INVESTIGATION INTO GAME-BASED CRISIS SCENARIO MODELLING AND SIMULATION SYSTEM

Pisit Praiwattana

A thesis submitted in partial fulfilment of the requirements of Liverpool John Moores University for the degree of Doctor of Philosophy

June 2018

DECLARATION

This dissertation is the result of my own work and includes nothing, which is the outcome of work done in collaboration except where specifically indicated in the text. It has not been previously submitted, in part or completely, to any university of institution for any degree, diploma, or other qualification.

Signed:____**Pisit Praiwattana**_____

Date:_____**8 Sep 2018**_____

Pisit Praiwattana

Department of Computer Science

Liverpool John Moores University

Byrom Street

Liverpool

L3 3AF

UK

ABSTRACT

A crisis is an infrequent and unpredictable event. Training and preparation process requires tools for representation of crisis context. Particularly, crisis events consist of different situations, which can occur at the same time combining into complex situation and becoming a challenge in coordinating several crisis management departments.

In this regards, disaster prevention, preparedness and relief can be conceptualized into a design of hypothetical crisis game. Many complex tasks during development of emergency circumstance provide an opportunity for practitioners to train their skills, which are situation analysis, decision-making, and coordination procedures.

While the training in physical workouts give crisis personal a hand-on experience in the given situation, it often requires a long time to prepare with a considerable budget. Alternatively, computational framework which allows simulation of crisis models tailoring into crisis scenario can become a cost-effective substitution to this study and training.

Although, there are several existing computational toolsets to simulate crisis, there is no system providing a generic functionality to define crisis scenario, simulation model, agent development, and artificial intelligence problem planning in the single unified framework. In addition, a development of genetic framework can become too complex due to a multi-disciplinary knowledge required in each component. Besides, they have not fully incorporated a game technology toolset to fasten the system development process and provide a rich set of features and functionalities to these mentioned components.

To develop such crisis simulation system, there are several technologies that must be studied to derive a requirement for software engineering approach in system's specification designs. With a current modern game technology available in the market, it enables fast prototyping of the framework integrating with cutting-edge graphic render engine, asset management, networking, and scripting library. Therefore, a serious game application for education in crisis management can be fundamentally developed early. Still, many features must be developed exclusively for the novel simulation framework on top of the selected game engine.

In this thesis, we classified for essential core components to design a software specification of a serious game framework that eased crisis scenario generation, terrain design, and agent simulation in UML formats. From these diagrams, the framework was prototyped to demonstrate our proposed concepts.

From the beginning, the crisis models for different disasters had been analysed for their design and environment representation techniques, thus provided a choice of based simulation technique of a cellular automata in our framework. Importantly, a study for suitability in selection of a game engine product was conducted since the state of the art game engines often ease integration with upcoming technologies.

Moreover, the literatures for a procedural generation of crisis scenario context were studied for it provided a structure to the crisis parameters. Next, real-time map visualization in dynamic of resource representation in the area was developed. Then the simulation systems for a large-scale emergency response was discussed for their choice of framework design with their examples of test-case study. An agent-based modelling tool was also not provided from the game engine technology so its design and decision-making

procedure had been developed. In addition, a procedural content generation (PCG) was integrated for automated map generation process, and it allowed configuration of scenario control parameters over terrain design during run-time.

Likewise, the artificial planning architecture (AI planning) to solve a sequence of suitable action toward a specific goal was considered to be useful to investigate an emergency plan. However, AI planning most often requires an offline computation with a specific planning language. So the comparison study to select a fast and reliable planner was conducted. Then an integration pipeline between the planner and agent was developed over web-service architecture to separate a large computation from the client while provided ease of AI planning configuration using an editor interface from the web application.

Finally, the final framework called CGSA-SIM (Crisis Game for Scenario design and Agent modelling simulation) was evaluated for run-time performance and scalability analysis. It shown an acceptable performance framerate for a real-time application in the worst 15 frame-per-seconds (FPS) with maximum visual objects. The normal gameplay performed capped 60 FPS. At same time, the simulation scenario for a wildfire situation had been tested with an agent intervention which generated a simulation data for personal or case evaluation.

As a result, we have developed the CGSA-SIM framework to address the implementation challenge of incorporating an emergency simulation system with a modern game technology. The framework aims to be a generic application providing main functionality of crisis simulation game for a visualization, crisis model development and simulation, real-time interaction, and agent-based modelling with AI planning pipeline.

ACKNOWLEDGEMENTS

Thanks to my supervisor, Professor Abdennour El Rhalibi for his constructive suggestions over this study. Both thanks toward family and friends for their supports are invaluable during a hardship and stress.

CONTENTS

LIST OF TABLES

LIST OF FIGURES

1 INTRODUCTION

In this research project, we have proposed to investigate the use of serious game approach in development and evaluation of game-based simulation framework for crisis scenario generation, disaster modelling, resource management, planning, and analysis of the outcome.

Serious games have become a cost-effective paradigm for supporting real-world educations in theoretically simulating, managing, and training in preparation before investing in physical workouts. Example of applications include areas such as defence, medical practice, and crisis management. In general, serious game approach normally includes a wide range of techniques and tools to support visualization, asset and environment creation, interaction plus decision-making support with defined knowledge in related application domains. With this aspect, rather than being exclusively assigned for training and education, serious game system can become a useful tool for real-time management of crisis.

Although the usage of serious game approach may have been identified implementing in a wide range of research systems, there are issues need to be addressed for development process of such framework involving a variety of components to provide fundamental functions and interfaces for solving real-world complex applications. Some of these issues include how to representing a real-world data into a computation model; creation and management of graphical assets; authorizing a narrative for crisis scenario; artificial intelligence modelling and simulation; conflict management and cooperation method. This thesis will highlight areas in which current researches does not properly fulfil the general solution for the development and analysis of game-based crisis scenario simulation and generation system then proposed a novel approach to design, implementation and provide self-evaluation.

1.1 Problem Statements

The major challenge in this research is to simulate the nature of real-world crisis, which its situation is difficult to model into general representation for every possible alternative of crisis scenario. While computer computation model has been applied to provide cost-effective solution for crisis modelling and testing,

insufficient parameters in attempt to represent crisis may lead to decease in quality and very straightforward generated scenario. Whereas too many variables increase computation complexity, thus, reducing the scalability of framework significantly. This issue requires experimentation on how to achieve a compromised model providing an acceptable realistic crisis scenario result.

Another problem is that a crisis simulation framework must consolidate different multi-disciplinary approaches and technologies such as visual content generation, artificial intelligent agent model, and its behaviour structure, AI planning, domain-and-problem knowledge representation for AI planning, and deployment of cross-platform solution. Integration of these hybrid-components are time-consuming in study, implementation, and testing. Evaluating the final system with multi-components also required consideration of quantitative and qualitative performance metrics. The existing systems are often inaccessible as a specific funding project from a government or else being too specific for only a single type of simulation case. Finding a comparative metric then become unreliable for which system is the best in performance or scale.

In addition, a further issue includes how to design and provide acceptable real-time interaction system. Such a system should deliver a flexibility to define generated scenario and crisis plan while can still interface with the simulation parameters during run-time. Functional interface can bridge a gap for domain experts to participate in the design phase and ongoing model without prior scripting knowledge.

From our preliminary study, we have evaluated and consolidated the current research showing that current approaches to develop such a crisis framework are not sufficient provided and formulated into a general practice of using a modern game technology. Especially, the mean to construct a fast-development framework with flexible crisis modelling solution able to subsist with real-world complex problem involving in all aspects of representation, content-generation, simulation, optimization, and analysis method.

Therefore, these issues influence the project aims and final design of the system so that it can become a generic tool for purpose of study crisis, design scenario, modelling disaster, and outcome simulation.

1.2 Project Aims

The project aims are to address the issues related to developing a computational crisis simulation framework for crisis scenario representation, simulation, agent automate behaviour, AI planning, and outcome evaluation. The classification study of core components and its technologies has been included in the project preliminary stage to address a design of the system as a software engineering solution.

The major aim of this research is to address the issues related to the use of serious game approach in development of game-based framework with the view to fulfil the limitation of current research and solutions mainly on integrating of existing necessary components into a generic crisis simulation framework especially AI planning.

While proposing model and architecture for integration of essential technologies such as procedural map and content generation, simulation logic, agent-based model package, and AI planning pipeline, this work also aims to include a mean for evaluation and analysis an outcome of hypothesis scenario of an emergency plan by addressing the example scenario with agents capable of AI planning to solve the crisis incidents.

We have expected that by developing and testing such a solution, we will eventually provide a unifying framework easing development of game-based system for crisis representation, scenario tailoring, AI planning, simulation, and analysis of realistic crisis situation outcome.

1.3 Project Objectives

To achieve the main aims of our research, we have proposed that the main objectives are to develop a prototype solution for a flexible crisis simulation model, which combines a key functionality from the study of existing literatures. Then we have focused on formulating an improvement idea on integrating AI planning with crisis simulation framework using web service architecture into our software specification. The framework design has been involved all fundamental components for a framework providing general features for crisis scenario simulation and design.

The objectives will be validated by examining and extending the existing frameworks, and approaches; and by contrasting them qualitatively and quantitatively to our proposed method.

The more specific research objectives will include:

- To provide a classification study for identifying essential design and technology for developing a serious game application in crisis simulation.
- To design an asset management system for gathering and managing the collective data representing a realistic complex crisis scenario which includes context-knowledge, environment and resources asset, agent's behaviour, events, relations and constraints between agents. This functionality supports scenario designer in representing and evaluating the possible outcome of proposed context case.
- To develop a base component for representing genetic crisis model then demonstrate an extension of the based model into a natural disaster case such as wildfire and flooding using cellular automata.
- To develop multi-agent architecture model dedicated for a serious game application in crisis simulation. It includes a communication between components such as the environment and simulator systematically.
- To develop an event manager component focusing in generating a plausible narrative structure for crisis scenario. This will include function to track the scenario progress and establishing interaction possibility during scenario simulation to modify world representation and context directly, which may reflect in change of direction for agent's final decision.
- To develop a procedure easing 3D environment visualization components for dynamic scenario simulation and allowing a real-time environment alteration feedbacks.
- To incorporate AI planning technology to improve agent decision-making process by integrating with web technology for ease of configuration, result tracking, and potentially, reduce of computation complexity.
- Finally, to develop evaluation metrics to evaluate the scalability of crisis-scenario simulation framework, and contrasting the results with existing systems or approaches in applicable aspects.

