**JULIAN RICARDO HERNANDEZ MARIÑO**

# A COMPUTATIONAL MODEL FOR GENERATING VISUALLY PLEASING VIDEO GAME MAPS

Dissertação apresentada à Universidade Federal de Viçosa, como parte das exigências do Programa de Pós-Graduação em Ciência da Computação, para obtenção do título de *Magister Scientiae*.

VIÇOSA
MINAS GERAIS - BRASIL
2016

**Ficha catalográfica preparada pela Biblioteca Central da Universidade Federal de Viçosa - Câmpus Viçosa**

T
M339c
2016

Hernandez Mariño, Julian Ricardo, 1988-
A computational model for generating visually pleasing video game maps / Julian Ricardo Hernandez Mariño. – Viçosa, MG, 2016.
x, 59f. : il. (algumas color.) ; 29 cm.

Inclui apêndice.
Orientador: Levi Henrique Santana de Lelis.
Dissertação (mestrado) - Universidade Federal de Viçosa.
Referências bibliográficas: f.55-59.

1. Jogos para computador. 2. Modelo computacional. 3. Mapas de jogos. 4. Video Game. 5. Heuristic Search. 6. Procedural Content Generation. I. Universidade Federal de Viçosa. Departamento de Informática. Programa de Pós-graduação em Ciência da Computação. II. Título.

CDD 22. ed. 794.932

JULIAN RICARDO HERNANDEZ MARIÑO

# A COMPUTATIONAL MODEL FOR GENERATING VISUALLY PLEASING VIDEO GAME MAPS

Dissertação apresentada à Universidade Federal de Viçosa, como parte das exigências do Programa de Pós-Graduação em Ciência da Computação, para obtenção do título de *Magister Scientiae.*

APROVADA: 25 de maio de 2016.

Cláudio Fabiano Motta Toledo

Sabrina de Azevedo Silveira

Levi Henrique Santana de Lélis
(Orientador)

This dissertation is dedicated to my Mother and my brother.

*It ain't what you don't know that gets you into trouble.*
*It's what you know for sure that just ain't so.* - Mark Twain

# Acknowledgements

I would first like to thank God, for allowing me live this unforgettable experience.

I would like to thank my mother Maria and my brother Oscar, for providing me with unfailing support and continuous encouragement. Without their love, I would not be here today.

I would like to thank my advisor Levi, for his guidance, support, invaluable shared knowledge, and permanent assistance. Thanks to him I grew as scientist and as person.

My thanks also goes to the Organization of American States (OEA), Grupo Coimbra de Universidades Brasileiras (GCUB) and CAPES for providing me with the opportunity to study in Brazil with a scholarship.

I thank the departamento de informática at Universidade Federal de Viçosa (UFV) for allowing me to make my Master's studies in that great university.

Last, but not least, my thanks goes to Jessica for her support and company, to my friends in Colombia for their continued support and to my friends in Brazil for making me feel at home.

# Contents

# List of Figures

# List of Tables

# Resumo

HERNANDEZ MARIÑO, Julian Ricardo, M.Sc., Universidade Federal de Viçosa, Maio de 2016. **A computational model for generating visually pleasing video game maps.** Orientador: Levi Henrique Santana de Lelis.

Neste trabalho apresentamos um modelo computacional baseado em teorias de design para gerar mapas de jogos de plataforma visualmente agradáveis. Nós estudamos o problema de geração de mapas como um problema de otimização e provamos que uma versão simplificada do problema é computacionalmente difícil. Em seguida, propomos uma abordagem de busca heurística para resolver o problema de geração de mapas e utilizamos ela para gerar níveis de um clone do Super Mario Bros (SMB), chamado Infinite Mario Bros (IMB). Antes de avaliar os níveis de IMB gerados pelo nosso sistema, realizamos um estudo detalhado das abordagens comumente utilizadas para avaliar o conteúdo gerado por programas de computador. A avaliação utilizada em trabalhos anteriores utiliza apenas métricas computacionais. Embora esses indicadores são importantes para uma avaliação inicial e exploratória do conteúdo gerado, não é claro se são capazes de capturar a percepção do jogador sobre o conteúdo gerado. Neste trabalho, comparamos os conhecimentos adquiridos a partir de um estudo com seres humanos usando níveis de IMB gerados por diferentes sistemas, com os conhecimentos adquiridos a partir de análise dos valores de métricas computacionais. Os nossos resultados sugerem que as métricas computacionais atuais não devem substituir estudos com seres humanos para avaliar o conteúdo gerado por programas de computador. Usando os conhecimentos adquiridos em nosso experimento anterior, foi realizado outro estudo com seres humanos para avaliar os níveis de IMB gerados pelo nosso método. Os resultados mostram a vantagem do nosso método em relação a outras abordagens em termos de estética visual e diversão. Finalmente, foi realizado outro estudo com seres humanos, mostrando que o nosso método é capaz de gerar níveis de IMB semelhantes aos níveis de SMB criados por designers profissionais.

# Abstract

HERNANDEZ MARIÑO, Julian Ricardo, M.Sc., Universidade Federal de Viçosa, May, 2016. **A computational model for generating visually pleasing video game maps.** Adviser: Levi Henrique Santana de Lelis.

In this work we introduce a computational model based on theories of graphical design to generate visually pleasing video game maps. We cast the problem of map generation as an optimization problem and prove it to be computationally hard. Then, we propose a heuristic search approach to solve the map generation problem and use it to generate levels of a clone of Super Mario Bros (SMB) called Infinite Mario Bros (IMB). Before evaluating the levels of IMB generated by our system, we perform a detailed study of the approaches commonly used to evaluate the content generated by computer programs. The evaluation used in previous works often relies on computational metrics. While these metrics are important for an initial exploratory evaluation of the content generated, it is not clear whether they are able to capture the player's perception of the content generated. In this work we compare the insights gained from a user study with IMB levels generated by different systems with the insights gained from analyzing computational metric values. Our results suggest that current computational metrics should not be used in lieu of user studies for evaluating content generated by computer programs. Using the insights gained in our previous experiment, we performed another user study to evaluate the IMB levels generated by our method. The results show the advantage of our method over other approaches in terms of visual aesthetics and enjoyment. Finally, we performed one last user study that showed that our method is able to generate IMB levels with striking similarity to SMB levels created by professional designers.

# CHAPTER 1

# Introduction

Video games have played a significant role on human activities, from satisfying basic human needs such as the need of relaxing to helping solving complex biological problems (Khatib et al., 2011).

Game content such as levels, maps, game rules, textures, stories, items, quests, music, weapons, vehicles, characters, sport commentaries, and other contents can determine the quality of the game. This is because content is essential for the player's immersion. For example, levels in a platform game defines the structure and challenges a player faces in the game. The game's story builds meaning to the actions of non-player characters and other elements that are part of the game, providing purpose and a clear goal to the player. Game rules create meaning to players by allowing or disallowing actions (Togelius, Shaker, & Nelson, 2015a).

Automatically creating content for computer games has become an important field of study in Computer Science. This is because automatically generating content for computer games can reduce the game production time and cost. Moreover, by automatically creating content the game can offer different content each time that is played (Togelius et al., 2015a).

The research area of automatic content generation is known as Procedural Content Generation (PCG). According to (Togelius, Kastbjerg, Schedl, & Yannakakis, 2011) this term is defined as: "*the algorithmical creation of game content with limited or indirect user input*", and according to (Togelius et al., 2015a), PCG can be seen "*as the computer software that can create game content on its own, or together with one or many human players or designers.*". Still

according to (Togelius et al., 2015a), there are different approaches to generate content for games.

Constructive methods, for instance, are methods to build special structures such as dungeons. The constructive methods need an abstract representation of the element, a method for constructing that model, and finally a method for creating the actual geometry of the specific element.

Planning algorithms is another PCG approach, since one way to think about procedurally generating stories is to consider them to be a planning problem. In a game, from an initial state, the player must reach a final state, the objective, as usually defined by an author. The sequence of events performed by the player defines the destiny in a game story.

Another approach is trough of fractals. Fractals are used to generate specific kinds of content, such as terrain. A lot of games include elements of nature such as trees, grass and different types of vegetation. Fractals can be used to represent the variation in constant frequencies that terrain could have. Grammars is another technique for generation of this type of content. With grammars, strings are rewritten in others through production rules. Those generated strings are then considered as drawing instructions for nature elements.

One of the most studied is the search-based approach, which is the approach used in this work. In search-based PCG, it is common the implementation of optimization and evolutionary algorithms to find the best representation of the content according to predefined desired qualities. When this approach is used, first one identifies how the content will be represented. That is, the content could be represented with a number, a structure, almost anything. An example is the content representation in (Togelius et al., 2010), where the representation of Starcraft maps are simply arrays of real numbers. A search algorithm then is implemented to explore different configurations of this content, where a model, usually called *evaluation function* or *fitness function*, will determine if a specific representation has the best properties according to predefined desired qualities. The output of an evaluation function could indicate, for instance, whether a given puzzle-based level has solution, or the aesthetic appeal and challenges of a game map.

One of the main goals for PCG researchers is the creation of content with good quality, through maximization of the player's enjoyment. According to (Shaker, Togelius, & Nelson, 2015), one important aspect that has not been investigated in depth is the aesthetics in game design using PCG, however recent works such as (Reis et al., 2015) show that it is important to generate content with good visual aesthetics, due to the strong correlation between visual aesthetics and

enjoyment. Thus, it is important to investigate the generation of visually pleasing and possibly enjoyable levels.

Several works show that adapted models based on theories of graphical design correlates with the visual aesthetics that humans perceive for graphical user interfaces (GUI) evaluation. For instance, (Bauerly & Liu, 2006) introduced models based on theories of graphical design such as symmetry (defined later in this dissertation) and performed experiments with human subjects. Bauerly and Yili observed in one of their experiments that the subject's perceived visual aesthetics is highly dependent on how symmetrical the GUI is. (Salimun, Purchase, Simmons, & Brewster, 2010) use some models based on theories of graphical design introduced in (Ngo, Teo, & Byrne, 2003) to build a user study about preferences among different theories of design. Among their results, they found that symmetry and cohesion were more influential than the others metrics.

We focus this work on the problem of automatically generating visually pleasing video game maps.

## 1.1 Objectives

The main objective of this work is to develop a computational model for generating visually pleasing video game maps, able to generate content with good aesthetics according to theories of graphical design. *Infinite Mario Bros* (IMB) a clone of *Super Mario Bros* (SMB) will be the testbed of the system produced in this research.

### 1.1.1 Specific Goals

- Develop an optimization-based algorithm that is able to generate visually pleasing game maps.
- Compare the proposed approaches to other PCG methods found in the literature, as well as with content generated by professional designers.

## 1.2 Contributions

In this work we introduce a computational model based on the concept of symmetry to generate visually pleasing maps of platform games. We assume that, by generating symmetrical game maps, one will be generating visually pleasing game maps. We cast the problem of generating symmetrical game maps as an optimization problem and propose a heuristic search approach to solve it, which

is used for generation of IMB levels. We show a simplified version of the problem and show that it is computationally hard.

For validating our symmetry hypothesis, we performed two user studies. First, a user study shows the advantage of our method over other approaches in terms of visual aesthetics and enjoyment. A second user study shows that our method is able to generate SMB maps as visually pleasing as maps created by professional designers.

Before evaluating the levels of IMB generated by our system, we perform a detailed study of the most common approaches used for evaluating the content generated by PCG systems. In this study we show that the most common evaluation techniques, the computational metrics, are important for an initial exploratory evaluation of the content generated and for measuring features that are difficult of measuring without computational assistance, but it is not clear whether they are able to capture the player's perception of the content generated. Comparing the insights gained from a user study with IMB levels generated by different systems with the insights gained from analyzing computational metric values we claim that current computational metrics should not be used in lieu of user studies for evaluating content generated by computer programs.

The content of Chapter 3 appears in the Proceedings of the Eleventh AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment (MARIÑO, Reis, & Lelis, 2015).

## 1.3 Dissertation Structure

The remainder of this work is organized as follows: Chapter 2 presents the background and related work, an empirical study is conducted on Chapter 3 over evaluation metrics for procedurally generated Mario levels. The knowledge gained in this study is used as a basis for the technique used to test the main proposal of this work: A model for generating visually pleasing video game maps, which is presented in Chapter 4. Finally, the conclusions of this work are presented in Chapter 5.

# CHAPTER 2

# Background and Related Work

This chapter presents concepts and techniques used in this work, as well as related works. Section 2.1 describes the key concepts related to PCG and search-based PCG, Section 2.2 presents IMB; the testbed of the experiments in this work. Section 2.3 and section 2.4 describe related works.

## 2.1 Procedural Content Generation

Togelius et al. (Togelius, Kastbjerg, et al., 2011) define the PCG problem as

> *The algorithmical creation of game content with limited or indirect user input.*

According to (Togelius et al., 2015a), a PCG system refers to a system that includes one or more algorithms to generate content from scratch. However a PCG system can work as an AI-assisted game design tool, helping developers, designers and artists to improve game-related content. Examples of generated content are weapons, characters, textures, levels, music, maps, game rules, stories, items, quests, vehicles and sport commentaries. Figure 2.1 shows an example of a map generated by a PCG system in the game of Civilization IV. The main goal of a PCG system is to generate content with good qualities according to predefined intentions. For example, if the main goal of a specific PCG system is the generation of visually pleasing content, a resulting content will be considered good if it has good visual aesthetics.

Figure 2.1: A typical map of civilization IV

There are several commercial video games adopting the PCG approach. Elite is a classical video game that takes advantage of PCG techniques to overcome storage limitations. Elite compresses in a few tens of kilobytes of memory hundreds of star systems. Nowadays there are several commercial games using PCG techniques. As examples, there is Minecraft, a sandbox independent video game that enables players to explore, build constructions, recollect and fight, among other activities. Minecraft uses PCG techniques to generate all the content available in the game's world. Figures 2.2 shows a screenshot of Minecraft. Civilization IV is a turn-based strategy computer game released in 2005. In Civilization IV, the player gathers resources, constructs buildings and manages people to build a civilization while fighting against other civilizations. Civilization IV uses PCG to create maps. In Spelunky the player explores underground tunnels fighting enemies and collecting items. Spelunky uses PCG to generate playable game maps (Togelius et al., 2015a).

While Elite takes advantage of PCG to overcome storage limitations, Civilization IV, Minecraft and Spelunky are benefited by another important feature given by PCG: "Re-playability" (Smith, Gan, Othenin-Girard, & Whitehead, 2011). This feature is very important for commercial purposes, because allows to a player play the same level several times, having different experiences each time.

PCG systems usually employ artificial intelligence (AI) and computational intelligence (CI) techniques to generate content. There are several approaches to generate content for games, however there is one approach strongly investigated: Search-based algorithms. Using search-based algorithms, the PCG is handled as

Figure 2.2: Minecraft. Extracted from (Bayliss, 2012)

a search/optimization problem where the goal is to find the best global optimal value that corresponds to a specific configuration of the content representation (Togelius, Shaker, & Nelson, 2015b).

According to (Togelius, Yannakakis, Stanley, & Browne, 2011), there are different types of search-based algorithms used for PCG problems. Examples of search-based approaches are metaheuristics such as simulated annealing or particle swarm optimization, simple stochastic local search algorithms and evolutionary algorithms. The choice of a specific search-based approach is largely based on the search space size and the available time to search. If the space is very small or the available search time is large, it is common to use an exhaustive search algorithm to know all the possible configurations. When it is too easy to generate good content, maybe a random approach works well, because it doesn't matter the content configuration chosen, there will not be difference for the player's perspective. On the other hand, when the search space is too large or the available time to generate content is short, metaheuristic optimization algorithms should be used instead. Although these approaches are not guaranteed to find an optimal value, they could well find content good enough in a reasonable time.

Regardless of the type of algorithm, in general there are two core components in a search-based approach: A content representation and an evaluation function. The content representation in search-based approaches is an indirect representation of the actual content that a player sees. After the search, that indirect representation is transformed in a direct representation; the actual entities being part of the content (Togelius et al., 2015b). An example is the content rep-

resentation in (Togelius et al., 2010), where the direct representation of Starcraft maps are the actual maps that the player sees, whereas the indirect representation are simply arrays of real numbers. Encoding this indirect representation as part of a search-based approach is easier than do it with a direct representation, also, the task of understanding the representation to decide whether the content is good enough or not is easier too. In evolutionary algorithms, that indirect representation is called *genotype*, later converted in *fenotype*; the actual entities being part of the content. Figure 2.3 shows an indirect representation of a car racing track (Togelius et al., 2007). This track is represented as a set of control points where Bezier curves were employed to connect these points.

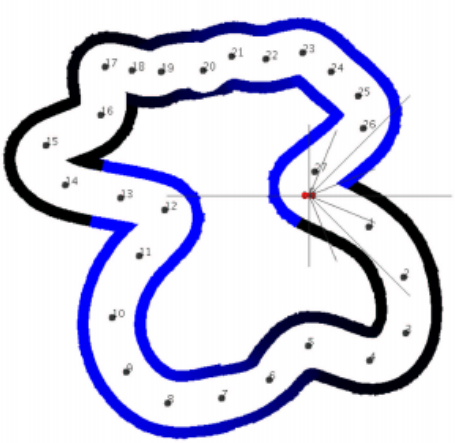


Figure 2.3: Indirect representation of a car racing track. Extracted from (Togelius et al., 2007)

The other component of a search-based approach is an evaluation function. An evaluation function determines the fitness value of the content. During the search, the search-based algorithm will find several different configurations, each one with a fitness value given by the evaluation function (Togelius, Yannakakis, et al., 2011). The evaluation function is built based on the predefined desirable content qualities, e.g. difficulty, immersion, fun, visual aesthetics, etc. As an example, (Ferreira & Toledo, 2014) implement a genetic algorithm to generate Angry birds levels, a physics-based video game that consists in using a slingshot to throw birds against structures composed of pigs and blocks, where the aim is to kill all the pigs. In this work, the main goal is to generate interesting levels based on the stability of the structures. They then implement an evaluation function that evaluates a level based on a simulation that measures the stability of the objects composing the level.

