–275, 1950.

John J Shaughnessy, Eugene B Zechmeister, and Jeanne S Zechmeister. *Research Methods in Psychology*. McGraw-Hill, 2000.

Rafet Sifa, Anders Drachen, Christian Bauckhage, Christian Thurau, and Alessandro Canossa. Behavior evolution in tomb raider underworld. In *2013 IEEE Conference on Computational Inteligence in Games (CIG)*, pages 1–8. IEEE, 2013.

David Silver, Aja Huang, Chris J Maddison, Arthur Guez, Laurent Sifre, George Van Den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, et al. Mastering the Game of Go with Deep Neural Networks and Tree Search. *nature*, 529(7587):484–489, 2016.

Chiara F Sironi, Jialin Liu, Diego Perez-Liebana, Raluca D Gaina, Ivan Bravi, Simon M Lucas, and Mark HM Winands. Self-Adaptive MCTS for General Video Game Playing. In *International Conference on the Applications of Evolutionary Computation*, pages 358–375. Springer, 2018.

Team Meat. *Super Meat Boy*. Team Meat, Oct 2010.

Gerald Tesauro. TD-Gammon, a Self-Teaching Backgammon Program, Achieves Master-Level Play. *Neural Computation*, 6(2):215–219, 1994.

Thatgamecompany. *Journey*. Sony Computer Entertainment, Mar 2012.

Tommy Thompson. Revolutionary Warfare The AI of Total War (Part 3), Feb 2018. URL `https://www.gamedeveloper.com/design/revolutionary-warfare-the-ai-of-total-war-part-3-`. Game Developer.

Julian Togelius, Noor Shaker, Sergey Karakovskiy, and Georgios N Yannakakis. The Mario AI Championship 2009-2012. *AI Magazine*, 34(3):89–92, 2013.

Alan M Turing. Digital Computers Applied to Games. *Faster than thought*, 1953.

Anders Tychsen and Alessandro Canossa. Defining Personas in Games Using Metrics. In *Proceedings of the 2008 Conference on Future Play: Research, Play, Share*, pages 73–80, 2008.

Valve. *Counter-Strike*. Valve, Nov 2000.

JG van Eeden. Analysing and Improving the Knowledge-based Fast Evolutionary MCTS Algorithm. Master's thesis, Utrecht University, Jul 2015.

Jan van Valburg. Automated Testing and Profiling for Call of Duty, 2018. URL `https://www.youtube.com/watch?v=8d0wzyiikXM`. GDC.

Vanessa Volz and Boris Naujoks. Towards Game-Playing AI Benchmarks via Performance Reporting Standards. In *2020 IEEE Conference on Games (CoG)*, pages 764–771. IEEE, 2020.

Vanessa Volz, Dan Ashlock, and Simon Colton. Gameplay Evaluation Measures. *Artificial and Computational Intelligence in Games: AI-Driven Game Design (Dagstuhl Seminar 17471)*, page 122, 2018.

Christopher JCH Watkins and Peter Dayan. Q-learning. *Machine learning*, 8(3-4):279–292, 1992.

Ari Weinstein and Michael L Littman. Bandit-Based Planning and Learning in Continuous-Action Markov Decision Processes. In *Twenty-Second International Conference on Automated Planning and Scheduling*, 2012.

Wizarbox. *So Blonde: Back to the Island*. dtp Entertainment, Mar 2010.

Marek Wydmuch, Michał Kempka, and Wojciech Jaśkowski. Vizdoom Competitions: Playing Doom from Pixels. *IEEE Transactions on Games*, 11(3):248–259, 2018.

Georgios N Yannakakis and Julian Togelius. Experience-Driven Procedural Content Generation. *IEEE Transactions on Affective Computing*, 2(3):147–161, 2011.

Georgios N Yannakakis and Julian Togelius. A Panorama of Artificial and Computational Intelligence in Games. *IEEE Transactions on Computational Intelligence and AI in Games*, 7(4):317–335, 2014.

Georgios N Yannakakis and Julian Togelius. *Artificial Intelligence and Games*. Springer, 2018. `http://gameaibook.org`.

Georgios N Yannakakis, Antonios Liapis, and Constantine Alexopoulos. Mixed-Initiative Co-Creativity. In *9th International Conference on the Foundations of Digital Games*, 2014.

Nick Yee. The Gamer Motivation Profile: What we Learned from 250,000 Gamers. In *Proceedings of the 2016 Annual Symposium on Computer-Human Interaction in Play*, pages 2–2, 2016.

Nick Yee. Gamer Motivation Model - Reference Sheets and Details (v2), Apr 2019a. URL `https://quanticfoundry.com/wp-content/uploads/2019/04/Gamer-Motivation-Model-Reference.pdf`. Accessed: 2021-10-06.

Nick Yee. A Deep Dive into the 12 Motivations: Findings from 400,000+ Gamers, Mar 2019b. URL `https://www.gdcvault.com/play/1025742/A-Deep-Dive-into-the`. GDC.

Imants Zarembo. Analysis of Artificial Intelligence Applications for Automated Testing of Video Games. In *ENVIRONMENT. TECHNOLOGIES. RESOURCES. Proceedings of the International Scientific and Practical Conference*, volume 2, pages 170–174, 2019.

Yunqi Zhao, Igor Borovikov, Fernando de Mesentier Silva, Ahmad Beirami, Jason Rupert, Caedmon Somers, Jesse Harder, John Kolen, Jervis Pinto, Reza Pourabolghasem, et al. Winning is not Everything: Enhancing Game Development with Intelligent Agents. *IEEE Transactions on Games*, 12(2):199–212, 2020.