1.4 Novelties and Contributions

We have developed a novel crisis game for scenario design and agent modelling simulation framework (CGSA-SIM) on Unity3D addressing the challenges in integrating a serious game technology with crisis management domain for an emergency study, experimentation, and preparedness.

The environment and crisis representation have been investigated from the literatures then designed and implemented using procedural content generation techniques, whilst crisis generation has been designed and implemented as a cellular automata technique using a hexagonal shape. This approach facilitates a development of hypothetical crisis simulation models with an observable relationship toward areas of incident. The crisis scenario designer can access the application toolkit to design and adjust a scenario context and scripting disaster events effectively. We have developed and demonstrated the integration of a wildfire model and flood models in the thesis with the visualization as a validation result while further models can be extended with ease. Moreover, procedural generation technique has been incorporated into a visualization of the 3D terrain. As a result, it allows a simulation model to reflect a dynamic change of environment state in a real-time. The agent-modelling toolkit to represent crisis stakeholders for study and simulate crisis intervention has been designed. It allows a freedom to model different type of intelligent decision-making behaviour. Another main novelty on the agent component is that we have designed a pipeline to integrate AI planning using a modern planner such as Fast Forward Planning (FF) wrapped in a Java runtime (JavaFF). It extends an option for complex decision-making capability for an agent on the game-based crisis simulation system. Finally, the evaluation of game-based crisis simulation performance has been evaluated with a test cast study on hypothetical wildfire scenario.

The CGSA-SIM framework provides strong solid system, demonstrating an example in integrating the current game-based technologies with computational aspects of crisis representation using the cellular automaton to deliver a generic framework assisting to create, simulate, and observe for deriving more understanding of crisis and generation of crisis scenario. In particular, it involves the following novelties and contributions:

- Integration of dynamic map procedural generation algorithm for interactive designing 3D environment and instantaneously visual crisis simulation feedbacks.
- Comparative studies on designing cellular automata models as a foundation technique for developing natural disaster model in the game-based framework.
- A new integration pipeline for AI planning on agent modelling with game-based crisis simulation framework using web service.
- A novel framework for crisis management system integrating all of the above.

1.5 Structure of Thesis

The structure of thesis consists of seven chapters including:

1. Introduction – This chapter covers the research challenge, project aims and objectives, novelties and contributions of thesis.

2. Background – This chapter provides the background knowledge and their recent literatures relevant to the field of crisis, crisis management, serious game, artificial intelligent, multi-agent system, AI planning , constraint satisfaction, and game engine technologies.
3. Related Literatures – This chapter introduces a comprehensive literature review of related research to discuss their solutions and limitations, and give comparison on their features, and possible improvement for a specification of thesis proposed solution.
4. Framework specification – This chapter lists a specification of a system requirement and technology for the proposed CGSA-SIM framework. A brief overview of system architecture is also presented.
5. Framework design– This chapter covers a design explanation of each components and a process description of each corresponding subsystem. Finally, a UML class diagram for the software development of the proposed CGSA-SIM framework is presented.
6. Implementation – This chapter covers the implementation detail for the proposed CGSA-SIM framework main process. A visualization process of map terrain is described. A disaster model development using cellular automata is explained. The example model implemented into the framework is also presented. In addition, an agent implementation and its AI planning component has been described.
7. Evaluations and Results – This chapter shows a collection of results, and then demonstrates the methodology of performance evaluation and quality evaluation with test-case scenario.
8. Conclusion – This chapter summarized the thesis. It covers the novelties and contributions to the knowledge in crisis management and crisis scenario software engineering framework including limitations and future work.

2 BACKGROUND

2.1 Introduction

In this section, we describe the base knowledge and technologies focused on definition, challenge, and finding of any potential open problems. A reference to a crisis definition and its structure are presented including how game-based simulation technique can be applied in management of disaster situation. From understanding of crisis and crisis scenario, a computational software solution can be developed using a modern game technology as a serious game. A definition of the serious game is presented with its roadmap and review on a barrier of entry to the approach. As simulation system, utilization of artificial intelligent in the crisis model is a common practice to develop a hypothetical case study. An introduction knowledge to artificial intelligence as an agent is highlighted with its usage as a model in social simulation study, common pitfalls, and its appropriateness to crisis simulation application. While a crisis scenario can be generated using a knowledge of landscape demographic in the real world data, an automated approach to alleviate a time-consuming effort of scenario designer is preferable. A concept of procedural content generation and its related application to computer graphics, games, and simulation are reviewed. Next, the crisis simulation system requires a flexible technique to create a variety of crisis disasters. A theory of automata and cellular automata are presented as they have a potential to become a general foundation of any crisis model. To add an improvement with traditional agent architecture in crisis simulation framework, AI planning approach, planning language, and the example of planning system are presented and noted for its performance and capability. As a related field of study toward AI planning technique, a constraint satisfaction aspect is discussed and provided insight on how this approach can be benefit to crisis scenario simulation. Lastly, a definition of game engine and its core components are presented with comparative performance and feature result reviewed from literature.

2.2 Crisis and Crisis Scenario

There are several discussions on the definition, topology (classification), and discrete conceptual characteristic of a term '*disaster'* in an aspect of classical/hazard situation and social-science psychological model (Perry, 2007), while the result is that there is no absolute universal definition for crisis as it can be based on multi-discipline perception, location, temporal factors and scale of its context.

For an example, Lipman-Blumen (1975) develops typology of crisis dimensions for their conceptual crisis framework for social system prediction of family changes during World War II. We believe some features of this typology can be applied to identify scope of crisis scenario structure such as the following: Pervasiveness versus Boundedness (degree of impact for the limited part or an entire system); Precipitate onset versus Gradual onset (degree of suddenness and warning); Transitoriness versus Chronicity (short-term or long-term problem); Randomness versus Expectability (degree to which crisis can be expected); Natural generation versus Artificial generation (distinction whether it raised from natural condition or through man-

made effects). From these dimensions, (Barkun, 1977) compiles them with recognition from historical references of disaster then described three disaster categories as *Homeostatic* disaster, *Metastatic* disaster, *Hyperstatic* disaster, which includes a natural disaster, man-made disruption of local social and economic system, finally, global large-scale situation such as nuclear war respectively. With much diverse classifications, to the computational and simulation exercise, it is appropriated to narrow down a specific scope of situation.

For the current study, we define crisis using a reference study in domain of serious game application and computational simulation (Walker, Giddings, & Armstrong, 2011) as any event that is, or is expected to lead to an unstable and dangerous situation affecting an individual, group, community, or the whole society. These undesirable events can have a cause from human actions or natural phenomenon whilst it also includes local accidents, climate phenomenon, and plotted situations. In addition, characteristic of crisis are commonly infrequent, which they can occur over a time in different places and may be fast-paced and have unpredictable outcome, thus each crisis is unique with its own characteristic setup.

In crisis typology, a broad classification of crisis, coincidently, can be defined into four main types: (a.) Natural Disasters – flood, hurricane, wildfire, earthquake, tsunami, landslide; (b.) Environmental emergencies – technological or industrial hazardous accident; (c.) Complex political emergencies – breaking down of authority, conflict, and war; (d.) Pandemic emergencies – sudden outburst contingency disease resulting in economical and transportation disruption.

As for crisis management, a research dedicated for disaster loss-reduction are involved four phases as disaster mitigation, preparedness, response and recovery (Sutton & Tierney, 2018). In detail, it includes a study of pre-disaster vulnerability with mitigation approach, disaster model and post-response and recover policy while preparedness is connecting these two phases with current analysis of individuals, units, society, and organizational structure for deliverance of responses.

During the crisis, authority and managerial organizations have to handle resources and facilities efficiently to control the total damage. These resources have its nature to change over-time depend on current critical situations, which easily causes countermeasure plan to become inconsistent or even ambiguous, leading to unintentionally experiment of trial and error on new triggered operation. To mitigate study in preparation procedure, Walker et al. (2011) suggests that training from game with relevant real-world environment and well-defined crisis scenario can be beneficial to crisis management personals in understanding a practical basis for upcoming emergency situations. As an advantage, development of game-based training system allows budget-effectiveness in pre-study of theoretical operation before further committed to a setup cost of real-world practice.

In regarding to a scenario, it has been described as a hypothetical situation that provides a necessary environmental setup, initial background of event and a final goal to represent a sense of real-world situation. Quade and Carter (1989) mention this early base idea of using context-based political or industrial hypothesis for related analyst and experts. They can communicate their perception and judgement by game-given roles leading to outcome prediction for a situation that is difficult to represent with quantitative optimization model. Furthermore, by formulating this approach into a scenario in specific for crisis, the possible crisis-scenario structure can be separated into *context* and *crisis* (Walker et al., 2011). *Context* represents a pre-incident of crisis to enable understanding of specific environment surrounding the area of event. In the other hand, *crisis*

describes a set of specific events that lead to dangerous, troubled, and challenging situations, with potentially cascading effects. To broaden more possible outcome of scenario, the script usually includes partial unpredictable or probability-based chain of events to emphasis its uncertainty consequence.

By representing a research situation into a scenario, it can be serialized into a computational simulation script for finding an answer to a question of what is the optimal emergency response qualitatively and quantitatively concerning available resources allocation, workforce deployment practice, and mitigation procedures if the solution should exist.