In this work we introduce a search-based approach for generating visually pleasing 2D platform game maps. Our approach uses a branch and bound

approach to create levels which have optimal symmetry values—our evaluation function, formally defined in Section 4.2.

## 2.2 Infinite Mario Bros

Infinite Mario Bros (IMB) is an open source clone of Nintendo's platform game Super Mario Bros (SMB). Several modern game consoles still develop versions of this classic game (Togelius, Shaker, Karakovskiy, & Yannakakis, 2013a). A screenshot of IMB is shown in Figure 2.4. The gameplay in IMB and SMB consists in reaching the rightmost spot of the level while avoiding and killing enemies such as turtles and cannons.

A level of IMB can be seen as a grid space containing objects such as platforms, mountains, enemies, pipes and cannons. Every object has a specific location on the grid ($x$ and $y$ coordinates). The PCG problem in IMB is to find appropriates $x$ and $y$ coordinates for a set of elements, according to predefined desired level features.

Our experiments are run on IMB because this framework is an excellent testbed for PCG systems; since the game allows a versatile (Togelius, Karakovskiy, Koutník, & Schmidhuber, 2009) and quite entertaining testbed (Reis et al., 2015).


Figure 2.4: IMB level. Extracted from (Reis et al., 2015)

## 2.3 PCG Approaches for Platform Games

In this section we review strategies used by other authors for automatically generating platform levels; we focus mainly on Mario games.

For evaluating the maps generated by our proposed model, we first perform a study comparing different evaluation strategies for IMB PCG systems (See Chapter 3); for this reason the works then are classified according the different techniques for evaluating systems. We group the works as follows: works based on user studies to evaluate PCG systems (user studies for evaluation), works using user studies not to evaluate PCG systems, but to collect data to learn predictive models (user studies for data collection), works using computational metrics and/or artificial agents to evaluate PCG systems (computational evaluation), and works that evaluate PCG systems with self critique or some other sort of evaluation.

### 2.3.1 User Studies for Evaluation

(Shaker, Yannakakis, & Togelius, 2010) describe a system for generating adaptive player-tailored IMB levels. Their system directly asks questions to the players about their preferences. Shaker et al.'s main experiment is carried out with artificial agents, but an experiment with human subjects compares the proposed adaptive approach with a non-adaptive one. (Dahlskog & Togelius, 2013) use evolutionary algorithms to generate IMB leves based on design patterns with reference to SMB. Dahlskog et al., also present an user study comparing different levels of SMB. (Bakkes et al., 2014) describe a system for balancing game challenging in IMB levels; their system is also evaluated with human subjects. (Reis et al., 2015) describe a system which uses human computation to evaluate small portions of levels generated by an existing system for IMB levels. They tested their system with human subjects.

### 2.3.2 User Studies for Data Collection

(Pedersen, Togelius, & Yannakakis, 2009) presented a system for modeling player experience based on empirical data collected in a user study. They were aiming at learning statistical models for predicting, given an IMB level **L**, the challenge **L** will offer to the player, and how much enjoyment and frustration the player will have while playing **L**. Similarly, in different works, (Shaker, Yannakakis, & Togelius, 2011, 2012, 2013; Shaker, Asteriadis, Yannakakis, & Karpouzis, 2013, 2011) showed how to extract features to learn predictive models

of the player's experience in IMB. Pederson et al.'s and Shaker et al.'s long-term goal is to use these models to guide the search for good-quality player-tailored IMB levels.

### 2.3.3 Computational Evaluation

(Smith & Whitehead, 2010) and (Horn, Dahlskog, Shaker, Smith, & Togelius, 2014) introduced several computational metrics for evaluating what the authors called the expressivity of PCG systems—we describe some of these metrics in Chapter 3. Their metrics were used in several works as a form of evaluating PCG systems.

For example, (Smith, Whitehead, & Mateas, 2010) presented Tanagra, a system for generating levels of 2D-platform games which was evaluated solely with computational metrics. Later, (Smith, Treanor, et al., 2011) presented Launchpad, a system that uses rhythm groups to generate levels of platform games. (Smith, Cha, & Whitehead, 2008) introduce the idea of Rhythm groups, which are defined as alternating periods of high and low challenge. Launchpad was also evaluated with computational metrics. (Shaker, Nicolau, Yannakakis, Togelius, & O'Neill, 2012) use a grammar to concisely encode design constraints for evolving IMB levels. Shaker et al.'s system is also solely evaluated with computational metrics similar to those used by Smith et al. to evaluate Tanagra and Launchpad. In another work (Shaker, Yannakakis, Togelius, Nicolau, & O'Neill, 2012) evaluate the personalized content generated by a grammar-based PCG system with artificial agents. In a recent work (Shaker & Abou-Zleikha, 2014) used non-negative matrix factorization to generate levels based on patterns learned from levels generated by other systems; their method is also evaluated with computational metrics.

(Dahlskog, Togelius, & Nelson, 2014; Dahlskog & Togelius, 2014a, 2014b) present systems which use patterns to generate levels for SMB games. All these works were evaluated with the computational metrics introduced by Smith and Whitehead (Smith & Whitehead, 2010).

(Sorenson, Pasquier, & DiPaola, 2011) presented a system which uses rhythm groups to define a computational model of player enjoyment to evolve levels of IMB. This model is also used to evaluate the resulting levels. Sorenson et al. also evaluate their system in terms of the results of the Mario AI Competition (Togelius, Shaker, Karakovskiy, & Yannakakis, 2013b).

### 2.3.4 Other Evaluation Strategies

Some PCG systems are evaluated neither with user studies nor with computational metrics. For example, the Occupancy-Regulated Extension (ORE) PCG system is evaluated by the authors themselves with an analysis of the levels generated (Mawhorter & Mateas, 2010). (Kerssemakers, Tuxen, Togelius, & Yannakakis, 2012) also presents a critique based in the opinion of the authors of the proposed approach in addition to an empirical running time analysis of the system. The seminal work of (Compton & Mateas, 2006) on content generation for platform games does not present evaluations of the proposed approach.

Recently, (A & Smith, 2015) introduced several metrics based on design theory for evaluating IMB levels. Their metrics were obtained through discussions with design students. However, in contrast with the computational metrics introduced by (Smith & Whitehead, 2010) and (Horn et al., 2014), some of those metrics are not formal enough to be implemented as a computer procedure.

## 2.4 Visually Pleasing Models

In this work we introduce a computational model based on theories of graphical design. To the best of our knowledge, this is the first work presenting a heuristic search algorithm working with a model based on theories of graphical design to generate platform game maps. To be specific, our heuristic search approach creates symmetrical game maps. We assume that by creating symmetrical maps we will be creating visually pleasing content.

(Liapis, Yannakakis, & Togelius, 2012) introduce a search-based approach to game content generation according to visual aesthetics where one of their evaluation functions is based on visual properties inspired by psychological and neurobiological research. Although they also use symmetry as one of the included properties, Liapis et al.'s system creates 2D game spaceships. By contrast, we are interested in map generation. In another work, (Liapis, 2016) evaluates game content created by evolutionary approaches according to visual features such as balance and shape's complexity. Similar to (Liapis et al., 2012), (Liapis, 2016) is focused on creating spaceship shapes.

Several other works use computational models based on theories of graphical design to evaluate content in general, not necessarily video game-related content. We focus our review on models that use the concept of symmetry.

(Bauerly & Liu, 2006) introduce methods to quantitatively analyze the composition of an image. The models measure symmetry and balance. Bauerly and Liu performed experiments with human subjects using images composed by

simple black and white objects and also images of web page interfaces. Bauerly and Liu observed that the visual aesthetics of images can be highly correlated with the symmetry of the objects composing the image. (Lai, Chen, Shih, Liu, & Hong, 2010) use symmetry and balance models based on the work of (Bauerly & Liu, 2006). Lai et al.'s experiments were performed on text-overlaid images. Their experiments suggest a relationship between a higher averaged visual balance and the aesthetic appeal of this type of images.

(Ngo, Samsudin, & Abdullah, 2000) and (Ngo et al., 2003) propose several aesthetics measures for graphic displays from theories of design. They introduce models that measure balance, equilibrium, symmetry, sequence, cohesion, unity, proportion, simplicity, density, regularity, economy, homogeneity, rhythm, and order and complexity. (Salimun et al., 2010) use some of the Ngo et al.'s models to build a user study about preferences between different theories of design. More specifically, Salimun et al. present the results of a study on the preference ranking of cohesion, economy, regularity, sequence, symmetry, and unity. Their user study was performed using images rated according to the Ngo et al.'s models. Among their results, they found that symmetry and cohesion were more influential than other metrics.

(Browne, 2012) is the first in propose computational models for the estimation of Shibui, a concept of visual aesthetics. Shibui includes several attributes, including the concept of asymmetry. The model proposed correlates with accepted principles of good game design. The main goal of the models is to evaluate the visual aesthetics of two-player combinatorial games.

# CHAPTER 3

# Evaluation of Computational Metrics

There are various approaches in the literature for automatically generating IMB levels. The evaluation of many approaches is often performed solely with computational metrics such as *leniency* and *linearity* (Smith & Whitehead, 2010; Horn et al., 2014). While these metrics are important for performing initial exploratory evaluations of the levels generated, it is not clear whether they are able to capture the player's perception of the content generated. A focus in many PCG research projects is to know whether the content generated has good quality from the player's perspective, and the literature lacks a systematic evaluation of the computational metrics used for evaluating PCG systems.

In this chapter, we perform a systematic user study with IMB PCG systems and compare the insights gained from our study with those gained from analyzing a set of commonly used computational metrics. As an example of the results we present in this chapter, all computational metrics used in our experiment rated the levels generated by two PCG systems very similarly, while subjects in our user study found that the levels generated by one of the systems were significantly more enjoyable to play than the levels generated by the other system. As another example, the computational metric of leniency, which was designed to approximate the difficulty of a given level, only weakly correlated with the difficulty rated by the subjects in our study. Perhaps the most important conclusion one can draw from our experiments is that although the computational metrics can be valuable for an initial exploratory study of the content generated by PCG systems and for verifying the diversity of levels a PCG system can generate, these metrics should not replace user studies for analyzing the player's perception of

the content generated.

This chapter is organized as follows. First, in Section 3.1, we define the PCG problem for Infinite Mario Bros. Following, in Section 3.2, we explain the computational metrics evaluated in our experiment. Finally, in Section 3.3 we describe our experiment and discuss the results obtained.

## 3.1 The PCG Problem for Infinite Mario Bros

In this chapter we are interested in the problem of evaluating the content generated by PCG systems for the game of IMB. The levels of IMB are grid spaces containing a set of objects such as platforms, mountains and shooting cannons. Every object is associated with a location on the grid ($x$ and $y$ coordinates) and some of the objects such as mountains can have different heights and widths.

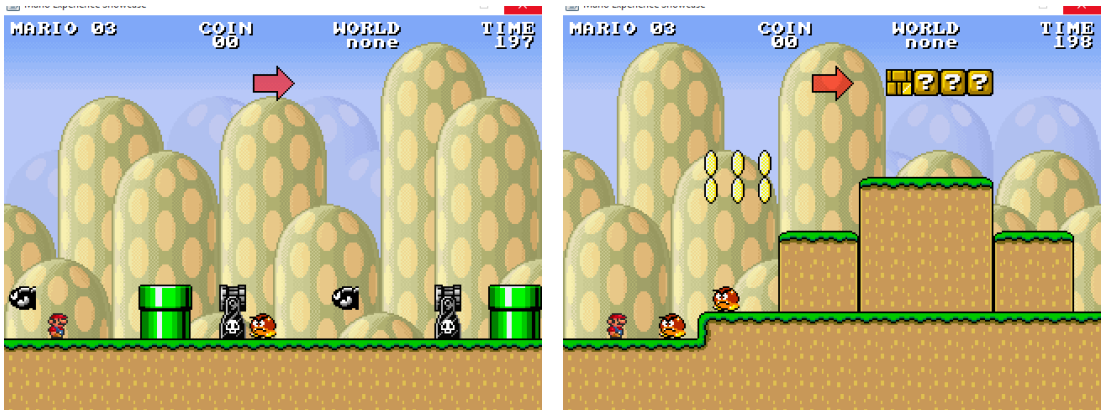
Let $\mathbf{L} = \{o_1, o_2, \cdots, o_n\}$ be a level of IMB where $o_1, o_2, \cdots, o_n$ are the $n$ objects composing the level. The PCG problem for IMB is to choose the set of objects in $\mathbf{L}$ as well as the objects' $x$ and $y$ coordinates. For some of the objects such as pits and mountains the PCG system also needs to define their height and width values. In this work we assume that the goal in PCG for IMB is to generate levels which are both visually appealing and enjoyable to play.

## 3.2 Computational Metrics

In this section we describe the computational metrics used in our experiment: *linearity* and *leniency* introduced by (Smith & Whitehead, 2010), *density*, and *Compression Distance* introduced by (Shaker, Nicolau, et al., 2012). Similarly to previous works, to ease the presentation of the results, we normalize all metrics to the $[0, 1]$ interval. Normalization is performed by accounting for the levels generated by all systems evaluated. Thus, the metric values we present in this work are not directly comparable to the values presented in other works as the normalized values depend on the systems evaluated. We note that the normalization we perform does not affect the results of our experiment.

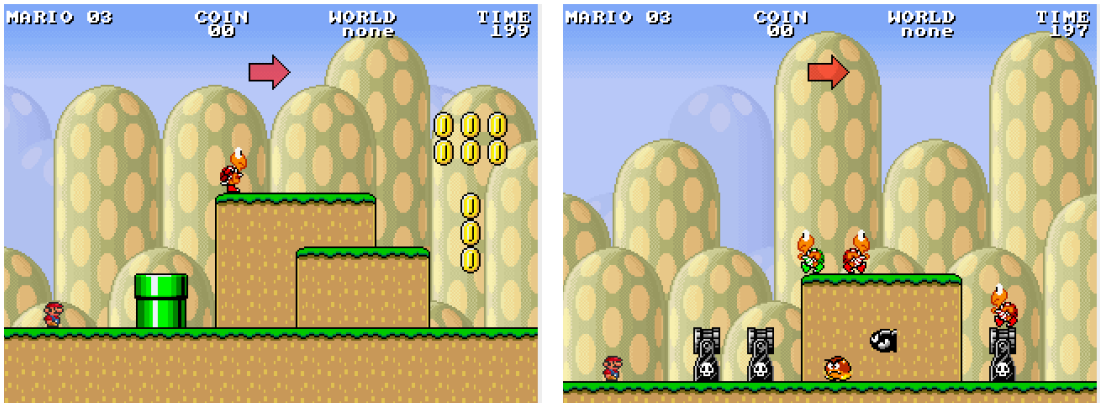
### 3.2.1 Linearity

The linearity of level $\mathbf{L}$ is computed by performing a linear regression on the center points of each platform and mountain contained in $\mathbf{L}$. Each possible position in the $x$ axis is considered as a different point for the linear regression. The linearity of $\mathbf{L}$ is the average distance between the center points and the linear




(a) Linear level. (b) Non-linear level.

Figure 3.1: Linear level vs Non-linear level




(a) High-leniency level. (b) Low-leniency level.

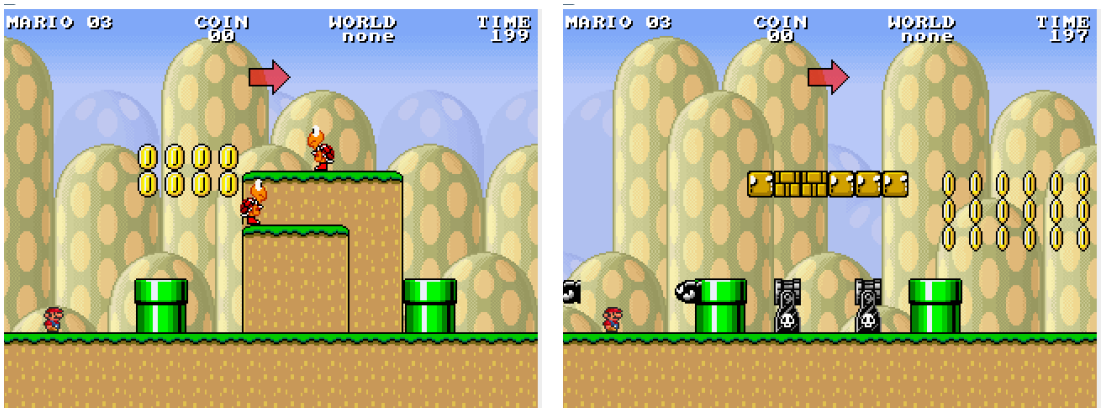
Figure 3.2: High-leniency level vs Low-leniency level

regression's line. Normalized values closer to one indicate more linear levels. The linearity of a PCG system $\rho$ is the average normalized linearity of the levels $\rho$ generates.