Focusing on the computer simulation system for crisis scenario aspect, its architecture requires to facilitate a flexible structure of scenario script while allows visual observation of resource allocation, strategic decision-making, and evaluation of disaster quantitative impact. With Gamification approach, game-based application featured role-playing or overview situation observation will provide elaborated understanding of ongoing disaster with a preliminary assessment, and then theoretically supports a deliverance of post recovery plan. In the next section, it describes the necessary to understand how game-based approach can be related to the study of crisis and crisis scenario.

2.3 Serious Game

Regarding the gaming technology, an advancement of widely accessible game engine framework such as Unity3D, Amazon Lumberyard, Unreal Engine and incoming other tools have improved the pipeline of game development where they also reduce the barrier to entry for small team of developers and researchers to make prototype application.

Inevitably, increasing in accessibility to game development application has encouraged many government departments and scientific organizations to invest and apply game-based application approach. It has been used as a tool for non-entertainment purposes aiming to provide a mean for giving learning experience and gamification concept. The aspects related to game-based application involves visualization, interaction, competition, and risk-and-rewarding system, immersiveness and many more. There are many researchers that explore a usage of serious game such as Zyda (2005) as also one who introduces a classification of terms for edutainment game as *serious game*. Moreover, Shubik (1972) also mentions the usage of gaming from related research in the purpose of teaching, training, operations and strategy exploration, and decision-making experimentation.

By considering the field of crisis management and disaster planning, serious game offers a great potential to address the aspect of integration of multi-disciplinary techniques such as disaster modelling, visualization, interaction, communication and assessment prediction of environment simulation. Since the crisis is consisting of ever-changing nature, by representing and testing the possibility of outcome in the virtual world, it give an alternative option to solve the inflexibility of fully setup a demonstrated practice in many variables in real world as a cost-effective training and simulation practice solution. From roadmap described in Figure 2-1, it shows a possible direction of how serious game can become more relevant with recent advancement and accessible of virtual reality and augmented reality technologies during past years. The prospective improvement to the current system is to investigate in the usage of this new user interface so it can contribute more immersive in user participation reaction for future development of new serious game training and education system.

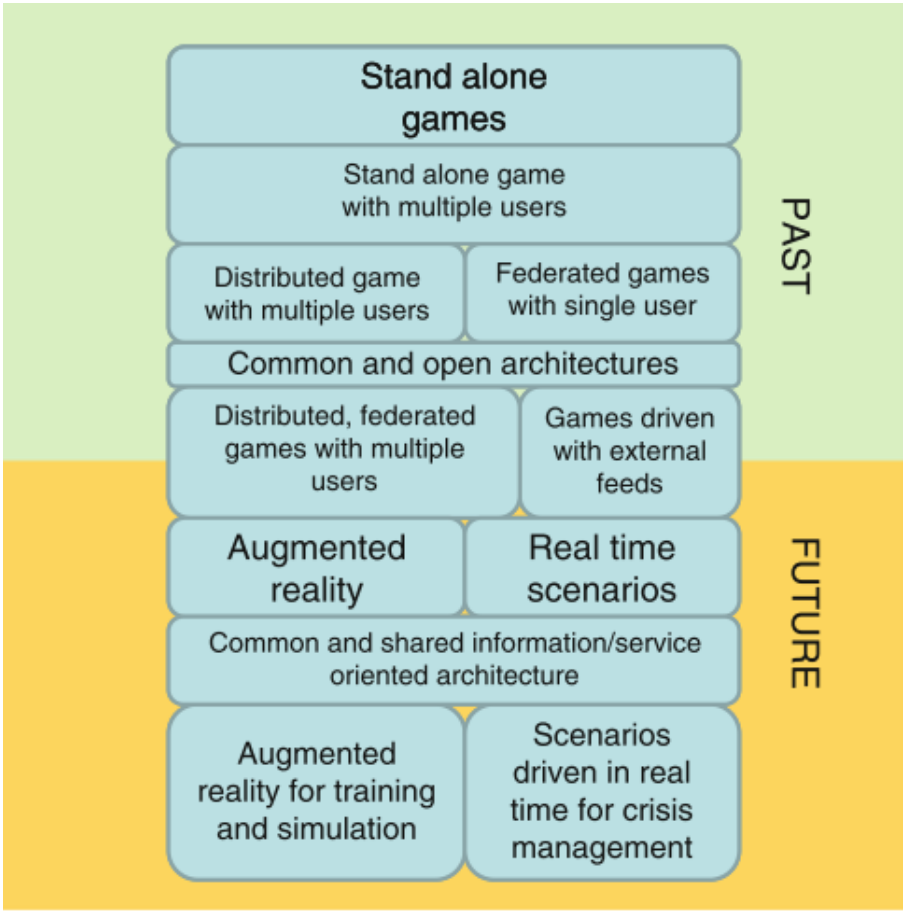


Figure 2-1 Gaming and Crisis Management Roadmap. Adapted from (Walker et al., 2011).

The prospect of gaming and simulation in crisis domain still suffer from these following issues (Walker et al., 2011): First, the involvement of domain experts are strongly required to ensure a subject-matter correctness of theory and evaluation. Most often, these participants are not available. Second, the retrieval of input data for establishing a simulation environment can become a strong barrier. This step often involves a data mining, data filtering, and data post-processing. While there is open geological database such as Open Street Map (OSM), the procedure required for converting a map dataset for visualization framework can become challenging. For an example, a square-meters data must be filtered into subset of building and road network, then interpolated for a missing position, extracting for 3D mesh composition, and they required efficient data structure for fast-query navigation (David Tully, El Rhalibi, Pan, Carter, & Sudirman, 2015a; D. Tully, Rhalibi, Carter, & Sudirman, 2016). Third, the validation and verification method to a model consisting of vary sub-models in behaviour representation and simulation mathematic concept can be difficult for traditional criteria. The cultural perception is also influenced how the prediction from simulation can be taken seriously or reflected as only an entertainment application with insufficient realistic aspect. Last, some simulation models may require considerable amounts of computation power. This issue has become less a barrier since an advancement in computational hardware and clustering architecture are improved constantly.

In addition, although it is very common that modern serious games stay focus on generating an immersive environment of real-world training, a delivering process of plausible scenario content is as crucial as a core engagement experience. Normally, these script and setup are directed from domain experts as mentioned in the earlier issue. This process itself is time-consuming and often lacks of benchmarks to evaluate

the quality of each different unique narrative. Due to this reason, with an unappropriated design, it will often lead to impractical preparation resulting a waste of computing time and training resources.

To assist in providing the content whether it is a visual environment object or scripting of the scenario, there are several systems aiming to generate these resources procedurally with a variety of computation techniques. However, the issue also resides in that many story-narrative scenario and visual environmental generation systems have been dedicated to only single simulation platform and, thus, is difficult to redeploy and integrated on different novel system.

In the next section, it introduces a knowledge of artificial intelligent in relation to its usage in a social science and crisis simulation system.

2.4 Artificial Intelligence, Agent Behaviour Model and Multi-Agent

Artificial intelligence (A.I.) is a field of study that aims to develop a software capable of autonomous and intelligent behaviour imitating a human reasoning process. Realizing this goal, there are several sub-problems that differentiate a trait in research direction such as reasoning for problem solving, knowledge representation, AI planning , machine learning, natural language processing and other related topics (Wooldridge & Jennings, 1995).

A set of software which is provided the ability to percept the changing in environment then choose to execute a set of behaviour actions to fulfil the objective programmed by developers. This type of software system are commonly regarded as an agent. Agents can be considered as a simple control vacuum unit, a chess player, or a simple actor in toy problem into a complex flexible space probe aiming to survey resource in an unknown environment. The key problem is how can agent evaluate and decide to select an action that can satisfy the given objective. S. Russell, Norvig, and Intelligence (1995) gives an explanation for the agent complexity by depending on the setup of environment properties as followings: (a.) an accessibility of an environment whether that the agent can percept the state of the world, which is accurate and up-to-date, or the agent must relies only local or limited knowledge; (b.) an effect of agent action whether the action is guaranteed for the outcome or depended on the probability of situation; (c.) a flow from the agent performance whether it is carried over to a subsequence event or just being limited into only current episode without worrying for the future interaction; (d.) an environment characteristic whether it remains unchanged without direct intervention effect of an agent action or just dynamically being influenced by other parties such that agent may not expect to control final outcome but take part on the result of every agent's responsibility; (e.) a situation where the environment can be expected for a finite number of actions and perceptions while, on the other hand, it may consist of unknown number of possible behaviour and perception sensor depending on current world status.

The most complex properties can be defined as inaccessible, non-deterministic, non-episodic, dynamic, and continuous. For the current agent software to operate feasibly, the assumption and implication of environment may require to be simplified. In addition, agent can be considered as intelligent agent if it were to sustain the capability to select a suitable operation on unpredictable environment although it may result in failure (Wooldridge & Jennings, 1995). The knowledge representation and different algorithms in the reasoning process distinguish different type of agents significantly.

Regarding an agent model for game-based system, a simple rule-based agent shown in Figure 2-2 is fundamentally sufficient to represent a single entity in the environment. In classical games such as chess or similar toy board games, an environment is often represented as a graph mapping where the agent and other object are residing in one specific node position, *V*, from all possible nodes in the environment space, *S*. Next, a direct path from a start node to a destination node is identified as an Edge, *E (Vi, Vj)*, with common practice of a distance (i.e. Euclidian or Manhattan) or other heuristic evaluation function as a traversal cost. Finally, path is a set of intermediate nodes from start to destination by traversing the smallest or with-in an acceptable length of total cost. The topic to search for possible shortest path has been referred as pathfinding and there are several algorithms being introduced over past decades such as breath-first search (Moore, 1959), Dijkstra algorithm (Dijkstra, 1959), and, most common practiced for game, A*(Hart, Nilsson, & Raphael, 1968). While the pathfinding is an important aspect for an agent, it is not necessary for every agent to move in the environment whereas, for an example, it can have only a role to detect in-burst of sensor perception.

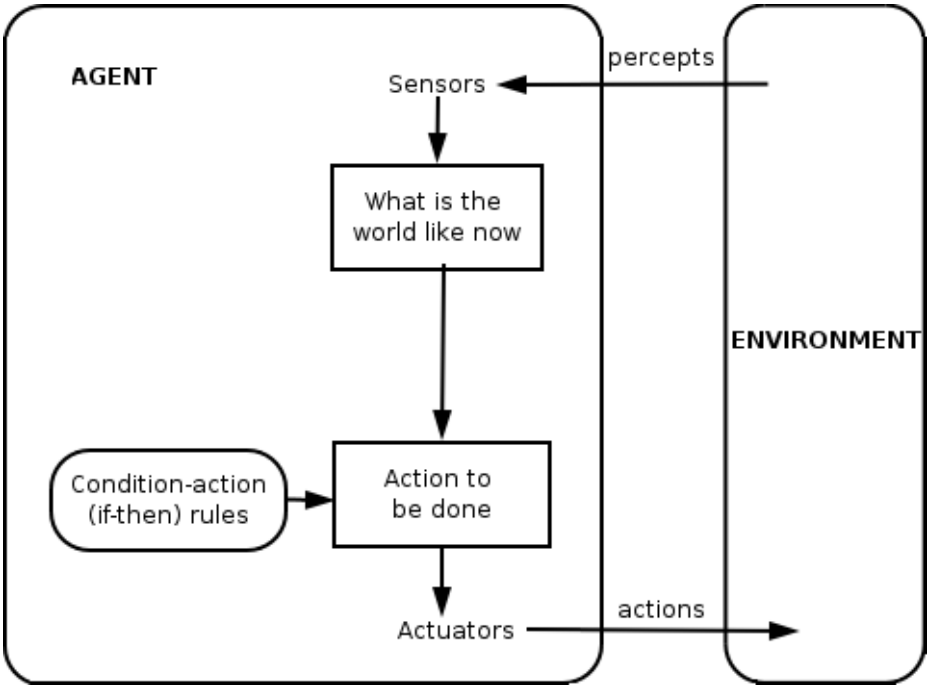


Figure 2-2 Simple Reflex Agent (S. J. Russell, Norvig, Canny, Malik, & Edwards, 2003)

In addition, there is often a demand in many applications to represent an interaction between individual agent cooperating to achieve a purpose or goal. For example, a logistic problem requires a management of transport route, medium, and quantity of goods to achieve a maximum efficiency, which can be defined as a utility function. Considering different autonomous individual with sensory and action capability as agent, the activity of coordination or negation even as one-to-one or one-to-organization as a formation of agent system can be called as multi-agent system (MAS) (Weiss, 1999).

From this aspect, the modelling of agent as an actor corresponding into one individual or hierarchical organization to study their interaction in virtual environment become an invaluable approach to both social science study and natural physic phenomenon simulation. In an example case, with multi-agent system, a scientist may model an environmental behaviour of wolf and sheep as a prey-predator ecosystem

(Kubera, Mathieu, & Picault, 2010) where sheep reproduces with grassland overtime while wolf reproduces with sufficient sheep as a food. Another example from social science study, Schelling (1971) has applied computer agent simulation to reveal clustering pattern of segregation from different ethical neighbourhoods such that each agent will try to group up on local individual with same tag while separated itself from another. Obtaining a visible or semantic-context solution from these systems result in better understanding of their behaviour, cascading effect from initial setup stage, and allow prediction of possible outcome regarding similar scenario's variables. Besides, there are no specific requirement that all agents must be equipped with the same capability and authority. Different agents can be provided with a common goal while varying in their own local objective. This situation introduces an approach to optimize the performance or utility functions in the system. Hierarchical organization and resource allocation are a main difficulty in the system that must be carefully defined since each agent may compete for the same resource over time in many cases. Sycara (1998) introduces the characteristic of multi-agent system as (1) each agent has incomplete information or capabilities for solving the problem thus having a different perspective in a problem viewpoint; (2) there is no system global control; (3) data are decentralized; and (4) computation is asynchronous. The problem with these properties often refers as distributed problem in which centralized system is not appropriate to operate. The multi-agent model for solving this specific type of problem is called distributed problem solver, DPS.

The design of MAS system often follows with these steps as model, analysis, and development of an autonomous individual agent to retain as an initial population of system. The integration between each agent can be cooperation, or self-pursue goal while neglecting other agent's demand. A good example of cooperation agents is ant colony where each agent aims to harvest food, fending off invaders, producing more population, and even pathfinding (Dorigo, Maniezzo, & Colorni, 1996). In more advance example, the team of robot compete with another in a collaboration performance of each team members whereas the challenging of this model resides in a dynamic and fast-paced decision-making environment (Kose, Tatlidede, Meriçli, Kaplan, & Akin, 2004). The most notable potential from the mentioned robot competition model is that each agent has been equipped with reinforcement learning capabilities where the assignment of behaviour can perform more accurate and appropriated for the next similar condition. In each simulation, learning agent can improve the notation of applying knowledge for greater benefit based on their trial-and-error experience or trained data set designed by an expert. Another example of successful learning agent is AlphaGo introduced using deep learning neural network that allows computer to successfully study and solve complex problem in game of GO then win the top professional players in the GO domains (Holcomb, Porter, Ault, Mao, & Wang, 2018). This prospect may become useful addition to crisis situation training and simulation.

As for the drawback from using agent-based model (ABS), the pitfall lies in the validity of such system is depending on an expertise of modeller while the calculation can get an error or produce unexpected artefact due to incomplete understanding of theory or, simply, coding error (Galán, 2009). Another noticeable challenge from using ABS is that it is difficult to calibrate a simulation outcome such that only one parameter may lead to disruption in an agent behaviour, this example is known as knife-edge threshold (Izquierdo & Polhill, 2006). While implementation of ABS can become tricky, an open framework for develop and test agent system is also available such as JADE multi-agent framework (Bellifemine, Caire, & Greenwood, 2007). In this regard, the study of ABS application could explore a suitability of the simulation theory using this tool before continue investing efforts in commercial or more complex framework.

In relation to crisis simulation applied ABS model, it seems appropriate that the representation of each role will be designed into a stakeholder agent capable of environment percept-response action. Within an early development phase, using if-else conditional syntax can provide quick verification of stakeholder model in regarding to different simulation system. Further, the straightforward individual agent model has a potential to be improved into multi-agent system with complex hierarchical structure and open for extension of advance decision-making such as heuristic planning or machine learning.

In the next section, the aspect of using automated content generation is discussed to ease the issue of reconstructing a crisis scenario information such as terrain and its details

2.5 Procedural Content Generation (PCG)

The study and development of algorithm for generation of content whether it is textures, structure of model vertices, game's object placement and even story narration automatically can be considered as procedural content generation (PCG). Roden and Parberry (2004) describe the limitation from handcrafted content has as at least four drawbacks. First, the advancement of technology requires more time to study and practice for artist in producing a qualified content. Second, the modification of created content become difficult once released since it may become impractical for framework developed in paralleled for that specific asset. Third, content creation tools often produce an exclusive format applicable in only specific engine or framework that indicates a redundant process of converting file format renowned for its different result from the original version. Last, the scale of upcoming systems is continuing to expand. The required time to create a content can become nearly impossible for human content creator to cover every aspect in the system. Therefore, the aspect of automatic content creation, which contents can be referred to any related adjustable entities that may affect the final perception experience of a user, is a promising improvement for costly and time-consuming process.

The requirement of using PCG is to define a relevant set of parameters that enables controlling of generating different output result. Trial and errors often played in a role of adjusting quantities of parameters in the interface for achieving desirable result. The algorithm often involves the usage of random generation number using specific set of integer called *seed* to reproduce a same set of output. The challenge for PCG is how it can be integrated the components effectively into target system with a demand of increasing quality and quantity of required content. One example pipeline of using PCG for content generation is such that the algorithm should produce a simplified set of data with random number generator, then synthesizing and amplifying it into more details in depth of hierarchical structures. Figure 2-3 shows the amplifying example of a basic 2D representation of terrain and its corresponding 3D version. With this pipeline, the major advantages are the reduction of storage for transferring and reproducing the very same content while provides a brief level-of-detail representation of content.

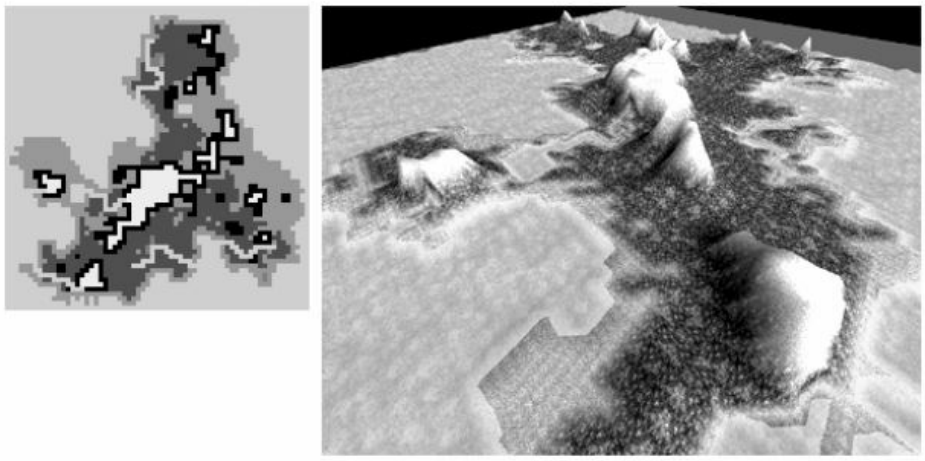


Figure 2-3 'Non-viewable' top-level coarse 3D height field (right) amplified directly from auto-generated 64 x 64 terrain map (left) (Roden & Parberry, 2004)

In early methods, the usage of PCG in 3D rendering techniques has gain more visibility from procedural texture generation by adjusting parameters of Perlin noise function (Perlin, 1985). The study preliminary focus on what the possibility of content can be generated. Then the researchers move onto how the desirable content can be achieve parametrically toward specific domain of output (Hendrikx, Meijer, Van Der Velden, & Iosup, 2013). One such example is to imitate the organic or pattern-based arrangement and decoration systematically, then the grammar symbol concept has been proposed and, commonly described as a mathematic computation technique - Linden Mayer system or L-system (Rozenberg & Salomaa, 1980). With description grammar describing the behaviour of drawing engine, it is applicable to produce a result of branching of graphical fractals, trees, road and street structure, and even urban expansion (Allen, Prusinkiewicz, & DeJong, 2005; Parish & Müller, 2001; Prusinkiewicz, 1986). Next, the generation of terrain also become another important focus of implementing PCG to facilitate an increasing demand of novel content with replayability (Sampath, 2004) since it results in achieving a feasible proportion of investment of time and resources. While the procedural content generation has been observed the usage in construct a data from scratch, it can be applied in a content selection process for generating more specific content by combining a several base independent modules. The recent example is the educational application using automated content generation in assisting of the content selection for tailoring dedicated user learning media via genetic algorithm (Hooshyar, Yousefi, & Lim, 2018).