Linearity measures the changes in height ($y$-coordinate) the player experiences while going through the level. (Smith, Treanor, et al., 2011) pose the linearity metric as a visual aesthetics metric, which is reasonable since levels with different linearity values are expected to look different from one another. Figure 3.1 shows the contrast between a linear and a non-linear level.

### 3.2.2 Leniency

Leniency measures how much challenge the player is likely face while playing the level. The leniency of level $\mathbf{L}$ is the sum of the lenience value $w(o)$ of all objects $o$ in $\mathbf{L}$, defined as $\sum_{o \in \mathbf{L}} w(o)$. We use the lenience values specified by (Shaker, Nicolau, et al., 2012). Namely, power-up items have a weight of 1,




(a) High-density level. (b) Low-density level.

Figure 3.3: High-density level vs Low-density level

cannons, flower tubes, and gaps of $-0.5$, and enemies of $-1$. We subtract the average gap width of the level from the resulting sum as defined by Shaker et al. Leniency is meant to approximate the difficulty of the levels. Normalized values closer to one indicate more lenient levels. The leniency of a PCG system $\rho$ is the average normalized leniency of the levels $\rho$ generates. Figure 3.2 shows the contrast between a high-leniency level and a low-leniency level.

### 3.2.3 Density

Mountains can occupy the same $x$-coordinate on the grid defining a IMB level by being "stacked-up" together. The density of $\mathbf{L}$ is the average number of mountains occupying the same $x$-coordinate on the grid. Intuitively, a level with high density could have different layouts and challenges than a level with low density. Normalized values closer to one indicate denser levels. The density of a PCG system $\rho$ is the average normalized density of the levels $\rho$ generates. Figure 3.3 shows the contrast between a high-density level and a low-density level.

### 3.2.4 Compression Distance

The Compression Distance (CD) measures the structural dissimilarity of a pair of levels. CD is computed as follows. First, we convert the pair of levels $\mathbf{L}$ and $\mathbf{L}'$ into two sequences of integers $\mathbf{S}$ and $\mathbf{S}'$, respectively. Each integer in $\mathbf{S}$ represents one of the following in $\mathbf{L}$: (i) an increase or decrease in the platform's height, (ii) the existence or the nonexistence of enemies and items, and (iii) the beginning or ending of a gap. The conversion of $\mathbf{L}$ into $\mathbf{S}$ is done by traversing the level's grid from left to right and for each $x$-value on the grid we insert the appropriate integer into the converted sequence (e.g., the integer 1 in position 10

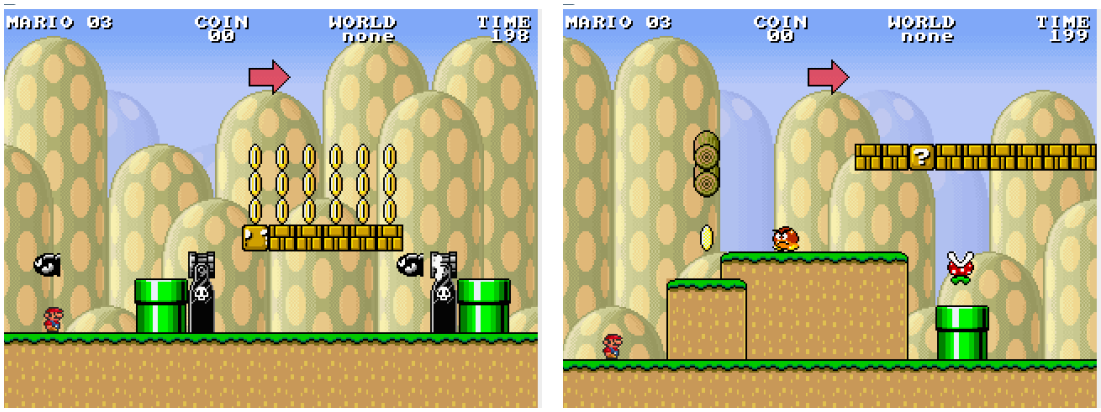



Figure 3.4: Two levels with high Compression Distance

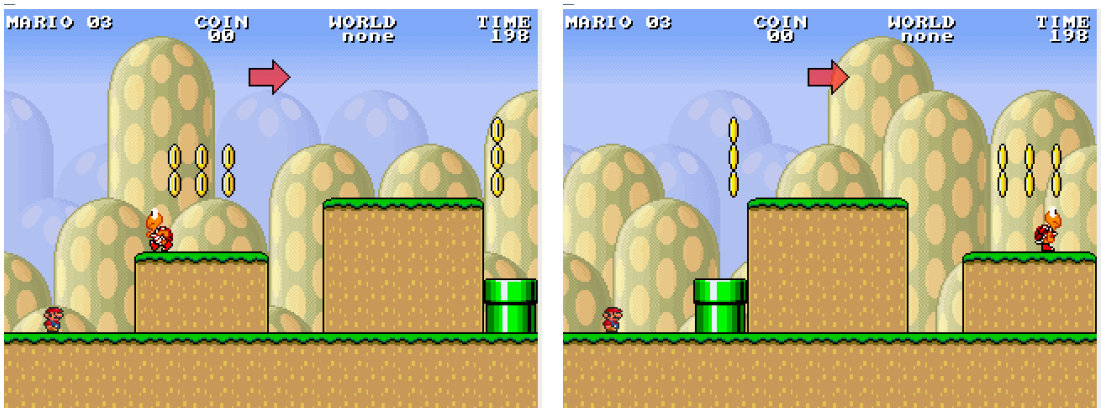



Figure 3.5: Two levels with low Compression Distance

could represent an enemy at $x$-coordinate 10 of the level). Intuitively, if sequences $\mathbf{S}$ and $\mathbf{S}'$ are very different, then one would expect $\mathbf{L}$ and $\mathbf{L}'$ to be structurally different.

The CD value of a PCG system $\rho$ is the average normalized compression metric (Li, Chen, Li, Ma, & Vit‡nyi, 2004) of $\mathbf{S}$ and $\mathbf{S}'$ for pairs of levels $\mathbf{L}$ and $\mathbf{L}'$ $\rho$ generates. Normalized values closer to one indicate that the PCG system is able to generate levels with a larger structural variety. Figure 3.4 and Figure 3.5 show a version of a system able to generate levels with high CD values, and a system able to generate levels with lower CD values respectively.

## 3.3 Evaluating Metrics

We now evaluate the computational metrics described above. First we describe the methodology of our experiment. Then, we present the results of the computational metrics, followed by the results of the user study. Finally, we discuss the insights gained from each evaluation.

### 3.3.1 Methodology

#### Systems Tested

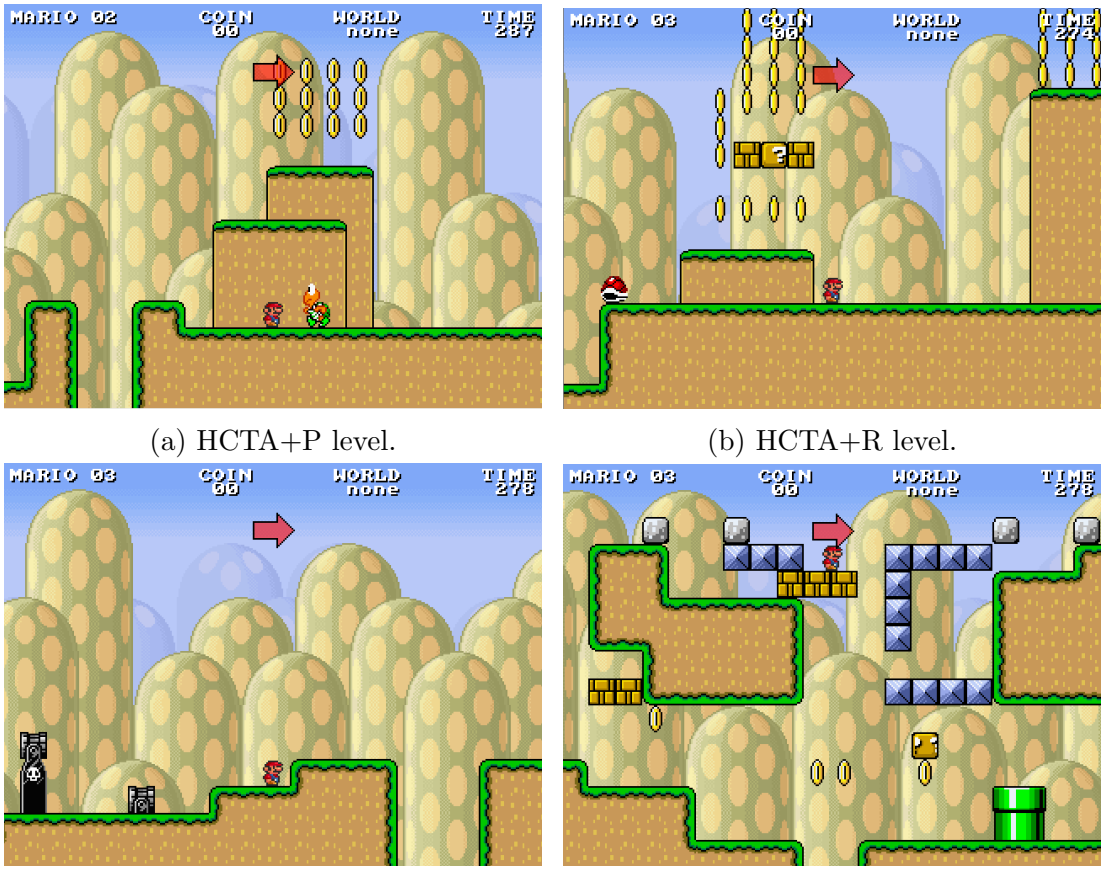
We used four different IMB PCG systems in our experiments: Notch Level Generator (NLG), Human-Computation Tension Arc-Based (HCTA) level generator with a random tension arc (HCTA+R) and with a parabolic tension arc (HCTA+P) (Reis et al., 2015), and Occupancy-Regulated Extension (ORE) generator (Mawhorter & Mateas, 2010).

The NLG system receives as input a difficulty value $d$ for stochastically determining the number of enemies and challenges to be placed in the level. The levels NLG generates tend to be harder for larger values of $d$. NLG starts with an empty level grid and adds objects to the grid according to the value of $d$. HCTA+R and HCTA+P are variants of NLG. The HCTA systems work by having human subjects rating a set of small levels generated by NLG. Then, HCTA combines the small levels into a regular-sized IMB level according to the human-rated difficulty of the small levels. HCTA+P combines the small levels into a regular level in a way that the difficulty of the resulting level follows a parabolic curve: difficulty increases as the player progresses into the level until reaching its largest value, then difficulty decreases until the end of the level. HCTA+R combines the small levels in a way that the difficulty is random (but still respecting a user-specified upper bound) throughout the level. See (Reis et al., 2015) for details on HCTA.

We chose NLG, HCTA+R, HCTA+P, and ORE for two reasons. First, the computational metrics will tend to give similar scores to the levels HCTA+R and HCTA+P generate since they both use similar strategies for level generation. Yet, levels generated by the HCTA systems could still be rated differently by the participants in the user study. Second, ORE generates levels which are structurally different from the ones the other systems generate, allowing us to verify whether the user study is able to capture nuances which are likely to be captured by the computational metrics. Ideally we would use more systems in our experiment, but the time required for each participant to complete the experiment could be prohibitively long should we required them to play extra levels. Figure 3.6 shows a screenshot of a typical level according the four systems used in this experiment.

#### Participants

Our within-subject experiment had 37 participants: 32 males and 5 females with an average age of 23.95 and standard deviation of 4.48. Each participant played one level generated by each system, resulting in the evaluation of 37 levels




(a) HCTA+P level. (b) HCTA+R level.

(c) NLG level. (d) ORE level.

Figure 3.6: Systems used for user study in the evaluation metrics experiment

of each PCG system. The experiment was carried out online: our system was made available in the Internet and our experiment advertised in different mailing lists. Participation was anonymous and volunteered.

### Evaluated Metrics

In the user study the systems are evaluated according to the following criteria: enjoyment, visual aesthetics, and difficulty. Each participant was asked to answer how much they agreed or disagreed, in a 7-likert scale, with the following sentences: "This level is enjoyable to play"; "this level has good visual aesthetics"; "this level is difficult". A score of 1 for enjoyment and visual aesthetics means that the participant strongly agrees that the level played is enjoyable and has excellent visual aesthetics; a score of 1 for difficulty means that the participant strongly agrees that the level is difficult.

We compute the computational metric values only for the levels evaluated in our user study: 148 levels in total (37 levels for each of the four systems). This is to allow a fair comparison of the insights gained from the computational metrics

with those gained from the user study.[1] The normalization of the computational metrics to the $[0, 1]$ interval was made by considering all 148 levels used in our experiment.

### Experimental Design

In the beginning of the experiment the subjects filled a questionnaire informing their age, and their skills in the game of Mario Bros. Subjects were instructed about the controls of the game before playing a practice level. The practice level is important so that the participants get acquainted with the controls of the game. NLG was used to generate the practice levels. Only after playing the practice level that the participants evaluated the levels generated by the PCG systems. Each participant played one level generated by each of the four PCG systems. After playing each level the participants gave scores according to the criteria described above in a 7-likert scale. In addition to the scores, the participants had the option to enter comments informing us of technical issues they might have had during the experiment. Since all participants played one level generated by each system, we used a balanced Latin square design to counteract ordering effects. The tested levels were generated during the experiment by the evaluated systems, we did not pre-select a set of levels to be tested.

In order to have a fair comparison of the levels generated by different systems we had all systems generating levels of the same size: $160 \times 15$. We chose this size because we did not want the experiment to be too long. In total each participant played 5 levels (1 practice level and 4 other levels for evaluation), and using larger levels could be tiring for the participants. Finally, to ensure a fair comparison of the different approaches, we tuned the systems to generate levels with similar difficulty. This was done by manually setting the $d$-values of NLG, HCTA+P, and HCTA+R so that the three systems generated levels which we thought to be of difficulty similar to the ones generated by ORE.

### Data Cleaning

The data of participants who did not finish playing all 5 levels (1 practice level plus 4 levels to be evaluated) is not included in the results. We also removed the data of one participant who had never played the game of Mario before. By examining the logs of the experiment we noticed that this participant was not able to get too far into the game and thus not able to properly evaluate the levels. The number of 37 participants was obtained after cleaning the data.

[1] We also computed the computational metrics for a larger number of levels and observed results similar to the ones we report in this work.

| | HCTA+P | HCTA+R | ORE | NLG |
|---|---|---|---|---|
| Leniency | $0.45 \pm 0.10^a$ | $0.48 \pm 0.18^a$ | $0.71 \pm 0.11^b$ | $0.77 \pm 0.15^b$ |
| Linearity | $0.52 \pm 0.17^a$ | $0.52 \pm 0.15^a$ | $0.33 \pm 0.14^c$ | $0.83 \pm 0.07^b$ |
| Density | $0.74 \pm 0.13^a$ | $0.73 \pm 0.12^a$ | $0.17 \pm 0.15^c$ | $0.49 \pm 0.09^b$ |
| CD | $0.61 \pm 0.02^a$ | $0.61 \pm 0.02^a$ | $0.60 \pm 0.02^a$ | $0.56 \pm 0.02^a$ |

Table 3.1: Computational metric results. Larger values of leniency, linearity, and density indicate are more lenient, linear, and dense levels; larger values of CD indicate that the PCG system is able to generate a larger variety of structurally different levels. Different letters in the same row indicate statistically significant results.

### 3.3.2 Computational Metric Results

We start by presenting the computational metric results. Although the computational metrics are systematic and do not represent a source of variance in our experiment, all PCG systems are stochastic and insert variance in the results. Moreover, as we explained above, the number of levels considered in this experiment is somewhat limited (37 levels for each system). Therefore, we present statistical tests for the computational metric results. Table 3.1 shows the average value and standard deviation for each metric and PCG system. Different letters in a given row of the table indicate that the two means are significantly different.

We now explain how the statistical significance was computed for the results in Table 3.1. First, we ran Shapiro-Wilk tests for each metric and verified that the leniency, density, and CD values were unlikely to be normally distributed. Thus, repeated-measures ANOVA was used only for linearity, and the test indicated statistically significant results ($p < .001$). The non-parametric Friedman test was applied to remaining metrics and indicated statistically significant results for leniency and density ($p < .001$), the differences in CD were not significant. Pairwise comparisons with Tukey tests for linearity showed that the only averages that are not significantly different are those of HCTA+P and HCTA+R, all other differences are significant ($p < .001$). Pairwise comparisons with Wilcoxon signed-rank tests for leniency and density showed that the averages that are not significantly different are those of the HCTA+P and the HCTA+R systems for both leniency and density, and ORE and NLG for leniency; all other results are statistically significant ($p < .001$).

We highlight the following observations from Table 3.1.

1. HCTA+P and HCTA+R generate similar levels as both systems scored similarly in all four metrics tested.

2. The average leniency value of the HCTA systems are much lower than ORE

| | HCTA+P | HCTA+R | ORE | NLG |
|---|---|---|---|---|
| Enjoyment | $2.24 \pm 1.75^{a}$ | $2.70 \pm 1.91^{b}$ | $3.35 \pm 2.04^{c}$ | $2.62 \pm 2.00^{ab}$ |
| Visual Aesthetics | $2.32 \pm 1.65^{a}$ | $2.38 \pm 1.64^{ab}$ | $3.43 \pm 2.21^{c}$ | $2.92 \pm 1.93^{b}$ |
| Difficulty | $3.46 \pm 1.76^{a}$ | $3.38 \pm 1.72^{a}$ | $3.27 \pm 1.90^{a}$ | $3.84 \pm 2.35^{a}$ |

Table 3.2: User study results. Lower values of enjoyment and visual aesthetics indicate levels which are more enjoyable to play and have better visual aesthetics; lower values of Difficulty indicate levels which participants found more challenging to play. Different letters in the same row indicate statistically significant results.

and NLG, indicating that the levels generated by HCTA are more difficult than those generated by the other two systems.