Evidently, there are several commercial games product as Rogue (Troy and Wichman, 1980), The Elder Scrolls II: Daggerfall (Bethesda Softworks, 1996), Diablo (Blizzard Entertainment, 1998), and, most famously, MineCraft (Duncan, 2011), Portal Knight (2018) for whole 3D level, Terraria (Re-Logic 2011), Dead Cells (2018) in 2D platformer genre, successfully implemented the core of their systems with PCG for dungeon and terrain content generation. These games sometimes refer the concept of level generation as procedural dungeon generation, which computationally resolves the generation of topology, geometry, and game-played related objects' attributes in the player-specific level including a non-player characters, decorations, and even selection of suitable media contents (lighting and music). The pipeline method described for these games are relevant to previously mentioned concept of PCG. They often includes these three basic elemental steps: 1.) a representation of model: a simplified version of map or dungeon with overview of result;

2.) a method for reconstructing more elaborated detail from previous step; 3.) a method for reconstructing geometry of a level from its representation model.

In addition, one of the famous procedural generation method applied for level generation in game and research framework is cellular automata, a self-organized structure consisted of a grid of cell in finite number dimensions. Johnson, Yannakakis, and Togelius (2010) used a self-organized characteristic of cellular automata to generate a cave-level with a usage of defining neighbour in 8 directions surrounding focus cell in three possible states as floor, rock and wall, then applying transition rule iteratively. As a result, the final cave-like level can be generated with infinitely variation and size as shown in Figure 2-4. The grey areas represent floor, red areas represent walls, and white areas represent rocks. The study for using CA in map generation has been further explored with the usage of a genetic algorithm for evolving a CA's rule to create a high quality of maze procedurally (Chad, 2018).

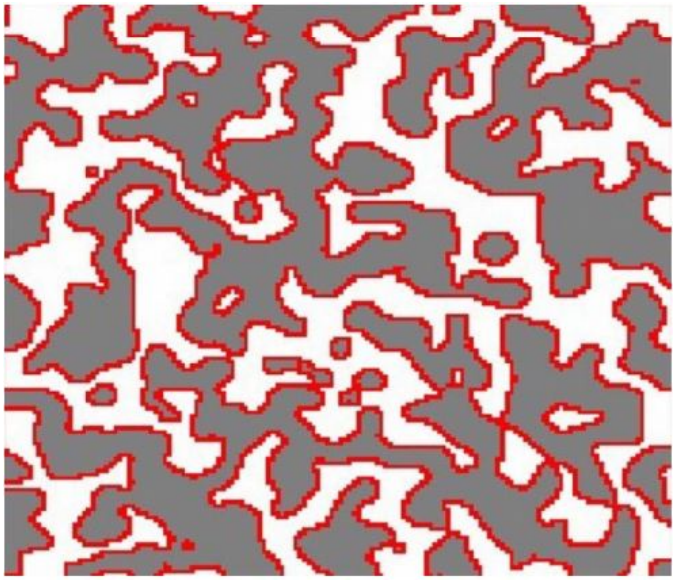


Figure 2-4 Map generated with cellular automata. (Johnson et al., 2010)

The potential of integrating PCG method for crisis simulation technique is invaluable. The issue of time-consuming content generation in especially level-design can be covered with supportive interface to control PCG parameters. However, the final detail should be corrected or evaluated by human designer to ensure the quality of the result.

Moreover, in a data acquisition process, for scenario context such as terrain, structure 3D model positioning, and textures, the terrain editor with PCG elements can become a quicker setup in contrast to pre-processing time for using large-GIS data. With theory and crisis model assumption verified in simpler system, the more time-consuming and data-storage requirement method with GIS have be deployed for a specific simulation later on.

2.6 Cellular Automata

Automata Theory has been a starting point for study of theory and design of automatic computation process imitating a human feature into a machine which is referred as *automata*. For *automaton*, it is a denotation to an abstract machine model which performs a computation based on first receiving input parameters then

moving it through internal finite configured states until reaching an acceptable or terminal outcome (Amal Dar Aziz, 2004). The theory itself has been a foundation for early finite-state-machine system and been developed into famous Turing machine (Turing, 1937) as a model to modern computer system.

The basic definition of automaton consists of four type of data (Eilenberg, 1974):

$$State: a\ finite\ set\ Q\ of\ element\ called\ states$$

$$Initial\ States: a\ subset\ I\ of\ Q; the\ state\ in\ I\ are\ called\ initials$$

$$Terminal\ states: a\ subset\ T\ of\ Q; the\ states\ in\ T\ are\ called\ terminal$$

$$Edges: a\ subset\ E\ of\ Qx\sum xQ.$$

A tuple (p, ∂, q) in E is called edge of the automaton representing the transition function starting from p to q states. With input set defined in a sequence of symbol from set X = $(x_1, x_2, x_3 \dots, x_k)$, where k is the number of input, the machines produced output in sequence of symbols from set Z = $(z_1, z_2, z_3, \dots, z_m)$, where m is the number of output corresponding to its state transition of input data. From this definition, finite state machine (FSM) referred to an automaton having state set Q only contain a finite number of elements. By contrast, FSM is considered to have less computational power than Turing machine since its memory is limited by number of states (Hopcroft, Motwani, & Ullman, 2001).

Likewise, the study of cellular automata (CA) is an another autonomous process first introduced by von Neumann and BURKS (1966), which the work is motivated to describe a biological concept of self-replicating model for universal constructor (UC) in two dimensional grid of cells with 29 possible states. To look further in the self-replication process, UC consists of two different parts: the former, a tape that contain an information to the construction process of cellular machine and, the latter, the constructer that reads the tape then construct another constructor using constructor arm shown in Figure 2-5. In this regard, if the information on the tape contain a description to fabricate another constructor, and the constructor can also duplicate an information on the tape, hence, the result is this system has obtain a self-replication ability (Rossier, Petraglio, Stauffer, & Tempesti, 2004).

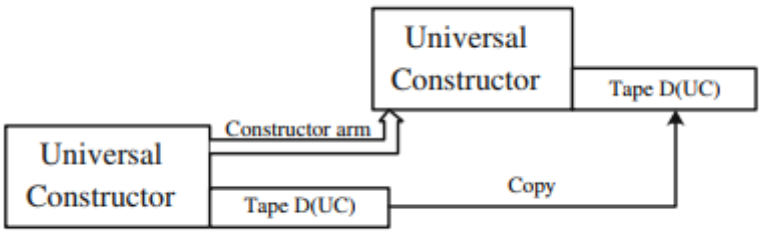


Figure 2-5 Universal Constructor Self-replication (Rossier et al., 2004)

In basic definition for cellular automata, each cell with possible finite state S inside a grid updates its state synchronously at a discrete time step. The stimuli for state-change is from neighbouring cell, or a nearest surrounding cell of target evaluation, based on the uniform rule applying to every cell. The common representation of CA in discrete dimensions is usually in the form of following format (Kari, 2005):

$$(S, N, f)$$

Where:

S is a states set for all cell in every distinct d − dimension address

N is a neighborhood vector for each cell

f is a local update rule mapping one state to the next

In application of CA in two-dimensional spaces, there are two common practices of neighbourhood model (Zaitsev, 2017) shown in Figure 2-6. The former is von Neumann's neighbourhood (von Neumann & BURKS, 1966), which considers state of five cells in the transition function. The latter, Moore's neighbourhood (Moore, 1962), which differentiates that the definition of his model is to involve eight coverage cells and the current centre. Moore (1962) explains that there is no different impact from using his model compared to von Neumann' as the additional four cells can be ignored in the update rule if desired. In motivation, both models are aiming to formulate a mathematical model of self-replication in grid-lattice automata, and a computational model.

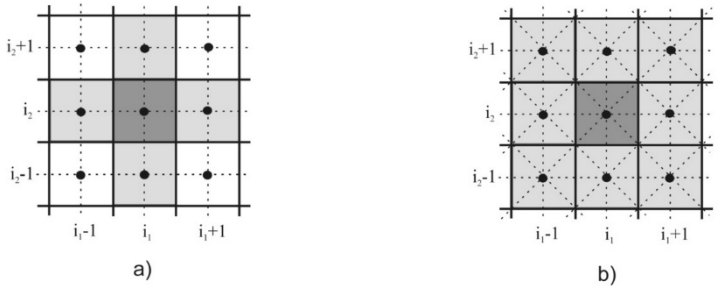


Figure 2-6 A classical neighbourhoods (2a): a.) von Neumann's neighbourhood; b.) Moore's neighbourhood (Zaitsev, 2017)

Together within each component functioning on the same rule, the complex pattern or behaviour can be observed, thus, it allows modelling of the living ecology and general principle for complex system in nature (McIntosh, 1990; Wolfram, 1984b). Moreover, the study of natural complex pattern can be investigated using cellular automata. These pattern gives observable evolution behaviour from starting row called seed. This elemental cellular automaton is described in a different basic rule preliminary assigned with 8-bits sequence code with zero and one to control next-row draw direction. The classification by rule of 8-bits representation have deliver a variety of rule-index typology (Wolfram, 1984a). Figure 2-7 demonstrates the example pattern produced from rule 110 and 22 respectively.

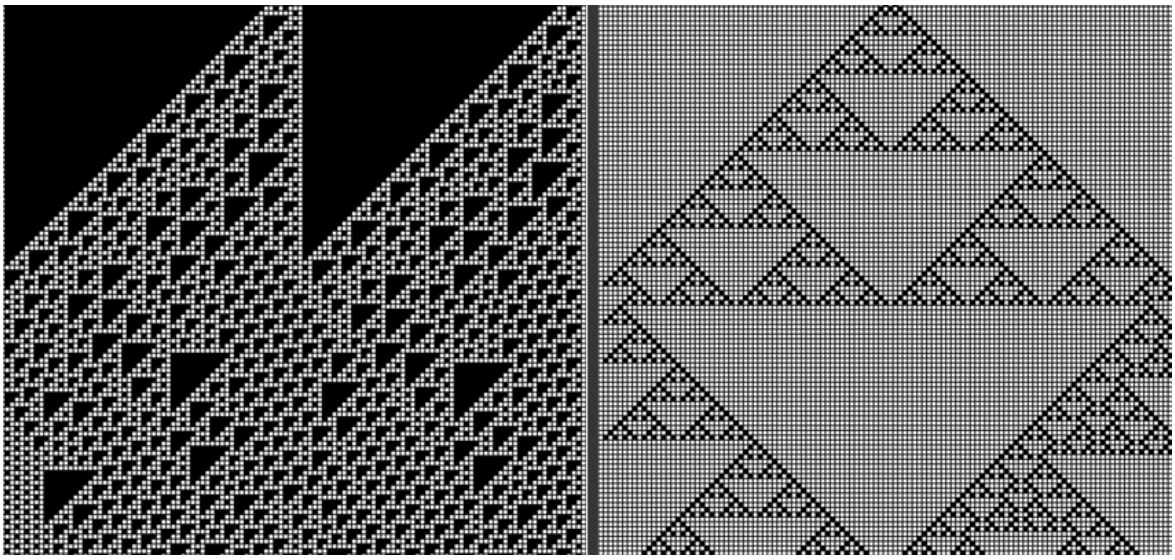


Figure 2-7 Result pattern from Elementary Cellular automata: (left) Rule 110; (right) Rule 22 reconstructed result using processing2D framework

Another famous elementary CA model implemented in 2D lattice is *Conway's Game of Life* (Bays, 2010) in which each cell has 2 states and by calculating the total eight neighbour cells, four adjacent orthogonally and for adjacent diagonally, new state can be derived from applying the sum of alive cells using a local rule to governing a death, alive and reproduce behaviour. The following is the base rules for Game of Life:

- *Survivals*. Every counter with two or three neighbouring counters survives for the next generation
- *Deaths*. Each counter with four or more neighbours dies (is removed) from overpopulation. Every counter with one neighbour or none dies from isolation
- *Births*. Each empty cell adjacent to exactly three neighbours--*no more, no fewer*--is a birth cell. A counter is placed on it at the next move.

Figure 2-8 shows the result of ongoing time-step from applying rules to each cell. The former, a pattern in left figure perform oscillating from death and alive of 2 cells at both edges creating a motion swap back-and-fourth of shapes. While the latter, this primitive arrangement creates a stable pattern of cell group with no motion. From this example, it demonstrates that with few rules, this technique can represent a dynamic evolution system of population of each cell and it is very powerful to simulate a phenomenon whereby its behaviour is depending on the interacting of nearby different type of state and basic type of transition function such as fluid, pollution, fire and epidemic outbreak.

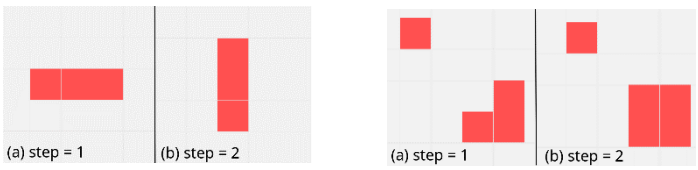


Figure 2-8 Cell oscillating between two patterns in consecutive steps (Left) Cell stabilizing pattern (Right)

Finally, the concept of cellular automaton have a flexibility to be integrated with procedural content generation technique especially in modelling the terrain topology and demonstrating a biological system or natural phenomenon ecology. Its general characteristic has ease of modelling an influential effect from surrounding neighbour effectively and, thus, become a suitable technique to formulate crisis simulation.

2.7 Planning and Planning Domain Definition Language (PDDL)

Planning is a process of generating a set of actions, which aims to achieve the best possible outcome or specific objective by executing the action list in sequence. In general, it is a valid sequence of actions from initial state to goal in state space set (Ghallab, Nau, & Traverso, 2004). Automated planning is an area in artificial intelligence to study this process computationally. There are several examples where planning is applicable. Path and motion planning is a synthesis of geometric or trajectory path from starting state to the goal state while configuring trajectory or movement variable. The good example is movement or truck and mobile mechanical robot part – arm. Perception planning is a process where the priority is to make a decision on selecting appropriate sensing action to gather most related information and, sometimes, identify the current state of environment. Navigation planning is a combination of two previous planning. It requires to gather the necessary information or using a sensor to determine a state of environment when allow movement or action to reach the defined goal given.

AI planning approach can have a domain specific or domain independent characteristics. The former is where all specific form of planning is well defined with a limitation in several aspects normally in commonality in each planning is not considered. The latter relies on generating an abstract model of actions to represent flexibility in representing a form of reasoning model. Actions may be defined with setup of temporal constraint, resource constraint, and conditional plus post-effect to state of the environment.

A typical description of AI planning problem consists of an initial state, goal, and available set of actions. A solution is generated as a sequence of actions that is executed in order to satisfy goal condition state. In this meaning, AI planning is to discover such solution. Planning becomes more feasible in many real-world applications such as space probes, robot, sensory networks, interactive storytelling, and including a game industry due to a trend of open-world and realistic environment satisfaction. However, the defining of domain, state variable, available action set, and its applicable condition may become overwhelming process to a designer. Moreover, planning process is often required a considerable effort to deliver a real-time solution in very complicate problems especially with a large set of objects and possible states.

Stanford Research Institute Problem Solver (STRIPS) represents a basic model of planning language and planning solver. It consists of mentioned stated properties, states, and actions mentioned above. States can be defined as a combination set of Boolean literal variable. Actions are described as precondition and post-effects. Goal states can be reached by triggering an appropriate satisfied-precondition action to have state transition. STRIPS is suitable for simple close-world assumption such as Block-World problem, but it is still not sufficient to include complex real-world constraints. Therefore, several extension features has been developed in STRIPS-based approach planner to consolidate flexible representation (Fikes & Nilsson, 1971).

The Planning Domain Definition Language (PDDL) is a description language used in modelling planning problem (Aeronautiques et al., 1998). Besides, on developing from STRIPS language, a further extension has been proposed to add capability to represent much more complex problems. It includes a

description of action definition language (ADL) to improve initial issue providing more capability and flexibility such as conditional goal, numeric function, metric, and durative action. Several extensions of PDDL are still in development and may be published in future. The most basic version supported in general planning system is PDDL 2.0 and PPDL 2.1 while fully supported ADL languages is commonly provided in PDDL3.0 and later.

In application aside of logistic and classical toy problems, planning has been majorly applied in a development of an interactive storytelling system (IS) (Barros & Musse, 2007; Duarte, Goudoulakis, El Rhalibi, & Merabti, 2013; El Rhalibi, Goudoulakis, & Merabti, 2012; Padia, Bandara, & Healey, 2018; Porteous, Cavazza, & Charles, 2010). The general approach is to define an initial setup of scene variable, possible role-playing action and introduce the narrated story using planning algorithm. The difficulty relied on how to define a suitable level of detail that results in an interesting expressiveness in story from a plan solution. Generating a story-rich result require several extensions from current PDDL such as equality, negative precondition, conditional result, exist, numeric and capability to produce partial-order plan. However, these extensions are not fully supported in all planner systems so that evaluation must be conducted to select most appropriate planner to the usage condition of the final application. There are also several AI planning systems available for integration with general application aside from storytelling domain. For PDDL is specified to have a modular feature, many systems adopted at least a STRIP characteristic while provide additional features noticeable on the keyword constructs.