3. The HCTA approaches generate levels with nearly equal linearity averages, ORE generates highly non-linear levels, and NLG generates highly linear levels. The linearity results suggest that the HCTA approaches generate levels with similar visual aesthetics while NLG and ORE generate levels which are visually different than the levels generated by the other systems.

4. The density averages follow a pattern similar to linearity's: the HCTA approaches have very similar values while NLG and ORE differ from the other systems. The density results indicate that the HCTA approaches often use the pattern of superposing mountains while ORE rarely uses such a pattern. Similarly to linearity, the difference in the density average values show that the levels generated by ORE are visually different than the levels generated by other systems.

5. The difference on the average values of CD is minimal, indicating that all systems generate levels with similar structural diversity.

### 3.3.3 User Studies Results

We now present the user study results. The mean results and standard deviations are shown in Table 3.2. Different letters in a given row indicate that the two means are significantly different. Shapiro-Wilk tests showed that our data is unlikely to be normally distributed ($p < .0001$ for all criteria). Thus, we used the non-parametric Friedman test which showed a significant difference on enjoyment ($p < .05$) and on visual aesthetics ($p < .05$) across different systems; there was no significant difference for difficulty.

Next, we use Wilcoxon signed-rank tests to perform pairwise comparisons of the results obtained by the evaluated systems. We present the effect size of the comparisons ($r$-values) in addition to $p$-values. HCTA+P generates levels

which are significantly more enjoyable to play than the levels HCTA+R generates ($p < .05$, $r = 0.21$) and the levels that ORE generates ($p < .001$, $r = 0.35$). The levels HCTA+R generates are significantly more enjoyable to play than the ones ORE generates ($p < .05$, $r = 0.26$). Finally, the levels NLG generates are significantly more enjoyable to play than the ones ORE generates ($p < .05$, $r = 0.24$).

Pairwise comparisons on visual aesthetics (Wilcoxon signed-rank test) showed that HCTA+P generates levels with significantly better visual aesthetics than the levels ORE generates ($p < .01$, $r = 0.27$) and than the levels NLG generates ($p < .05$, $r = 0.24$). HCTA+R generates levels with significantly better visual aesthetics than the levels ORE generates ($p < .01$, $r = 0.37$).

All pairwise comparisons reported as statistical significant have effect sizes around the medium size mark of 0.3, indicating substantial differences among the levels generated by the different systems.

We highlight the following observations from Table 3.2.

1. The system that generates the most enjoyable levels is HCTA+P. The difference between enjoyment of HCTA+P and HCTA+R is significant and substantial. That is, HCTA+P yielded an average score of 2.24 which is close to 2 (score marked by participants who *agreed* that the level played is enjoyable). By contrast, HCTA+R yielded an average score of 2.70 which is close to 3 (score marked by participants who *somewhat agreed* that the level played is enjoyable).

2. The HCTA approaches generated the levels with best visual aesthetics, followed by NLG and then ORE. In particular, HCTA+P generates levels with significantly better visual aesthetics than NLG and ORE.

3. There is little difference amongst the difficulty scores of the systems, indicating that the evaluated systems generate levels with similar difficulty.

Next, we discuss the strengths and weaknesses of the user study evaluation and of the computational evaluation by comparing the conclusions drawn from the two evaluations.

### 3.3.4 Strengths of the User Study Evaluation

We organize the discussion of the strengths of the user study evaluation by the evaluated criteria.

### Enjoyment

The user study shows a significant and substantial difference between the average enjoyment score of the levels generated by HCTA+P and by HCTA+R, while the computational evaluation yielded nearly the same score for both systems in all metrics. Spearman correlation tests between enjoyment and the computational metrics yielded coefficients close to zero, indicating that none of the metrics correlated with enjoyment.

Enjoyment is perhaps the most important evaluation criterion for PCG systems as we are interested in generating content which users find enjoyable to play. The computational metrics used in our experiment were not able to estimate the player's enjoyment. This result is not surprising. First, none of the computational metrics used in the literature were designed for measuring enjoyment. Second, enjoyment is difficult to measure without accounting for human input as it depends on various factors such as cultural background.

Our user study required the participants to answer questions after playing each level. Another promising way of receiving human input for evaluating PCG systems is by analyzing facial expressions of the players (Shaker & Shaker, 2014; Tan, Bakkes, & Pisan, 2014).

### Visual Aesthetics

Both linearity and density indicated that HCTA+P and HCTA+R would generate levels with similar visual aesthetics, while ORE and NLG would generate levels with different visual aesthetics. The user study indicates that the HCTA approaches have nearly the same score for visual aesthetics, while ORE and NLG have higher values (indicating worse visual aesthetics). While the computational metrics indicated levels with different visual aesthetics, the metrics are not able to distinguish good from bad aesthetics. By contrast, through the user study we are able to rank the systems with respect to the visual quality of the levels generated. A Spearman's test shows that linearity weakly correlates with visual aesthetics (coefficient of 0.18 and $p < .05$); none of the other metrics correlates with visual aesthetics.

### Difficulty

While there is a large difference in the leniency values of the systems tested, according to the user study, there is little or no difference in the difficulty rated by the participants. Difficulty is also an important criteria for evaluating PCG systems as it is closely related to enjoyment. That is, it is known that the

Yerkes-Dodson law (Yerkes & Dodson, 1908) applies to computer games in the sense that enjoyment will be maximum somewhere in between the largest and the smallest difficulty (Piselli, Claypool, & Doyle, 2009). Thus, when comparing different PCG systems the difficulty of the levels generated should be controlled to yield a fair comparison of the systems. It is hard to automatically measure difficulty because difficulty depends on factors such as the disposition of objects on the level. For example, there could be a level full of enemies and challenges (non-lenient level according to the metric) but with an easy path for Mario to follow and win the game—in such cases leniency will be misleading. Although our leniency results were somewhat misleading, we observed a weak but significant correlation between leniency and difficulty—the Spearman's coefficient was of 0.199 with $p < .05$.

The weak correlations observed between the computational metrics and the human-evaluated criteria of visual aesthetics and difficulty indicate that there is hope that future research will develop novel computational metrics to automatically (without asking the user) estimate the visual aesthetics and difficulty of IMB levels.

### 3.3.5 Strengths of the Computational Evaluation

Although our experiment showed that the computational metrics can be misleading, this kind of automatic evaluation also has its strengths. In contrast with the user study, one can easily achieve statistical significance by computing the metrics for a large number of levels. Moreover, we found the metrics to be very easy to implement. Taken together, these features make the computational metrics an easy and cheap way to perform an initial exploratory evaluation of the content generated by PCG systems.

The computational metrics can be particularly useful for gaining insight on evaluation criteria which are hard to test in user studies. For example, if one wants to verify the diversity of levels generated by a given PCG system in a user study, then the participants would have to play several levels generated by the same system and then inform the diversity of levels played. If the subjects had to play several levels of each system, then the experiment would likely be too long to be practical. One could then use the CD metric—or another similar metric such as edit distance (Smith, Treanor, et al., 2011)—to gain insight on the structural diversity of levels generated.

## 3.4 Conclusions

We tested several computational metrics used to evaluate levels generated by PCG systems for the game of IMB. We conducted a user study for evaluating four PCG systems according to the following criteria: enjoyment, visual aesthetics, and difficulty. Then, we compared the results obtained in the user study with those obtained by using computational metrics. Our evaluation showed that the computational metrics (i) are not able to accurately estimate enjoyment, (ii) provides limited information about the visual aesthetics of the levels, and (iii) can be misleading with respect to the player's perceived difficulty. Yet, the computational metrics can provide important information by measuring features which are hard to be measured in user studies, such as the evaluation of the diversity of levels generated by a PCG system.

We show that a well-designed user study cannot be replaced by the current computational metrics. Although we believe that the current computational metrics can help in the design process giving an initial idea about the quality of content in a cheap and easy way, one of the most important challenges for PCG researchers is to improve the current computational metrics as well as the creation of new metrics that reflect better the human's perspective.

# CHAPTER 4

# Visually Pleasing Video Game Maps

In this chapter we introduce the main proposal of this work, the computational model based on theories of graphical design to generate visually-pleasing video game maps. We cast the map generation problem as an optimization problem and present a simplified version of the problem that we use to prove the problem of generating symmetrical levels to be computationally hard. We then propose a heuristic search algorithm that is able to quickly solve the map generation problem. Namely, we implement a Branch and Bound search procedure (B&B) which guarantees to return an optimal solution to the problem. A user study using a clone of the Super Mario Bros (SMB) game shows the advantage of our method over other approaches in terms of visual aesthetics and enjoyment. Another user study shows that our method is able to generate maps with striking similarity to the original SMB maps produced by professional designers.

This chapter is organized as follows. First, in Section 4.1, we formulate the problem. Following, in Section 4.2, we describe the objective function used in the heuristic search algorithm proposed, in Section 4.3 we describe the level generation as an optimization problem, in Section 4.4 we present search procedures that are able to find optimal solutions, in Section 4.5 we compare the number of nodes expanded among the search procedures used, in Section 4.6 we explain how we generate a complete IMB level, in Section 4.7 we evaluate with a user study the *symmetry* system introduced in this work and compare it to other existing approaches for generating IMB levels, and finally in Section 4.8 we show the another user study where we compare the visual aesthetics of level chunks generated by our system with those created by professional designers.

## 4.1 Problem Formulation

The *level generation problem* is similar to the problem formulation described in Section 3.1, however in his chapter we introduced some new terms. We divided the problem into two problems. First one has to choose a set of objects $\mathbf{G} = \{o_1, o_2, \cdots, o_m\}$ that will compose the game level. Then, one has to define the objects' $x$ and $y$ coordinates in a grid space of size $L \times k \cdot W$, i.e., $L$ rows and $k \cdot W$ columns. In this work we assume that $\mathbf{G}$ is provided as input by the user or some other system, and our algorithm sets the $x$ and $y$ coordinates of the objects in the grid aiming at generating visually pleasing levels.

Some objects composing a level can be placed on "top of each other" by occupying the same cells in the grid; we call these objects *shareable*. Other objects must not occupy a cell that is already occupied by another object. For example, in the game of *Super Mario Bros.* multiple mountains may occupy the same cell, while blocks must not. We assume that there is enough space in the grid to place all $m$ objects.

In this work we transform the problem of generating a level of size $L \times k \cdot W$ into the problem of generating $k$ smaller levels of size $L \times W$. Then, the $k$ smaller levels are concatenated into a level of size $L \times k \cdot W$ with a method that is similar to the one introduced by (Reis et al., 2015) (described in Section 4.6).

## 4.2 Symmetry as Objective Function

In this section we describe the objective function we optimize while placing objects on the grid of size $L \times W$. The level is divided into four regions of equal dimensions by a vertical and a horizontal lines, as illustrated by the dashed lines in Figure 4.1. The vertical and horizontal line mark the boundaries of the regions, which are named Upper Left (UL), Upper Right (UR), Lower Left (LL), and Lower Right (LR), as shown in Figure 4.1. Figure 4.1 also shows the placement of four objects (gray rectangles), one in each of the regions. Intuitively, the object placement shown in Figure 4.1 results in a perfectly symmetrical image: there are identical objects on both sides of the vertical dashed line, as well as on both sides of the horizontal dashed line. We define next an objective function that captures this intuitive notion of symmetry.

We define as $G_{UL}$, $G_{UR}$, $G_{LL}$, and $G_{LR}$ the set of objects in regions $UL$, $UR$, $LL$, and $LR$, respectively. Note that an object $o$ can be placed simultaneously in more than one region. For example, half of a rectangle $o$ could fall in $LL$ and the other half in $LR$. If this happens, we replace $o$ by two objects $o_1$ and $o_2$, where

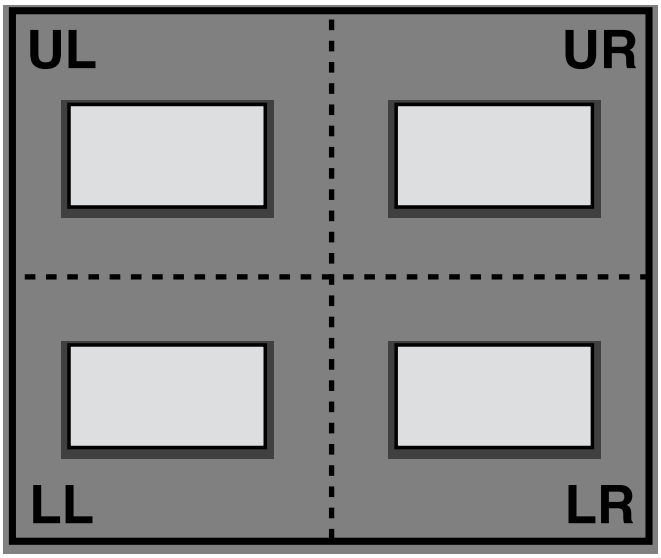



Figure 4.1: An illustrative example of a perfectly symmetrical placement of objects

$o_1$ is in $LL$ and $o_2$ in $LR$. Following this approach, the sets of objects in different regions are disjoint.

We define a symmetry value of a level in terms of function $V(\cdot)$, which is defined for the objects in region $LL$ as follows.

$$V(G_{LL}) = \sum_{o \in G_{LL}} dx(o) + dy(o) + A(o)\,, \tag{4.1}$$

where $dx(o)$ and $dy(o)$ are the distances between the center of the rectangle-shaped object $o$ and the vertical line and the horizontal line, respectively; $A(o)$ is the area of object $o$. $V(G_{LR}), V(G_{UL}), V(G_{UR})$ are defined analogously. The symmetry value of a level is defined as the sum of the absolute differences of $V$-values of different grid regions as shown next.

### 4.2.1 The Vertical Symmetry Function

One formulation of symmetry we use in our experiments is defined as follows.

$$S_{vert}(\mathbf{G}) = |V(G_{UL}) - V(G_{UR})| \tag{4.2}$$

$$+ |V(G_{LL}) - V(G_{LR})|\,. \tag{4.3}$$

$S_{vert}(G)$ accounts for the symmetries defined by the vertical line, i.e., the absolute differences between the upper left and the upper right regions (term 4.2 above), as

well as the differences between the lower left with the lower right regions (term 4.3 above).

### 4.2.2 The All Symmetry Function

Another objective function we consider is the following.

$$
\begin{aligned}
S_{all}(\mathbf{G}) = &|V(G_{UL}) - V(G_{UR})| && (4.4)\\
&+ |V(G_{LL}) - V(G_{LR})| && (4.5)\\
&+ |V(G_{UL}) - V(G_{LL})| && (4.6)\\
&+ |V(G_{UR}) - V(G_{LR})| && (4.7)\\
&+ |V(G_{UL}) - V(G_{LR})| && (4.8)\\
&+ |V(G_{UR}) - V(G_{LL})|\,. && (4.9)
\end{aligned}
$$

Terms 4.4 and 4.5 of $S_{all}$ account for the symmetries across the vertical line, terms 4.6 and 4.7 for the symmetries across the horizontal line, and terms 4.8 and 4.9 for the symmetries across both the horizontal and vertical lines.

## 4.3 Level Generation as an Optimization Problem

In the *symmetry problem* one receives as input two sets of objects, $G$ and $G'$, such that $\mathbf{G} = G \cup G'$, altogether with the information of which objects are shareable. The objects in $G'$ are already placed in the grid, i.e., their $x$ and $y$ coordinates are defined as part of the input and they must not be changed. The integers $L$ and $W$ defining the size of the grid are also part of the input. The height and width of the objects in $G$ and $G'$ are no larger than $L$ and $W$, respectively. The task is to place the objects in $G$ in the grid while minimizing $S(\mathbf{G})$, where $S$ could be $S_{vert}$ or $S_{all}$.

We consider the set $G'$ of objects already placed in the grid because we envision an algorithm that solves the symmetry problem being used to assist a designer in the process of creating levels of a platform game, where the designer places a few objects in the grid and the algorithm finishes the work by placing the remaining objects.

### 4.3.1 The Simplified Symmetry Problem

We also consider a simplified version of the symmetry problem that we use to prove the problem of generating symmetrical levels to be computationally hard. In the simplified version of the problem the grid has size $1 \times W$, i.e., one row and $W$ columns. Since there is just a single row in the grid, in this simplified version of the problem there are only two regions of equal area: left (L) and right (R), with $G_L$ and $G_R$ being the set of objects placed in L and R, respectively. The objective function of this simplified version considers only the symmetry with respect to the the area $A(o)$ of objects $o$ in L and R, and is defined as follows,

$$S_{sim}(\mathbf{G}) = \left| V'(G_L) - V'(G_R) \right| .$$

Here, $V'(O) = \sum_{o \in O} A(o)$ for any set $O$ of objects.

**Proposition 1.** *The simplified symmetry problem is NP-hard.*

*Proof.* We make a reduction from the NP-complete Number Partition Problem (NPP) (Garey & Johnson, 1979), where one is given a finite set $U$ of positive integers and has to find a subset $C \subseteq U$ such that $|\sum_{u \in C} u - \sum_{u \in U - C} u|$ is minimized.

In our polynomial-time reduction each element $u$ in $U$ is treated as a shareable object $o$ in $G$ with dimensions $1 \times u$, which yields $A(o) = u$. The grid has 1 row and $2 \cdot (\max_u + 1)$ columns, where $\max_u$ is the largest $u$ amongst all $u$ in $G$. The set of objects already placed, $G'$, contains two non-shareable objects of size $1 \times 1$. One of the objects in $G'$ is placed on the grid cell immediately to the left of the vertical line separating regions L and R, and the other on the grid cell immediately to right of the vertical line.

Due to the two non-shareable objects on the boundary of L and R, no objects in $G$ can be placed simultaneously in L and R. The non-shareable objects occupy one cell each, leaving $\max_{s(u)}$ grid cells for each region, which allows any object in $G$ to be placed in either L or R. A solution of this simplified symmetry problem represents a solution to the NPP. □

We conjecture that the regular symmetry problem with either $S_{vert}$ or $S_{all}$ objective function is not computationally easier than the simplified symmetry problem.

## 4.4 Symmetry Optimization as Search

In this section we present search procedures that are able to find optimal solutions of instances of the symmetry optimization problem. First, we define the optimization problem as a state-space search problem. Then, we present a brute-force search algorithm for solving the problem. Following, we introduce a branch-and-bound search procedure (B&B) that uses a heuristic function to prune unpromising nodes in the search tree while still finding optimal solutions.

### 4.4.1 Brute-Force Search

The search tree of the symmetry problem is defined as follows. In each level of the search tree we assign values to the $x$ and $y$ coordinates of one of the given objects in $G$. The root of the search tree is an empty partial solution to the problem, i.e., no objects have their coordinate values assigned yet. Each child of the root accounts for one possible position on the grid for a given object in $G$. Each leaf node of this search tree represents a solution to the symmetry problem. We are interested in finding the optimal solution, i.e., the leaf node with lowest symmetry value.

Since in every level we assign the coordinate values to an object in $G$, the height of the search tree equals the cardinality of $G$. The branching factor varies within the search tree because we consider a set of constraints while assigning values to the objects' coordinates. For example, an object cannot occupy a cell that is occupied by a non-shareable object. Intuitively, a node in the search tree representing a partial solution that contains many non-shareable objects will have a smaller branching factor than a partial solution with fewer non-shareable objects. This is because the non-shareable objects reduce the number of positions where the remaining objects can be placed. We also consider a set of domain-specific constraints. For example, in IMB, objects such as pipes and cannons have to be placed on the ground.

A brute-force search (BFS) algorithm traverses the search tree in a depth-first manner and returns a solution with lowest symmetry value. Next, we present a B&B procedure that also finds optimal solutions to the symmetry problem while possibly expanding many fewer nodes than BFS.

### 4.4.2 Branch and Bound Search Procedure

The B&B procedure we introduce in this work uses a *heuristic function* $H(n)$ to guide its search, where $n$ is a node in the search tree. A heuristic

**Algorithm 1** Branch-and-Bound

**Require:** root node $n$, objective function $S$, and heuristic function $H$.
**Ensure:** optimal solution $n^*$.
1: $B \leftarrow \infty$, $n^* \leftarrow \emptyset$ //$B$ and $n^*$ are global variables
2: return Depth-First-Search$(n, S, h)$ //see Algorithm 2

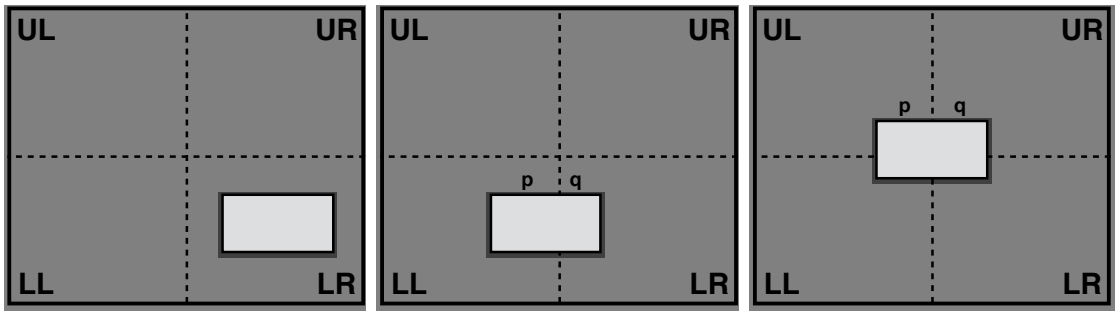
**Algorithm 2** Depth-First-Search

**Require:** root node $n$, objective function $S$, and heuristic function $H$.
**Ensure:** optimal solution $n^*$.
1: **for** each child $n'$ of $n$ **do**
2:  **if** $H(n') < B$ **then**
3:  **if** $n'$ is a leaf node and $S(n') < B$ **then**
4:  $B \leftarrow S(n')$ //update bound
5:  $n^* \leftarrow n'$ //update solution
6:  **else**
7:  Depth-First-Search$(n', S, h)$ //recursive call
8: return $n^*$

function $H(n)$ provides an estimate of the symmetry value of a complete solution obtained from partial solution $n$. $H$ is called *admissible* if it never overestimates the lowest symmetry value encountered amongst all leaf nodes in the subtree rooted at node $n$, for all $n$. Let $B$ be the symmetry value of the best incumbent solution encountered by B&B. When using an admissible heuristic function $H$, if $H(n) \geq B$, then node $n$ can be safely pruned as solutions reachable from $n$ are no better than $B$. We describe the heuristic we use in our experiments in Section 4.4.3 below.

Algorithm 1 and 2 show the pseudocode of a recursive implementation of B&B. B&B uses two global variables, $B$ and $n^*$, for storing the symmetry value of the best solution and the best solution encountered thus far, respectively. $B$ is initialized with infinity and $n^*$ is initially empty (see line 1 of Algorithm 1). Then, B&B performs a depth-first search. During its search, B&B updates the values of $B$ and $n^*$ (lines 4 and 5 of Algorithm 2) if it encounters a complete solution $n'$ with symmetry value smaller than the symmetry value of $n^*$ (line 3 of Algorithm 2). Node $n'$ is only expanded if the lower bound of its symmetry value, $H(n')$, is strictly lower than $B$.

The B&B procedure presented above is guaranteed to return an optimal solution to the symmetry problem given that $H$ is a lower bound on the symmetry value of a partial solution $n$.




(a) Object spanning a single region.

(b) Object spanning two regions.

(c) Object spanning four regions.

Figure 4.2: Different placements of a rectangle-shaped object show that it can occupy one region, two regions, or four regions.

### 4.4.3 Heuristic Function

In this section we describe the heuristic function $H(n)$ that returns a lower bound on the lowest symmetry value amongst all leaf nodes in the subtree rooted at node $n$. We consider the objective function $S_{all}$ in this section.

For node $n$, we denote as $P_n \subseteq G$ the set of objects already placed on the grid (objects that had their $x$ and $y$ coordinates assigned by the search procedure). The set of objects yet to be placed is defined as $M_n = G - P_n$. A lower bound on the symmetry value of $n$ is computed as follows.

$$H(n) = S_{all}(P_n \cup G') - 3 \times \left( \sum_{o \in M_n} A(o) + \frac{W}{2} + \frac{L}{2} \right), \tag{4.10}$$

where $A(o)$ is the area of the rectangle-shaped objects $o$, $\frac{W}{2}$ is the largest possible distance from the center of an object to the vertical line, and $\frac{L}{2}$ is the largest possible distance from the center of an object to the horizontal line. The term $S_{all}(P_n \cup G')$ in Equation 4.10 accounts for the symmetry value of the objects already placed on the grid. $H(n)$ considers that the remaining objects $M_n$ can be placed in a way that they only decrease the symmetry value of the partial solution $n$.

The values of $H$ according to Equation 4.10 could be negative, while the value of $S_{all}(\mathbf{G})$ is always non-negative. Thus, the value of $H(n)$ is trivially set to zero if it becomes negative.

The following theorem states that $H$ is admissible.

**Theorem 1.** *Let a symmetry problem be defined with a grid of size $L \times W$, two sets of objects $G$ and $G'$, where the height and the width of all $o \in G \cup G'$ are at most $W$ and $L$, respectively. Also, let $n$ be a partial solution to the symmetry problem where the set of objects $P_n \cup G'$, with $P_n \subseteq G$, are already placed on the*

*grid. There is no placement of the remaining objects $M_n = G - P_n$ that yields $S_{all}(G \cup G') < H(n)$.*

*Proof.* We start by proving that the placement of object $o$ can reduce the symmetry value $S_{all}(P_n \cup G')$, in terms of area, by at most $3 \times A(o)$. Then we prove that an object can reduce the value of $S_{all}(P_n \cup G')$ by at most $3 \times \frac{W}{2}$ and $3 \times \frac{L}{2}$, in terms of distance to the vertical and horizontal lines, respectively.

A rectangle-shaped object can be placed completely inside one of the four regions, partially in two regions, or partially in all four regions, as illustrated by Figures 4.2a, 4.2b, and 4.2c, respectively. If $o$ is placed partially in more than one region, then we treat $o$ as though it was replaced by multiple objects, one for each region it occupies. For example, if $o$ occupies two regions, then we replace $o$ by objects $o_1$ and $o_2$ with $A(o) = A(o_1) + A(o_2)$. Analogously, if $o$ occupies all four regions, then we replace $o$ by four objects whose areas sum up to $A(o)$. The set of objects in a given region of the grid is accounted for in three terms of Equations 4.4–4.9 describing $S_{all}(G)$ (e.g., $G_{UL}$ appears in terms 4.4, 4.6, and 4.8). Thus, when an object $o$ is placed on the grid, its area $A(o)$ can reduce the symmetry value by at most $3 \times A(o)$. This is true even if $o$ is placed in two or four regions, since $o$ is replaced by objects with total area equal to $A(o)$.

Since the grid is $W$ wide, each region is $\frac{W}{2}$ wide. Any object $o$ placed in one of the regions, as illustrated in Figure 4.2a, reduces $S_{all}$ by at most $3 \times \frac{W}{2}$ in terms of the distance of $o$'s center to the vertical line. This is because the center of an object cannot be placed farther than $\frac{W}{2}$ from the vertical line, and $dx(o)$ is accounted for three times in $S_{all}$.

It can also be that $o$ is placed in two regions, as illustrated in Figure 4.2b. Let $o$ have width of $D = p + q$. Moreover, since $o$ is placed in two regions, $o$ is replaced by two objects, one with width $p$ and another with width $q$. Since the center point of the objects have distance $\frac{p}{2}$ and $\frac{q}{2}$ from the vertical line, the two objects can reduce at most $3 \cdot \frac{p}{2} + 3 \cdot \frac{q}{2} = 3 \cdot \frac{D}{2}$ from $S_{all}$. Since $D \leq W$, an object placed in two regions can decrease at most $3 \times \frac{W}{2}$ from $S_{all}$. The object $o$ may also occupy all four regions, as illustrated in Figure 4.2c. In this case $o$ is replaced by objects $o_1 \in UL$, $o_2 \in UR$, $o_3 \in LL$, and $o_4 \in LR$. Since $o$ is rectangle-shaped, $o_1$ and $o_3$ have width $p$ and $o_2$ and $o_4$ have width $q$. Note that $dx(o_1)$ cancels out with $dx(o_3)$ and $dx(o_2)$ cancels out with $dx(o_4)$ in $S_{all}$. Thus, considering only $dx$-values, $S_{all}$ can be written considering the four non-cancelled terms as $4 \cdot \frac{(|p-q|)}{2} = 2 \cdot |p - q|$, which is maximized when $p = \frac{W}{2}$ and $q = 0$. Thus, the value of $3 \times \frac{W}{2}$ is also an upper bound on how much $S_{all}$ can be reduced in terms of $dx$ by placing $o$.

The proof for the term $3 \times \frac{L}{2}$ is analogous to the proof just shown for

$3 \times \frac{W}{2}$. □

According to Theorem 1, B&B employing $H$ is guaranteed to find a solution with optimal symmetry value, possibly expanding many fewer nodes than BFS.

### 4.4.4 Object Ordering

The B&B procedure described above traverses the search tree while using $H$ to prune unpromising nodes. As explained, B&B prunes all nodes $n$ for which $H(n) \geq B$. B&B will tend to prune more nodes if $H$ provides values that are closer to a perfect estimate (i.e., larger values if $H$ is admissible). We now describe how the order in which the objects are placed on the grid might affect the accuracy of the $H$-values.

The negative (second) term of $H(n)$ (see Equation 4.10) accounts for the area of the objects yet to be placed, $M_n$. Thus, if B&B places the objects with larger $A$-values earlier during search, it reduces the negative term of $H$ thus increasing the heuristic value of nodes at deeper levels of the tree. Aiming at obtaining more accurate heuristic estimates, in our implementation of B&B we sort the objects in $G$ according to their areas and place larger objects before smaller ones.

### 4.4.5 Region Ordering

Another factor that affects how much pruning B&B is able to perform is the cost of the incumbent solution $B$. Intuitively, B&B is able to prune larger portions of the search tree if it quickly finds an optimal or near-optimal solution to the symmetry problem. This is because smaller $B$-values allow more nodes $n$ to be pruned by satisfying the $H(n) \geq B$ condition. In addition to defining an ordering of objects to improve the estimates returned by $H$, as explained in Section 4.4.4 above, we define an ordering of regions to allow B&B to find optimal or near-optimal solutions more quickly.

Once the search procedure defines which object $o$ is going to be placed next during search, B&B verifies which region $R \in \{UL, UR, LL, LR\}$ has the smallest value of $V$. B&B will try all possible positions of $o$ within $R$ before trying to place $o$ in another region. The intuition is that B&B tries to place the objects aiming at keeping all regions balanced, with similar $V$ values. Then, the $V$-values cancel out and B&B is able to find good solutions early in search.

Table 4.1: Average number of nodes expanded and average running time in seconds of BFS and B&B for the PCG type

| | PCG Type | | | | | |
|---|---|---|---|---|---|---|
| $\|G\|$ | BFS | | B&B | | Ratio | |
| | Nodes | Time | Nodes | Time | Nodes | Time |
| 4 | 8,660.0 | 0.04 | 3,777.6 | 0.03 | 2.29 | 1.47 |
| 5 | 57,848.4 | 0.24 | 18,705.5 | 0.12 | 3.09 | 1.92 |
| 6 | 230,576.7 | 1.25 | 71,317.7 | 0.51 | 3.23 | 2.44 |
| 7 | 197,811.1 | 1.34 | 68,170.0 | 0.65 | 2.90 | 2.08 |
| 8 | 2,181,380.6 | 17.81 | 780,964.5 | 8.80 | 2.79 | 2.02 |
| 9 | 3,863,823.9 | 41.48 | 1,947,896.5 | 24.60 | 1.98 | 1.69 |

## 4.5 Node Expansion and Running Time Experiments

In this section we compare the number of nodes expanded by BFS and B&B, as well as their running times.

### 4.5.1 Experimental Setup

In this experiment we use sets $G$ of sizes in $\{4, 5, 6, 7, 8, 9\}$ and a grid of size $20 \times 15$ which is initially empty (i.e., $G' = \{\}$). We choose these sizes of $G$ because levels of Super Mario Bros. often have 4 to 9 objects distributed in chunks of size equivalent to the $15 \times 20$ grid used in this experiment.

We consider two types of sets of objects $G$. In the first type we allow shareable and non-shareable objects of varied dimensions. We employ Notch's Level Generator system (NLG), to define different $G$ sets. That is, we execute NLG to generate a level $e$ of size $15 \times 20$ and use the objects in $e$ to create the set $G$ of an instance of the symmetry problem. This way we use sets of objects whose dimensions and properties are similar to the objects that appear in actual maps of a game. We call this type of set the *PCG type*.

In the second type of sets $G$ we use a collection of non-shareable objects of size $1 \times r$, where $r$ is a random integer ranging from 1 to 4. We chose to experiment with sets $G$ of this type because they emulate the scenario where one applies our method to automatically create symmetrical graphical user interfaces (GUIs) . That is, one is given a set of buttons another GUI elements to be placed on a software window. The small objects of varying size used in this experiment emulate several GUI elements. We call this type of set the *GUI type*.

Table 4.2: Average number of nodes expanded and average running time in seconds of BFS and B&B for the GUI type

| GUI Type | | | | | | |
|---|---|---|---|---|---|---|
| $\|G\|$ | BFS | | B&B | | Ratio | |
| | Nodes | Time | Nodes | Time | Nodes | Time |
| 4 | 233,512.7 | 1.19 | 16,832.1 | 0.21 | 13.87 | 5.53 |
| 5 | 4,813,781.2 | 12.14 | 829,463.5 | 3.13 | 5.80 | 3.88 |
| 6 | 69,871,632.3 | 188.49 | 15,463,431.4 | 63.15 | 4.52 | 2.98 |
| 7 | 1,147,264,687.0 | 3,626.42 | 285,681,694.7 | 1,350.54 | 4.02 | 2.69 |
| 8 | 7,049,476,561.0 | 26,785.45 | 2,474,114,680.9 | 13,320.39 | 2.85 | 2.01 |
| 9 | 49,489,256,774.6 | 233,623.03 | 19,493,911,071.8 | 127,756.31 | 2.54 | 1.83 |

We perform this experiment with 20 problem instances of each type of sets (PCG and GUI) and present the average number of nodes expanded and average running time in seconds for different cardinalities of $G$. All experiments are run on 2.67 GHz machines. The results are presented in Tables 4.1 and 4.2, for the PCG and GUI type, respectively. The tables also show the ratio between BFS's average number of nodes expanded to B&B's, as well as BFS's average running time to B&B's. Ratio values greater than 1.0 indicate that BFS expands more nodes and is slower to retrieve an optimal solution. For example, the ratio value of 2.29 for the number of nodes expanded for $|G| = 4$ in Table 4.1 indicates that BFS expands 2.29 times more nodes than B&B on average.