For an example, Graphplan (Blum & Furst, 1997) is a classical planning system that represents a problem state into a planning graph with a feature to obtain optimal plan and can perform a partial-order planning. Interface progression planner, IPP (Koehler, Nebel, Hoffmann, & Dimopoulos, 1997), is extended on the Graphplan while provides additional language features in ADL with ability to obtain optimal and the partial-order similarly. On the different approach, a method of converting a planning problem into a satisfaction problem then solved with another type of solver has been introduced in SatPlan (Kautz & Selman, 1992). With Satisfaction problem represented, the solution is optimal in theory but Barros and Musse (2005) reported that it may include a non-meaningful action in the plan. Next, by combining planning graph and satisfaction problem approach, local search for planning graph, LPG, is introduced (Gerevini & Serina, 2002). The most noticeable features from LPG is that it supports for action costs as numeric variable and provides a partial-order plan. In different to both planning graph and satisfaction problem, a heuristic search planner (HSP) (Geffner & Haslum, 2000) treats a planning problem as a traditional search problem using a domain independent heuristic. The result has not always been optimal unless an admissible heuristic function is defined. TLPlan (Bacchus & Kabanza, 2000) uses a temporal domain-knowledge to reduce a search space into the forward-chaining search system. With the hybrid features and novel techniques from Graphplan, IPP, and HSP techniques, a fast forward (FF) (Hoffmann, 2001) is a non-optimal planner supporting almost basic PDDL constructs except numeric and action costs while produces total-order plan. Another extension from FF was Metric-FF (Hoffmann, 2003) that supports numeric variable and works on maximizing or minimizing the given numeric parameters. In addition, Fast-downward planning (FD) (Helmert, 2006) was deploying different approach without using propositional PDDL but a multi-valued planning graph to support heuristic search for plan. Hierarchical task networking (HTN) is another classic technique solving for the plan by presenting initial state and a goal as single action task that must be decomposed into lower action iteratively then construct a plan. One extension using HTN approaches is simple hierarchical ordered planner (SHOP)

(Nau, Cao, Lotem, & Munoz-Avila, 1999) and its java implementation JSHOP2 (Ilghami, 2006) with domain-independent features.

By testing planners, Barros and Musse (2007) provides an evaluation on variety of existing planners in performance and with their compared features on give keywords important for interactive story telling such as *type hierarchy*, *equality operator*, *negative preconditions*, *conditional effects*, *existential precondition* in regard to their original simple scenario "*Ugh's Story*" (Barros & Musse, 2005). Moreover, El Rhalibi et al. (2012) provided an another evaluation of existing planners adapted for interactive story-telling benchmark with the same "Ugh's Story" scenario in comparison with Barros and Musse (2007) shown in Table 1. While testing planning computation time generate hardware-specific result varied due to different in test machine, their finding reports that JSHOP2 becomes the fastest planning system followed up with FF, Metric-FF, and Marvin respectively.

Table 1 Evaluation of AI planning Algorithms for Digital Interactive Storytelling Benchmarks (El Rhalibi et al., 2012)

Planning Algorithm	Barros' Results	El Rhalibi, Goudoulakis et al's Results
FF	0.015s	0.023s
Graphplan	0.138s	0.240s
HSP	0.031s	-
LPG-TD	0.169s	0.463s
Marvin	0.017s	0.040s
Metric-FF	0.018s	0.036s
SatPlan	0.247-0.512s	0.630s
STAN4	0.093s	-
JSHOP2	-	0.021s

Likewise, regarding to crisis management aspect, a problem of resource allocation and action selection can be converted into planning problem in PDDL. To be flexible, a domain and problem language syntax should rely on basis of PDDL2.1 as it has been widely facilitated by existing parser and planning systems. Moreover, by considering pipeline for integration of planning mechanism to the crisis simulation system, it has relied on authoring domain-specific file based on crisis knowledge, then autonomously generating a current problem file with environment information in PDDL format dynamically. The drawback of using AI planning approach is that the plan computation time can have bad scalability with an increasing entity amount such as agents, possible actions, existing problem objects. The simulation system usually aims to achieve a real-time interaction, which demands the capability to control a size of planning problem or else implementing a real-time plan-searching algorithm. The other issue is some planning systems have not

provided their binaries in every platform, and suffer accessibility to source codes. In the decision of developing planner algorithm from scratch or employing a solution from existing planner, the deployment of the existing system has become more favourable for rapid prototyping a serious game as it must be developed in parallel with other core components. In the next section, the aspect of constraint satisfaction problem has been mentioned since it is related to the AI planning implementation technique.

2.8 Constraint Satisfaction Problem (CSP)

Constrain satisfaction problem (CSP) consists of a finite number of variables where each variable can be assigned with a finite possible value. Constraint restricts the assignment of possible value to a variable instance and it can be abstracted as setting of a requirement property in role, action, event, and related possible description. The constraint satisfaction solution is an instantiating of all variables fulfilling all the requirement constraints. In possible setup, partial constraint satisfaction can also be applied to create an optional preferable solution. The worst-case complexity of CSP is exponential but the reasonable design can simplify the requirement and result in more simple computation (Yokoo, Durfee, Ishida, & Kuwabara, 1998).

The common computation approach to solve CSP is to perform a backtracking search and forward checking. In backtracking search, the system assigns a possible value into a selected variable to instantiate then test for the constraints. In the case that the test fail (i.e. Constraints are not satisfied), the search algorithm then assigns another candidate values until there is no possible value in the pool. After that, it backtracks into previous variable and changes the value assignment. The process repeats itself until the solution is found or fails with exhaust of possible assignments. The forward checking algorithm removes any value that violating current instantiated variable constraints from the possible value pool of unassigned variables. This approach reduces a search space for later variables by exploiting an inference process. An example problem of CSP is the graph-colouring problem. The objective is to find an assignment of colour for each node so that the adjacent nodes do not share the same colour. The representation of this problem in CSP can be defined as each node colour is a variable and the possible assignment colours become a domain of variable.

The AI planning problem can be converted into CSP then be solved for the possible assignment of actions to satisfy goal condition, states of the world represented as goal variable setup. Since crisis scenarios are also always consisting of constraints in resource allocation and responsivities, an integration of CSP models applied to enhance scenario resolution has a potential to provide better observation results. In the next section, the knowledge on modern game engine technology has been discussed in details with the suitability of choosing an existing product as a framework foundation.

2.9 Game Engine Technology

In recent years, several software development kit (SDK) dedicated to create games have been developed and introduced into open community. Rather than being dedicated for each individual company, some of them have been released freely or on discount for common education and commercial purposes. These software frameworks are commonly called game engine.

In the early stage of game development, a designer and software developer usually start from ground-up for every stage of their current game without or little an aspect of recusing similar modules as the computing performance requires a straightforward delicacy to maximize playable framerate.

As game popularity is increasing over decades, there is a pressure to satisfy an over-complexity and quality of game products to demanding clients. Hence, producing game from scratch is not an ideal solution to a newcomer or even a professional studio, they require a toolset that consists of assisting tool and already integrated development environment for developers. The game engine concept is largely developed for first-person shooting game during 1990s such as DOOM and Quake in the aspect that both game and engine are not fully layer-separated. To give a general definition, a game engine is a software designed for creation process and deployment of video game. However, the game engine is not a single software that its only tasks is to draw graphics on the screen but it consists of a collection of subcomponents interacting together to run a game (Nilson & Söderberg, 2007).

In general, there is no universal architecture for game engine but, commonly, subsystems for game engines can be identified with Physics, Render engine, Input, Audio, Scripting language, Networking, and game engine manager shown in .

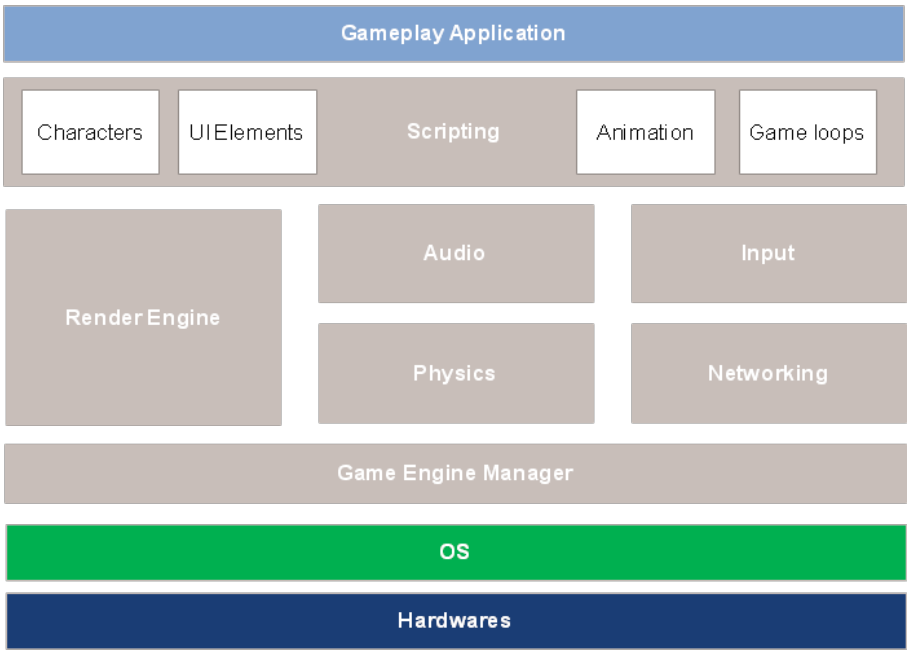


Figure 2-9 Game engine architecture

Physics, this system control how the physic mathematically calculated in gameplay and, sometimes, is not necessary to be applied to every object as it consumes high computation power. Rigid body physics is another subclass of the system that is the easiest to apply as it describes a solid object without deformation. In contrast, soft body physics such as cloth, water blob, and highly deformed object normally required to be simplified in an abstract bounding box of rigid body rather than fully calculated. In addition, this part of the system also includes a collision detection by performing variety of bounding box intersection algorithm. The purpose of physics in the game is not to fully simulate the complexity of real-world environment but to fake a perception of reality. The well-known limitation in this system is rounding problem occurring from not enough

precision. The small error from low precision calculation may built-up into a sub part of bigger entity and causes visible fluctuation.

Render engine, this is another dedicated system which provdes final image onto the screen. The rendering pipeline is a successive stage such as converting geometric information with its transformation matrix and virtual camera, and then finally generate 2D image result on the screen pixels. It works as an abstraction wrapper to computer graphical hardware layer by allowing a command to draw text, bitmaps, effect such as transparency and noise filter to be sent into lower abstract graphical API pipeline. For an example, the rendering pipelines that are default to variety of system such as Direct3D, OpenGL, OpenGL ES, and WebGL has been used in game engine for cross-platform deployment on Windows, Ubuntu, Android, IOS, and Web HTML5 platforms.