### 4.5.2 Discussion of Results

The problem instances tend to be harder for larger values of $|G|$—this trend can be observed in both Table 4.1 and 4.2. The results presented in Table 4.1 show that BFS expands 2 to 3 times more nodes than B&B, and is 1.5 to 2 times slower than B&B. The reduction in the number of nodes expanded does not translate directly to the same reduction in running time because the heuristic function represents an overhead to the B&B approach. That is, B&B expands fewer nodes but has a higher per-node cost. Nevertheless, the B&B approach is substantially faster than BFS, showing that the overhead of computing $H$ pays off.

The speedup of B&B over BFS is larger for the GUI type (see Table 4.2). For example, for $|G|$ of 4, B&B expands almost 14 times fewer nodes than BFS, and is almost 5 times faster than BFS. In general, the problem instances of the GUI type are harder than those of the PCG type. This happens because the

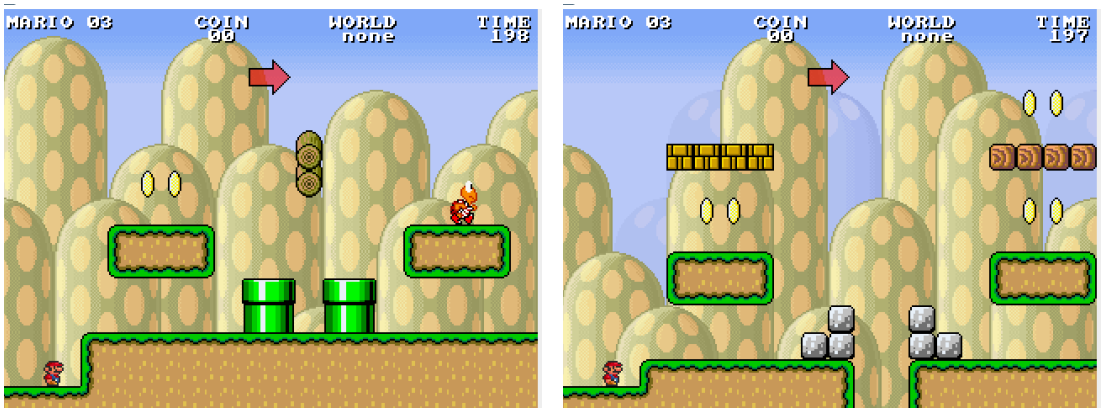



Figure 4.3: Levels generated by our system using the vertical symmetry function.

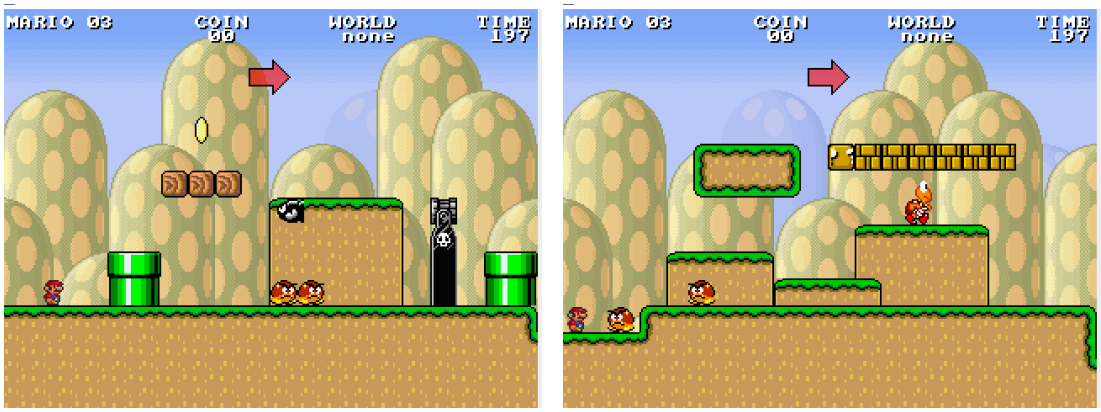



Figure 4.4: Levels generated by our system using the all symmetry function.

objects of the PCG type have constraints associated to them. For example, a pipe object in IMB can be placed in a very limited number of grid cells, as pipes must not be aloft. By contrast, the objects in the GUI type might be placed in any empty grid cell, which increases substantially the problem's state space. Although we frame the GUI type as a problem of designing a GUI, levels of platform games may contain objects with properties similar to the objects of the GUI type. For example, chunks of IMB levels could contain only small blocks that can be placed virtually anywhere on the grid.

## 4.6 Generating Larger Levels

The search algorithm described above can be used to generate small chunks of symmetrical game levels. In this section we explain how we combine these small levels of size $L \times W$ into a level of size $L \times k \cdot W$. We use an approach that is similar to the one introduced by (Reis et al., 2015).

Reis at al. describe a system that uses human computation to evaluate the difficulty, visual aesthetics, and enjoyment of a set of small chunks of a plat-

form game level. We refer to these small chunks of levels as *small levels*. Their method constructs a larger level of the platform game by concatenating small levels together according to a mathematical function they called *tension arc*. A tension arc receives as input the position $j$ of the small level within the larger level and returns the difficulty value of the small level occupying the $j$-th position. For example, for $j = 1$ (i.e., the first small level composing the large level) the tension arc used by Reis at al. returns a small value $d$, indicating that the first small level composing the large level must be an easy one. Then, their system randomly selects a level from their library of small level that has a difficulty of $d$ and was rated highly with respect to enjoyment and visual aesthetics by the human workers. The idea behind Reis at al.'s tension arc is to have the difficulty of the level increasing as the player advances into the level.

In this work we use Reis et al.'s tension arc-based system to build a larger level out of the small symmetrical levels generated by our B&B procedure. However, in contrast with Reis et al.'s work, our library of small levels is not labelled by human workers. Instead of using human computation, we trust our B&B procedure to generate visually pleasing small levels, and we use the metric of *leniency* (Smith & Whitehead, 2010) to replace Reis at al.'s human-computed difficulty values. Leniency measures how much challenge the player is likely face while playing the level. The leniency of level $\mathbf{L}$ is the sum of the lenience value $w(o)$ of all objects $o$ in $\mathbf{L}$, defined as $\sum_{o \in \mathbf{L}} w(o)$.

We use the $w$-values specified by (Shaker, Nicolau, et al., 2012). Namely, items that make the game easier such as power-up items have a weight of 1, some of the enemies of $-0.5$, and more challenging enemies of $-1$. Although the metric was originally defined for IMB, it can be translated to other platform games.

Our method generates an IMB level of size $L \times k \cdot W$ as follows. First we use the B&B method described above to generate a library $\mathbf{I}$ of symmetrical levels of size $L \times W$ with different sets $G$ of objects. As in our previous experiment, the sets $G$ are defined by the NLG system. Then, we compute the lenience of each level $i \in \mathbf{I}$ and normalize the resulting values to a number in $[1, 7]$, and round them up. After this process each level $i \in \mathbf{I}$ has a lenience integer value associated to it.

We have to concatenate $k$ small symmetrical levels to generate a level of width $k \cdot W$. The lenience of each of the $k$ levels composing the level of size $L \times k \cdot W$ is given by the tension arc function. In our experiments we use $k = 9$ and the tension arc shown in Table 4.3. For example, the fourth level of size $L \times W$ composing a level of size $L \times k \cdot W$ has to have a leniency of 4, while the last level has to have a leniency of 3. For each position $j \in [1, k]$ our method

Table 4.3: Tension arc used in our experiments.

| position | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
|---|---|---|---|---|---|---|---|---|---|
| lenience | 1 | 2 | 3 | 4 | 4 | 5 | 5 | 4 | 3 |

selects from **I** a level $i$ with the leniency value shown for entry $j$ in Table 4.3. In Figures 4.3 and 4.4 we show screenshots of levels generated by our system using the Vertical Symmetry function, and levels generated by our system using the All Symmetry function as evaluation function.

## 4.7 Comparison with Other Systems

In this section we evaluate with a user study the *symmetry* system introduced in this work and compare it to other existing approaches for generating IMB levels. The experiment presented is similar to the user study presented in Chapter 3, however there are some few differences.

We use three different IMB PCG systems in our experiments: *symmetry*, Human-Computation Tension Arc-Based (HCTA) with a parabolic tension arc, as described by citereis2015human, and Occupancy-Regulated Extension (ORE) generator (Mawhorter & Mateas, 2010). HCTA and ORE were used in the user study of Chapter 3.

We chose HCTA and ORE to compare against *symmetry* because the former was shown to outperform other approaches (Reis et al., 2015) and the latter was the winner of the 2011 Mario AI Competition.[1]

In addition to a Turing test, in the user study the systems are evaluated according to the following criteria: enjoyment, visual aesthetics, difficulty. Each participant was asked to answer how much they agreed or disagreed, in a 7-likert scale, with the following sentences: "This level is enjoyable to play"; "this level has good visual aesthetics"; "this level is difficult"; "this level was developed by a machine". A score of 1 for enjoyment and visual aesthetics means that the participant strongly agrees that the level played is enjoyable and has excellent visual aesthetics; a score of 1 for difficulty means that the participant strongly agrees that the level is difficult; a score of 1 for the Turing criterion means that the participant strongly agrees that the level was designed by a machine.

Subjects were instructed about the controls of the game before playing a practice level. The practice level is important so that the participants get

[1] see http://www.marioai.org/LevelGeneration.

acquainted with the controls of the game. NLG was used to generate the practice levels. Only after playing the practice level that the participants evaluated the levels generated by the PCG systems. Each participant played one level generated by each of the three PCG systems. After playing each level the participants gave scores according to the criteria described above in a 7-likert scale. In addition to the scores, the participants had the option to enter comments informing us of technical issues they might have had during the experiment. At the end of the experiment the subjects filled a questionnaire informing their age, gender, and if they had played SMB before.

Our within-subject experiment had 47 participants: 43 males and 4 females with an average age of 27.53 and standard deviation of 5.59. All participants had played SMB before. Each participant played one level generated by each system, resulting in the evaluation of 47 levels of each PCG system. The experiment was carried out online: our system was made available in the Internet and our experiment advertised in different mailing lists. Participation was anonymous.

Since all participants played one level generated by each system, we used a balanced Latin square design to counteract ordering effects. The tested levels were generated during the experiment by the evaluated systems, we did not pre-select a set of levels to be tested.

In order to have a fair comparison of the levels generated by different systems we had all systems generating levels of the same size: $160 \times 15$. We chose this size because we did not want the experiment to be too long. Finally, to ensure a fair comparison of the different approaches, we tuned the systems to generate levels with similar difficulty.

The data of participants who did not finish playing all levels is not included in the results. The number of 47 participants was obtained after cleaning the data.

### 4.7.1 Results

The mean results and standard deviations are shown in Table 4.4. Different letters in a given row indicate that the two means are significantly different. Shapiro-Wilk tests showed that our data is unlikely to be normally distributed ($p < .01$ for all criteria). Thus, we used the non-parametric Friedman test which showed a significant difference on enjoyment ($p < .01$), on visual aesthetics ($p < .01$), and on Turing ($p < .001$) across different systems; there was no significant difference for difficulty.

We then use Wilcoxon signed-rank tests to perform pairwise comparisons of the results obtained by the evaluated systems. We present the effect size of the comparisons ($r$-values) in addition to $p$-values. *Symmetry* generates levels that

| | SYM | HCTA | ORE |
|---|---|---|---|
| Enjoyment | $2.70 \pm 1.69^a$ | $2.97 \pm 1.68^a$ | $3.89 \pm 2.01^b$ |
| Aesthetics | $2.72 \pm 1.69^a$ | $2.78 \pm 1.72^a$ | $3.36 \pm 1.79^b$ |
| Difficulty | $3.74 \pm 1.85^a$ | $4.06 \pm 1.78^a$ | $3.29 \pm 2.12^a$ |
| Turing | $3.06 \pm 1.83^a$ | $3.95 \pm 2.12^b$ | $2.31 \pm 1.93^c$ |

Table 4.4: Lower values of enjoyment and aesthetics indicate levels which are more enjoyable to play and have better visual aesthetics; lower values of Difficulty indicate levels which participants found more challenging to play; larger values of Turing indicate levels that participants were more prone to believe that were generated by humans. Different letters in a given row indicate statistically significant results.

are significantly more enjoyable to play than the levels ORE generates ($p < .001$, $r = 0.50$). HCTA also generates levels that are significantly more enjoyable to play than those generated by ORE ($p < .01$, $r = 0.42$). There is no statistical difference between the levels generated by *symmetry* and HCTA with respect to enjoyment.

Pairwise comparisons on visual aesthetics also showed that *symmetry* generates levels with significantly better visual aesthetics than the levels ORE generates ($p < .05$, $r = 0.37$). Similarly, HCTA also generates levels with significantly better visual aesthetics than the levels ORE generates ($p < .01$, $r = 0.39$). There is no statistical difference between the levels generated by *symmetry* and HCTA with respect to visual aesthetics.

Pairwise comparisons on Turing showed that the levels HCTA generates trick more people into thinking they were generated by human designers than the levels generated by *symmetry* ($p < .05$, $r = 0.30$) and by ORE ($p < .0001$, $r = 0.56$). Also, the levels *symmetry* generates trick more people into thinking they were generated by human designers than the levels generated by ORE ($p < .05$, $r = 0.31$).

All pairwise comparisons reported as statistically significant have effect sizes equal or higher than the medium size mark of 0.3, indicating substantial differences among the systems. Some of the pairwise comparisons have large effect sizes ($r$-values of 0.50 or higher), showing large differences among the systems.

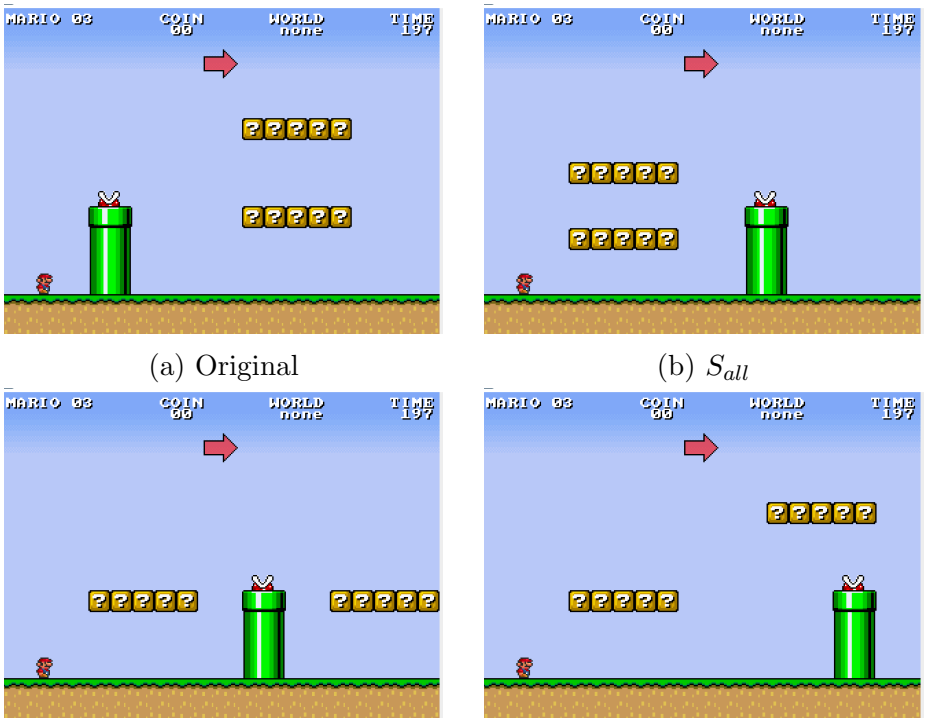
### 4.7.2 Discussion

*Symmetry* generates levels that are as enjoyable and as visually pleasing as those generated by HCTA. HCTA is a method that uses human workers to select the set of small levels composing the IMB level. *symmetry* does not use any sort of human input, it solely relies on the quality of the symmetrical levels

generated by our B&B approach.

The only criterion that *symmetry* and HCTA differ is Turing. This shows that although the levels generated by both systems are visually pleasing, the participants were able to notice more often that the levels generated by *symmetry* were actually generated by a computer program. We believe that *symmetry* could achieve results similar to HCTA on the Turing criterion if we introduce a symmetry error factor $\epsilon$. That is, instead of having B&B return a level with the smallest symmetry value, we could have it return a level with symmetry value within a factor of $\epsilon$ of the optimal value. This is an interesting future research direction.

*symmetry* generates levels that are significantly and substantially more enjoyable and more visually pleasing than those generated by ORE, another system that does not account for human input while generating IMB levels.




(a) Original (b) $S_{all}$

(c) $S_{vert}$ (d) Random

Figure 4.5: Small levels created by a human designer, by our approach using $S_{all}$ and $S_{vert}$, and by a random placement that respects domain-specific constraints.

## 4.8 Comparison with Human Designers

In this section we compare the visual aesthetics of level chunks generated by our system with those created by professional designers for fixed sets of objects $\mathbf{G}$.

We reproduced in IMB chunks of size $15 \times 20$ of original SMB levels. In total we reproduced 10 of such small levels. This set of small levels were selected arbitrarily from an online repository of game maps.[2] We collected the set of objects $\mathbf{G}$ in each small level reproduced from the original SMB levels and provided them as input to our B&B procedure so it could produce a level of equal size and with exactly the same set of objects as the original levels. We tested two objective functions in this experiment: $S_{all}$ and $S_{vert}$.

We ran a user study in which we presented the images of the small levels created by professional designers side by side with those generated by our B&B procedure while minimizing $S_{all}$ and $S_{vert}$. In addition to the small levels created by professional designers and those generated by our system, we also presented to the participants images of levels that were generated by randomly placing the objects in $\mathbf{G}$. In this random approach we used the same constraints used with out approach.

In contrast with the IMB game, the images of the levels were presented to the participants with a blue background. This is to ensure that the random background generated by the game would not bias the participant's perception of the object's placement. Figure 4.5 shows a representative set $\mathbf{G}$ out of the 10 sets used in our experiment—see the others 9 in appendix A.

Each participant visualized all four images at once: one generated by a professional designer (Original), two generated by our system ($S_{all}$ and $S_{vert}$), and one generated by the random placement (Random). The participants answered for each level shown on the screen if they agreed with the statement "This level has good visual aesthetics" in a 7-likert scale. After evaluating all 4 images, the participant could choose to either quit the experiment or to evaluate another set of images for another set $\mathbf{G}$ of objects.

Since each participant was allowed to interrupt the experiment at any time, they could evaluate from 1 to 10 sets $\mathbf{G}$ during their participation. In order to account for possible ordering effects, we randomized the order in which the sets $\mathbf{G}$ and the resulting levels of each approach were presented to each participant. Our experiment was performed online and advertised in different mailing lists and social network groups. At the end of the experiment the subjects filled a questionnaire informing their age, gender, and if they had played SMB before. Our within-subject experiment had 23 participants: 19 males and 4 females with an average age of 26.38 and standard deviation 3.81. The 23 participants evaluated 215 sets $\mathbf{G}$. They all had played SMB before.

[2] http://ian-albert.com/games

| $S_{all}$ | $S_{vert}$ | Human | Random |
|---|---|---|---|
| $3.26 \pm 1.74^a$ | $3.08 \pm 1.91^a$ | $3.08 \pm 1.85^a$ | $5.04 \pm 1.96^b$ |

Table 4.5: Visual aesthetics results. Lower values indicate levels that have better visual aesthetics. Different letters in a given row indicate statistically significant results.

### 4.8.1 Results

Table 4.5 shows the average score and standard deviation each approach obtained in our user study. Lower values mean that the participants tended to agree more strongly that the levels generated by a given approach have good visual aesthetics. Different letters indicate that the two means are significantly different. A Shapiro-Wilk test showed that our data is unlikely to be normally distributed ($p < .0001$). The non-parametric Friedman test showed a significant difference on the means ($p < .0001$). Pairwise comparisons with Wilcoxon signed-rank tests showed statistical difference between Random and all the other approaches ($p < .0001$ and $r > 0.50$ for all comparisons with Random). There was no statistical difference between the levels generated by our system and those created by professional designers.

### 4.8.2 Discussion

Our system generated levels with striking similarity with those created by human designers (see Figures 4.5a and 4.5b for a representative example). The results presented in Table 4.5 show that the participants thought that our system employing $S_{vert}$ generated levels as visually pleasing as those created by human designers. Moreover, the visual aesthetics of the levels generated by $S_{all}$ were rated only slightly worse than those generated by $S_{vert}$ and the professionals.

According to the results, the levels generated by our system and by the professionals have better visual aesthetics than those generated by the random placement of objects. These results support our hypothesis that our system is able to generate visually pleasing game maps.

# CHAPTER 5

# Conclusions

In this work we have introduced a computational model for generating visually pleasing solutions for the object placement problem. Our computational model optimizes objective functions to generate symmetrical scenarios. We showed that a simplified version of the problem of finding perfectly symmetrical scenarios to be computationally hard and introduced a heuristic search approach to solve it optimally.

The content generated by our algorithm was tested in two user studies. In the first user study the participants played IMB levels generated by our approach and by competing systems. The participants rated the levels generated by our system as enjoyable and as visually pleasing as the levels generated by a system that requires human input (HCTA), and more enjoyable and more visually pleasing than the levels generated by another system (ORE). In the second user study we compared the visual aesthetics of the IMB levels generated by our system with those created by professional designers. The participants rated the levels generated by our system as visually pleasing as the levels created by the professionals.

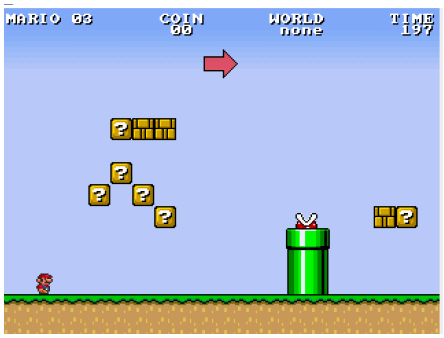
Although we applied our method to game design, we believe our computational model for visual aesthetics and our search algorithm are general and could be applied to other problems, such as the placement of buttons and icons in graphical user interfaces. The application of our method to other domains is an interesting direction of future work.

In this work we also tested several computational metrics used to evaluate levels generated by PCG systems for the game of IMB. We conducted a user study

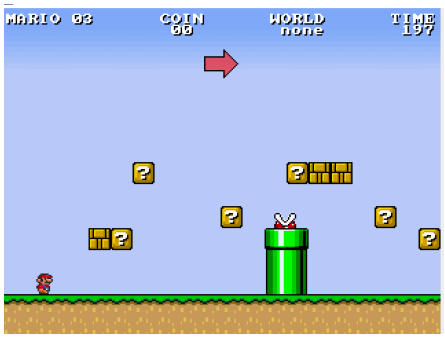
for evaluating four PCG systems according to the following criteria: enjoyment, visual aesthetics, and difficulty. The most important conclusion drawn from this experiment about metrics is that a well designed user study cannot be replaced by the current computational metrics. However, we believe that the computational metrics are suitable to be used during the design process, for quick and easy exploratory evaluations of the PCG system.

# APPENDIX A

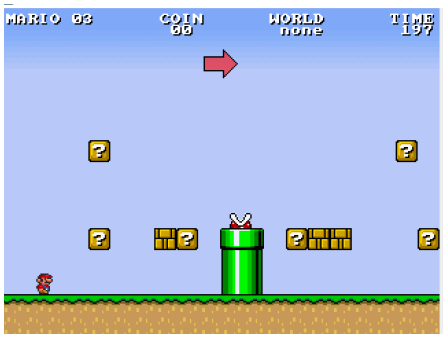
## APPENDIX: Set of images used in the experiment with Human designers



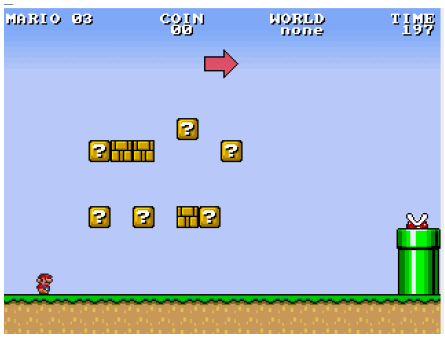

(a) Original



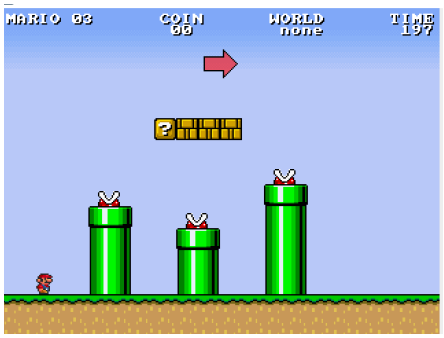

(b) $S_{all}$



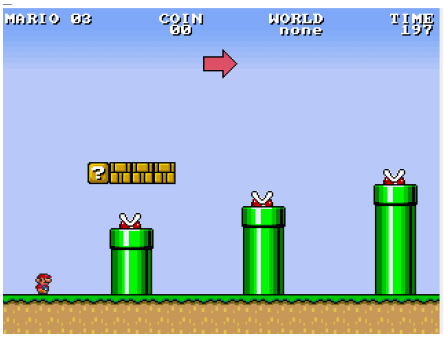

(c) $S_{vert}$



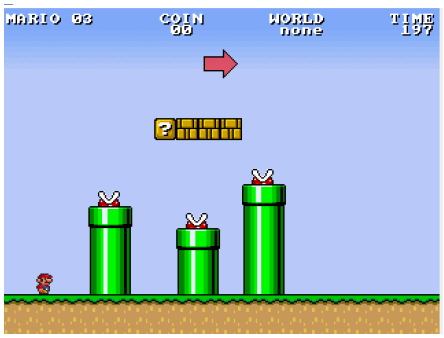

(d) Random



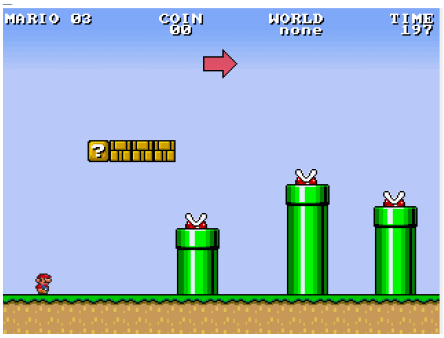

(a) Original



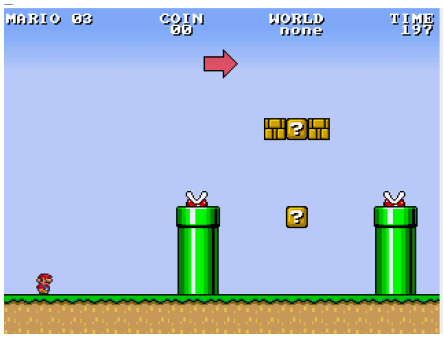

(b) $S_{all}$



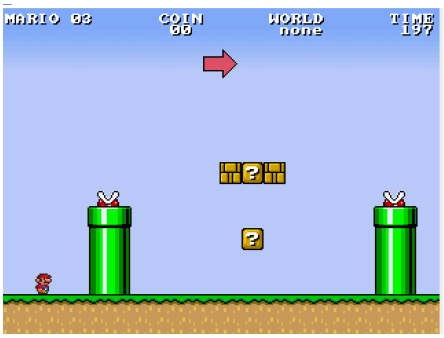

(c) $S_{vert}$



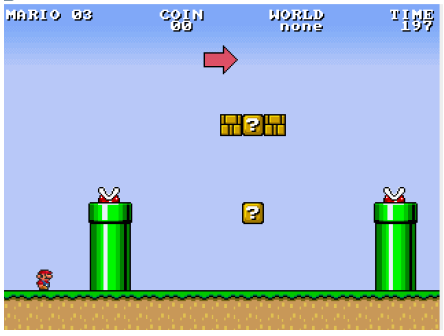

(d) Random



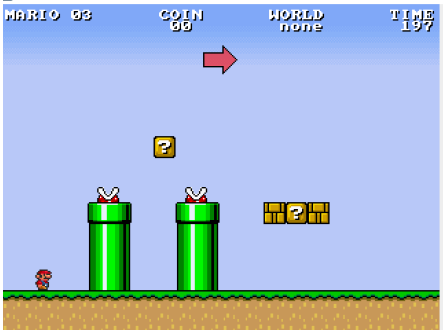

(a) Original



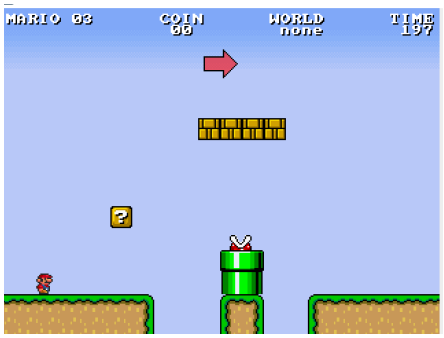

(b) $S_{all}$



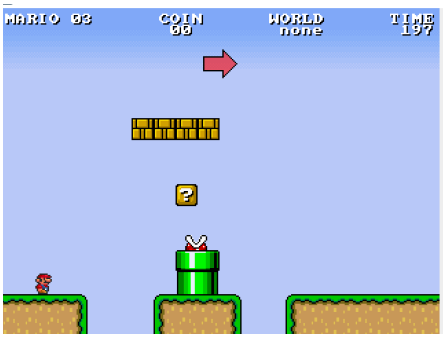

(c) $S_{vert}$



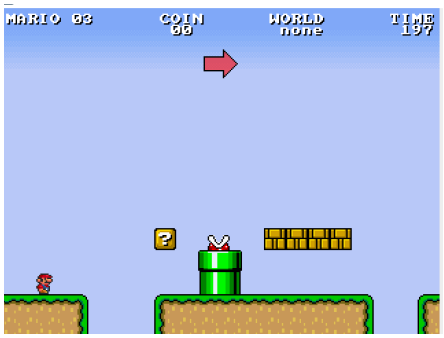

(d) Random



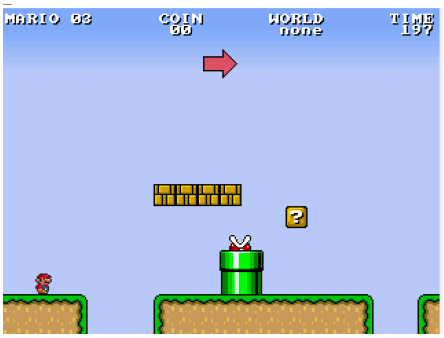

(a) Original



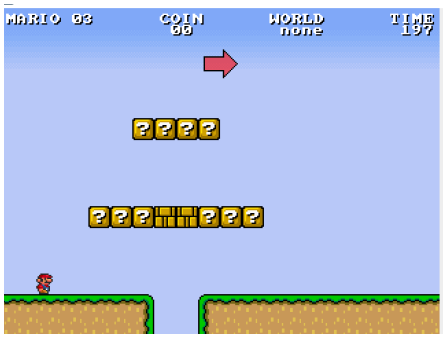

(b) $S_{all}$



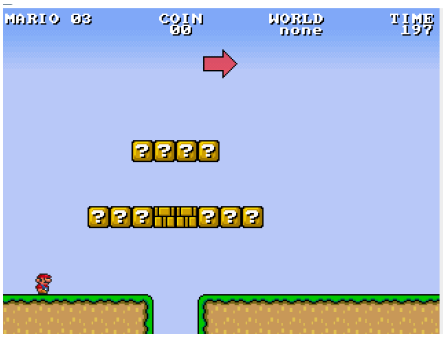

(c) $S_{vert}$



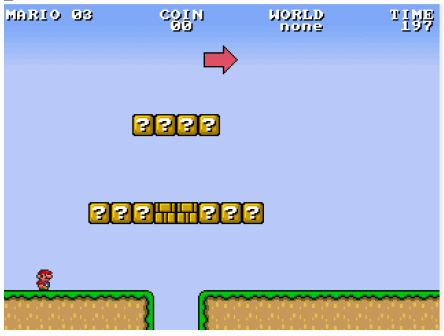

(d) Random



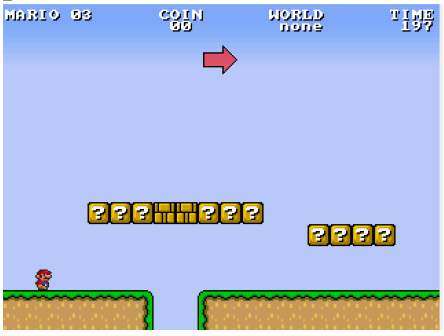

(a) Original



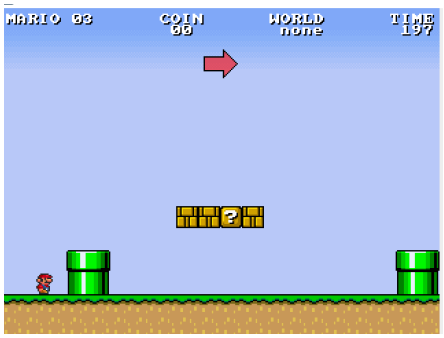

(b) $S_{all}$



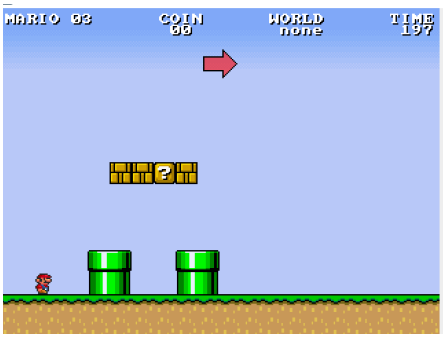

(c) $S_{vert}$



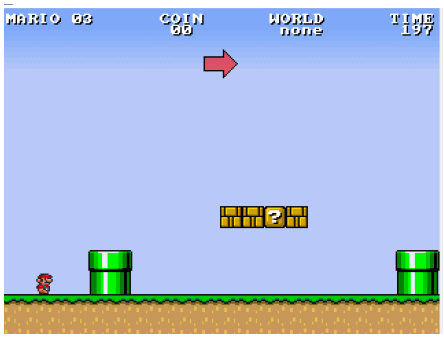

(d) Random



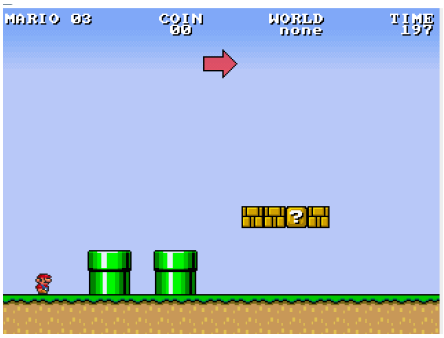

(a) Original



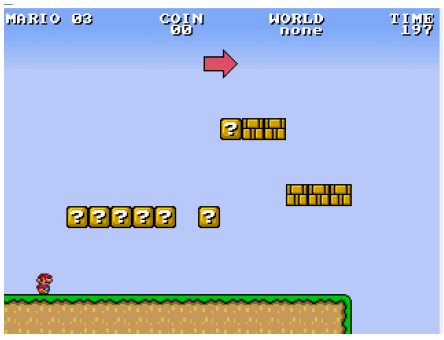

(b) $S_{all}$



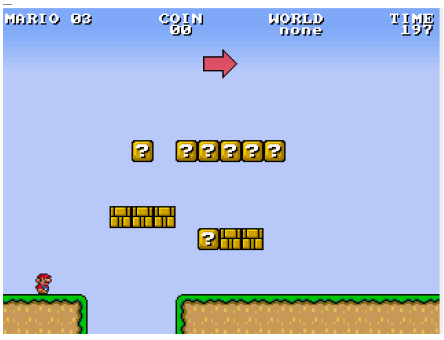

(c) $S_{vert}$



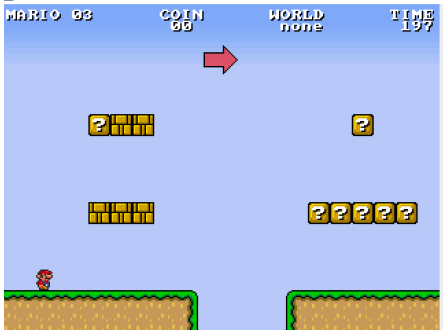

(d) Random



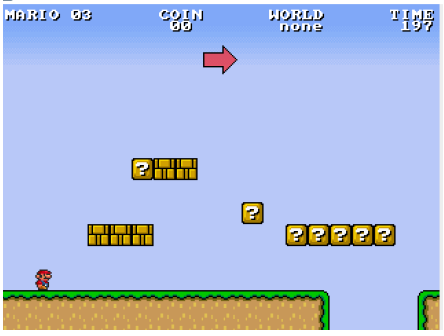

(a) Original



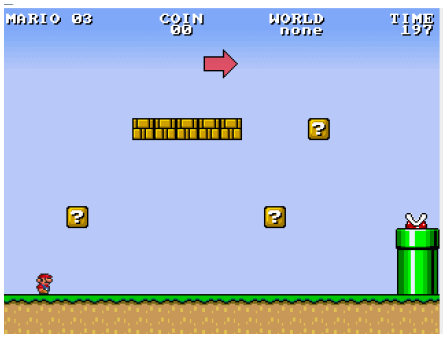

(b) $S_{all}$



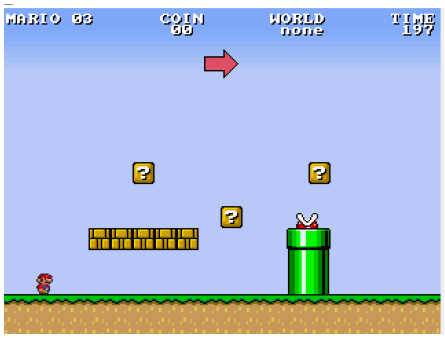

(c) $S_{vert}$



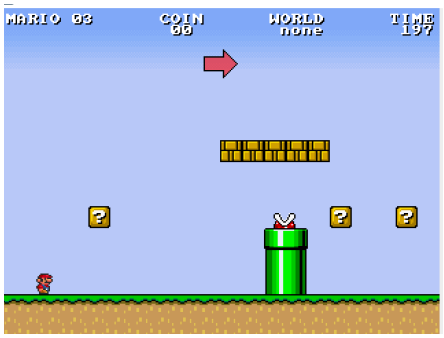

(d) Random



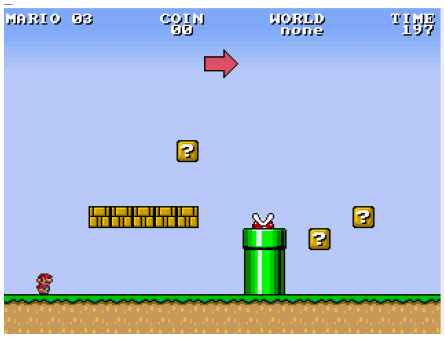

(a) Original



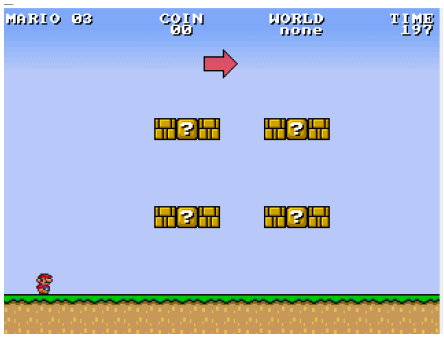

(b) $S_{all}$



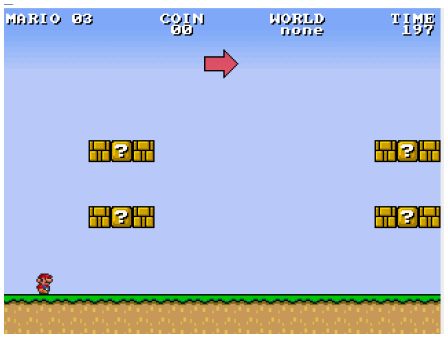

(c) $S_{vert}$



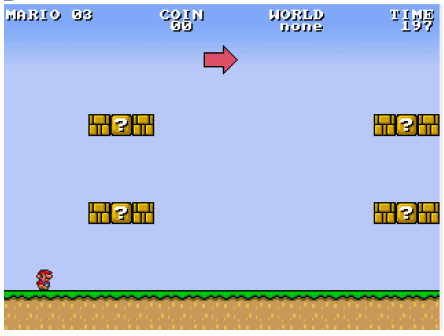

(d) Random



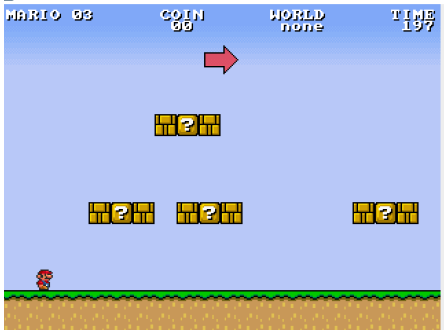

(a) Original




(b) $S_{all}$




(c) $S_{vert}$




(d) Random

# References

A, C., & Smith, G. (2015). Towards a procedural evaluation technique: Metrics for level design. In *Fdg.* ACM.

Bakkes, S., Whiteson, S., Li, G., Visniuc, G. V., Charitos, E., Heijne, N., & Swellengrebel, A. (2014). Challenge balancing for personalised game spaces. In *Games media entertainment* (pp. 1–8). IEEE Press.

Bauerly, M., & Liu, Y. (2006). Computational modeling and experimental investigation of effects of compositional elements on interface and design aesthetics. *International Journal of Human-Computer Studies*, *64*(8), 670–682.

Bayliss, J. D. (2012). Teaching game ai through minecraft mods. In *Games innovation conference (igic), 2012 ieee international* (pp. 1–4).

Browne, C. (2012). Elegance in game design. *Computational Intelligence and AI in Games, IEEE Transactions on*, *4*(3), 229–240.

Compton, K., & Mateas, M. (2006). Procedural level design for platform games. In *Conference on artificial intelligence and interactive digital entertainment* (p. 109-111). AAAI Press.

Dahlskog, S., & Togelius, J. (2013). Patterns as objectives for level generation. In *Proceedings of the workshop on design patterns in games at fdg.*

Dahlskog, S., & Togelius, J. (2014a). A multi-level level generator. In *IEEE conference on computational intelligence and games* (pp. 1–8).

Dahlskog, S., & Togelius, J. (2014b). Procedural content generation using patterns as objectives. In *Proceedings of the european conference applications of evolutionary computation* (pp. 325–336).

Dahlskog, S., Togelius, J., & Nelson, M. J. (2014). Linear levels through n-grams. In *Proceedings of the international academic mindtrek conference.*

Ferreira, L., & Toledo, C. F. M. (2014). A search-based approach for generating angry birds levels. In *Computational intelligence and games (cig), 2014 ieee conference on* (pp. 1–8).

Garey, M. R., & Johnson, D. S. (1979). *Computers and intractability: A guide to the theory of np-completeness.* New York, NY, USA: W. H. Freeman & Co.

Horn, B., Dahlskog, S., Shaker, N., Smith, G., & Togelius, J. (2014). A comparative evaluation of level generators in the Mario AI framework. In *Fdg.* ACM.

Kerssemakers, M., Tuxen, J., Togelius, J., & Yannakakis, G. N. (2012). A procedural procedural level generator generator. In *Conference of comp. intell. and games* (p. 335-341). IEEE.

Khatib, F., DiMaio, F., Cooper, S., Kazmierczyk, M., Gilski, M., Krzywda, S., ... Baker, D. (2011, 10). Crystal structure of a monomeric retroviral protease solved by protein folding game players. *Nature on Structural and Molecular Biology*, *18*(10), 1175–1177.

Lai, C.-Y., Chen, P.-H., Shih, S.-W., Liu, Y., & Hong, J.-S. (2010). Computational models and experimental investigations of effects of balance and symmetry on the aesthetics of text-overlaid images. *International Journal of Human-Computer Studies*, *68*(1), 41–56.

Li, M., Chen, X., Li, X., Ma, B., & Vit‡nyi, P. M. B. (2004). The similarity metric. *IEEE Transactions on Information Theory*, *50*(12), 3250 - 3264.

Liapis, A. (2016). Exploring the visual styles of arcade game assets. In *Evolutionary and biologically inspired music, sound, art and design* (pp. 92–109). Springer.

Liapis, A., Yannakakis, G. N., & Togelius, J. (2012). Adapting models of visual aesthetics for personalized content creation. *Computational Intelligence and AI in Games, IEEE Transactions on*, *4*(3), 213–228.

MARIÑO, J. R. H., Reis, W. M. P., & Lelis, L. H. S. (2015). An empirical evaluation of evaluation metrics of procedurally generated mario levels. In *Conference on artificial intelligence and interactive digital entertainment* (pp. 44–50). AAAI Press.

Mawhorter, P. A., & Mateas, M. (2010). Procedural level generation using occupancy-regulated extension. In *Conference of comp. intell. and games* (p. 351-358). IEEE.

Ngo, D. C. L., Samsudin, A., & Abdullah, R. (2000). Aesthetic measures for assessing graphic screens. *J. Inf. Sci. Eng*, *16*(1), 97–116.

Ngo, D. C. L., Teo, L. S., & Byrne, J. G. (2003). Modelling interface aesthetics. *Information Sciences*, *152*, 25–46.

Pedersen, C., Togelius, J., & Yannakakis, G. N. (2009). Modeling player experience in Super Mario Bros. In *Conference on computational intelligence and games* (pp. 132–139). IEEE Press.

Piselli, P., Claypool, M., & Doyle, J. (2009). Relating cognitive models of computer games to user evaluations of entertainment. In J. Whitehead & R. M. Young (Eds.), *Fdg* (p. 153-160). ACM.

Reis, W. M., Lelis, L. H., & Gal, Y. K. (2015). Human computation for procedural content generation in platform games. In *Computational intelligence and games (cig), 2015 ieee conference on* (pp. 99–106).

Salimun, C., Purchase, H. C., Simmons, D. R., & Brewster, S. (2010). Preference ranking of screen layout principles. In *Proceedings of the 24th bcs interaction*

*specialist group conference* (pp. 81–87).

Shaker, N., & Abou-Zleikha, M. (2014). Alone we can do so little, together we can do so much: A combinatorial approach for generating game content. In *Conference on artificial intelligence and interactive digital entertainment* (pp. 167–173). AAAI Press.

Shaker, N., Asteriadis, S., Yannakakis, G. N., & Karpouzis, K. (2011). A game-based corpus for analysing the interplay between game context and player experience. In *Affective computing and intelligent interaction* (pp. 547–556). Springer.

Shaker, N., Asteriadis, S., Yannakakis, G. N., & Karpouzis, K. (2013). Fusing visual and behavioral cues for modeling user experience in games. *Cybernetics, IEEE Transactions on*, *43*(6), 1519–1531.

Shaker, N., Nicolau, M., Yannakakis, G. N., Togelius, J., & O'Neill, M. (2012). Evolving levels for Super Mario Bros using grammatical evolution. In *Conference of comp. intell. and games* (p. 304-311). IEEE.

Shaker, N., & Shaker, M. (2014). Towards understanding the nonverbal signatures of engagement in Super Mario Bros. In *Proceedings of the conference on user modeling, adaptation, and personalization* (pp. 423–434).

Shaker, N., Togelius, J., & Nelson, M. J. (2015). *Procedural content generation in games: A textbook and an overview of current research*. Springer.

Shaker, N., Yannakakis, G., & Togelius, J. (2013). Crowdsourcing the aesthetics of platform games. *IEEE Transactions on Computational Intelligence and AI in Games*, *5*(3), 276–290.

Shaker, N., Yannakakis, G. N., & Togelius, J. (2010). Towards automatic personalized content generation for platform games. In *Conference on artificial intelligence and interactive digital entertainment* (pp. 63–68). AAAI Press.

Shaker, N., Yannakakis, G. N., & Togelius, J. (2011). Feature analysis for modeling game content quality. In *Conference of comp. intell. and games* (pp. 126–133). IEEE.

Shaker, N., Yannakakis, G. N., & Togelius, J. (2012). Digging deeper into platform game level design: Session size and sequential features. In *Evoapplications* (Vol. 7248). Springer.

Shaker, N., Yannakakis, G. N., Togelius, J., Nicolau, M., & O'Neill, M. (2012). Evolving personalized content for Super Mario Bros using grammatical evolution. In *Conference on artificial intelligence and interactive digital entertainment* (pp. 75–80). AAAI Press.

Smith, G., Cha, M., & Whitehead, J. (2008). A framework for analysis of 2d platformer levels. In *Acm siggraph symposium on video games* (pp. 75–80).

ACM.

Smith, G., Gan, E., Othenin-Girard, A., & Whitehead, J. (2011). Pcg-based game design: enabling new play experiences through procedural content generation. In *Proceedings of the 2nd international workshop on procedural content generation in games* (p. 7).

Smith, G., Treanor, M., Whitehead, J., Mateas, M., Treanor, M., March, J., & Cha, M. (2011). Launchpad: A rhythm-based level generation for 2d platformers. *IEEE Transactions on Computing Intelligence and AI in Games*, *3*(1), 1-16.

Smith, G., & Whitehead, J. (2010). Analyzing the expressive range of a level generator. In *Proceedings of the workshop on procedural content generation in games* (pp. 1–7). ACM.

Smith, G., Whitehead, J., & Mateas, M. (2010). Tanagra: a mixed-initiative level design tool. In *Fdg* (p. 209-216). ACM.

Sorenson, N., Pasquier, P., & DiPaola, S. (2011). A generic approach to challenge modeling for the procedural creation of video game levels. *IEEE Transactions on Computing Intelligence and AI in Games*, *3*(3), 229-244.

Tan, C. T., Bakkes, S., & Pisan, Y. (2014). Inferring player experiences using facial expressions analysis. In *Proceedings of the conference on interactive entertainment* (pp. 7:1–7:8). ACM.

Togelius, J., De Nardi, R., & Lucas, S. M. (2007). Towards automatic personalised content creation for racing games. In *Computational intelligence and games, 2007. cig 2007. ieee symposium on* (pp. 252–259).

Togelius, J., Karakovskiy, S., Koutník, J., & Schmidhuber, J. (2009). Super mario evolution. In *2009 ieee symposium on computational intelligence and games* (pp. 156–161).

Togelius, J., Kastbjerg, E., Schedl, D., & Yannakakis, G. N. (2011). What is procedural content generation?: Mario on the borderline. In *Proceedings of the 2nd international workshop on procedural content generation in games* (p. 3).

Togelius, J., Preuss, M., Beume, N., Wessing, S., Hagelback, J., & Yannakakis, G. N. (2010). Multiobjective exploration of the starcraft map space. In *Computational intelligence and games (cig), 2010 ieee symposium on* (pp. 265–272).

Togelius, J., Shaker, N., Karakovskiy, S., & Yannakakis, G. N. (2013a). The mario ai championship 2009-2012. *AI Magazine*, *34*(3), 89–92.

Togelius, J., Shaker, N., Karakovskiy, S., & Yannakakis, G. N. (2013b). The Mario AI championship 2009-2012. *AI Magazine*, *34*(3), 89-92.

Togelius, J., Shaker, N., & Nelson, M. J. (2015a). Introduction. In N. Shaker, J. Togelius, & M. J. Nelson (Eds.), *Procedural content generation in games: A textbook and an overview of current research.* Springer.

Togelius, J., Shaker, N., & Nelson, M. J. (2015b). search-based-approach. In N. Shaker, J. Togelius, & M. J. Nelson (Eds.), *Procedural content generation in games: A textbook and an overview of current research.* Springer.

Togelius, J., Yannakakis, G. N., Stanley, K. O., & Browne, C. (2011). Search-based procedural content generation: A taxonomy and survey. *IEEE Transactions on Computational Intelligence and AI in Games*, *3*(3), 172–186.

Yerkes, R. M., & Dodson, J. D. (1908). The relation of strength of stimulus to rapidity of habit formation. *Journal of Comparative Neurology and Psychology*, *18*, 459–482.