Input, this system reflects how the information from user interface hardware such as keyboard, mouse, joypad, or detecting motors can be detected and interpreted as a game control. The input hardware in the commercial market are normally abstracted their product interface with platform-independent library such as DirectX. The important aspect is that the user should have a control over how game applications map their command of input signal.

Audio, this system is similar to renderer and input as it acts as an abstraction layer to sound hardware I/O that exists in the computer or console. The wrapper library that is commonly used is OpenAL. A game developer has to prepare their audio in a common format then identify the instruction to play, stop, or loop over the audio API pipeline. Authorising an audio using game engine is not a common practice.

Scripting language, this is a soft architecture of the game engines where the hardware part and subcomponents abstraction is defined in separate layer. The scripting part allows game to be created as unique to the others produce from very same engine, thus it allow graphical content, reaction and behaviour of gameplay to be programmed systematically. Some engines provide extra features or allow modular middleware to be injected for assisting this creation process by adding a script template or visual code editor that compile part of the completed game separately. The another aspect of scripting component is that it allows reuses of common utility source codes such as pathfinding, data structure, artificial intelligence model and hardware communication.

Networking, this system handles a capability of multiplayer feature between different computers or consoles. A communication layer has to manage low-level transmission protocol in network layer. This system provides an abstract API for sending and receiving a key information to be synchronized or reported with another client while includes a controller to deal with signal lost or congested traffic delay using the communication layer seamlessly

Game engine manager, it is presented as a core of game engine applications by bridging a communication and execution flow of each subsystems. Its task also includes; management of memory, performing file input/output, and starting the game application on a deployed platform.

In the commercial and research, there are several game engine platforms available. Some are open-source and dedicated for both 2D and 3D application while include limitation in features or programming language. For an example, *GameMaker* Studio is originated from product called *Animo*, and have extensively support 2D game engineering. While 3D graphics and physics are partially support, it is not suitable tool due

to its limitation on 3D virtual camera control (Ciesla, 2017). *Cocos2D* is game engine supported 2D application highlighting a free feature for physics and visualization but some features require payment (Hussain, Gurung, & Jones, 2014). *Cocos2D* support C++, Lua, and JavaScript programming. *JMonkey* is a free game engine including cross-platform export. It is java-based engine and provide a 3D graphics-rendering pipeline (Reese & Johnson, 2015). *OGRE* is an Object-Oriented Graphics Rendering Engine, which is a fully multi-platform with OpenGL and Direct3D support (Rocha, Rocha, & Araújo, 2010). It is free open-source system developed with C++. It is capable of 2D and 3D rendering with export ability to mobile platform, iOS, android and Windows. Many education institutes deploy *Orge3D* engine in their study of game-engine architecture development. *Torque game engine* is released as free game engine originally developed from a Tribe engine by a company named Dynamix in 2001 (Maurina, 2006). It includes general scripting language, networking, and rendering pipeline to generate all-purposed games. *Shiva3D* was first released in July 2007, with an ability to deploy cross-platform over 20 systems (Shiva, 2018). The most famous product is Prince of Persia and Babel Rising. It provides Lua as a programming language with cross-compiler with C++ for optimization. *Unreal Engine 4 (UE4)* is another power engine released as free and open-source. It is one of best-popular game engine as many showcase game has been developed (UnrealEngine4, 2018). The main development language is C++ and being in-used with visual scripting system called blueprint allowing a rapid prototyping for game logic without actual C++ coding. *Unity3D* is one of publicly well-known game engine since it provides accessibility to advance 3D graphics and deployment-ability on cross-platform including mobile devices (Unity3D, 2018). The programming language is C# and JavaScript, which provides ease of learning with developer who are more familiar with object-oriented programming without manual memory management. While researchers can select any game engine system for developing their prototype, the available features and tools are an important aspect to be concerned in an early stage as it affects the time and effort for learning then deploying a final application.

Furthermore, there are several studies related to analysis and classification of features on available game engines (Christopoulou & Xinogalos, 2017; Cowan & Kapralos, 2014; Uskov & Sekar, 2014; V. B. Vasudevamurt & A. Uskov, 2015). In their study, the criteria to evaluate game engine are different based on no common framework but the important features for their final application. Interestingly, Pattrasitidecha (2014) has conducted an evaluation of game engines for 3D mobile platform with these metrics: programming libraries, usability in particular to ease a learning by tutorials and communities; Efficiency of use; available deployment platform; and price. Moreover, Christopoulou and Xinogalos (2017) uses these criteria with an additional evaluation framework for serious game engine from (Petridis et al., 2012) such as composability, developer toolkits, and audiovisual fidelity to provide further study in support of selecting a suitable game engine from their comparison.

The result from this research mentioned GameMaker as the least developed engine for audiovisual fidelity followed with JMonkey, Ogre3D, and Shiva while the most outstanding are both Unity3D and UE4. UE4 has many very strong features in global-illumination category in comparison to Unity3D. For functional fidelity including scripting language support, artificial intelligence libraries, and physics engine, UE4 and Unity3D can be observed as both supporting all features in the test with PhysX suitable for physical simulation, followed by Shiva while the rest like to not provide the features. In composability, which defines an ability to import/export content from Maya, blender3D, 3D studioMax indicated that only GameMaker lack this features. Next, for developer toolkits, Unity3D and GameMaker requires extra-fee for installation SDKs to

export content into iOS, Android, PS4 and XBOX game consoles while the others include the features automatically. In accessibility, the usability in an ease of learning from tutorials, examples, documents and community is studied along with their pricing program. The result indicated that Unity3D becomes the most usable engine with tis abundance of free tutorials, assets, and large community while its limitation is technical support is only with paid professional version unlike open-source system of UE4, JMonkey, Orge3D and Torque game engine which are completely free. The professional version of some commercial engine has approximately cost up to GameMaker - 299$; Unity3d - 395$; Shiva – 200$; and commission for UE4 after first 3000$ of the first year is 5%. In networking systems, only game engine called Turbulenz as an open-source system provides a peer-to-peer module while the others have a general client-server and multiplayer supports. Christopoulou and Xinogalos (2017) gives a summary: GameMaker is not suitable for 3D applications but provides a user-friendly for non-programmer designer while Ogre3D and JMonkey are best open-source game engine for learning game architecture environment but lack of portability and commercial features in rendering. In the most preferable choice, Unity and Unreal Engine4 are the most powerful game engines with all desirable features for game development but a final recommendation to select a suitable game engine is relied on the user expertise in programming language, knowledge to game engine system, budget, and the final-target platform.

In this project, we developed a crisis simulation software engineering framework with Unity3D as the engine has been evaluated with high fidelity in graphics, workflow editor, networking, programming language, and deployment option for both 2D/3D applications especially for network client-server games. Evidently from analysed result of game engine in serious game application (Vinay Bhargav Vasudevamurt & Alexander Uskov, 2015), Unity3D has an outstanding number of a being a developer choice for educational, simulation and virtual reality application are due to its capability in rendering SDK and variety of supported programming language such as C#, JavaScript, Boo. Moreover, asset stores provide a channel for purchasing license of framework scripts and 3D models, which can ease the rapid development of experimenting project. 3D Models can be imported in FBX format, Blender3D, and Maya source file. The community is quite active and there is plenty of learning material officially provided to a new user. Unity3D portability also being taken into to account as the project may be deployed on cross-platform in the future.

2.10 Summary

In this chapter, we introduce the background knowledge and keyword definitions in related to our research including a technology choice and common approaches for the crisis management and simulation applications.

First, the crisis definition can be ambiguous but in our research, it refers an infrequent event of emergency situations, which can have an origin from a natural phenomenon or human-made that can affect the individual, society, and environment. Management of crisis disaster involves a process of resource allocation whether it is a supply logistic and manpower assignment. In crisis preparedness, scenario has been introduced as a hypothetical situation allowing a practice of existing countermeasures and experiment with prospective new solutions. Crisis scenario often be separated into context, a setup background including an environment of situation, and crisis script that controls a logical narrative of important events.

As a game technology has become more advanced and accessible compared to the past, researchers can exploit an advantage of using a game development tool to study the emergency situation. An end product

using the game development toolkit in a purpose other than an entertainment has been called as a serious game application. While an application in the field of crisis management and simulation has been developed using features in game technology, the main issue is how to deliver a narrative content without exhaustively involve a domain expert. The solution partial to the issue is to procedurally generate a content in related to the crisis scenario setup whether a terrain, infrastructure, and events.

To study an outcome of crisis, using an agent as an autonomous entity capable of sensing an environment then interact with it or another agent with a suitable behaviour has become a main approach to simulate a real-world event with analytical result. By representing each agent as a firefighter, police, medical team or even a non-living concept such as hospital, a mitigation of specific disaster can be proposed and tested hypothetically in the virtual world simulation. Multi-agent system (MAS) has been investigated to describe a system that contains each agent as an independence individual capable of its own decision and priority. These agents can cooperatively take action toward a common goal or negatively neglect an unrelated request as necessary. This approach is suitable to apply into a crisis management simulation as it reflects that each stakeholder may have different objectives but, sometimes, able to cooperative in the limited sub-goals. However, a pitfall for using agent-based model (ABS) lies in the validity of the model itself as a simulation result which might not reflect a real situation thus produce a wrong assumption. Nevertheless, exploring the disaster outcome using agents have been an important process to early assess an impact and a feasibility of crisis plan.

As the scale of modern computation systems is increasing, it demands more in contents. Procedural content generation (PCG) is an approach that employs algorithms to automate producing a system data using configurable setup parameters. The challenge for PCG is how useful or accurate the approach can be integrated with a desired system with reduction to a trial-and-error on the control parameters. The PCG has a potential for a crisis simulation and management since it reduces the time-requirement to setup an early stage of desired environments. Then the draft environment can be finely edited by the human for a final detail to ensure the quality result. Interestingly, the content for narrative event can use a similar approach to produce an ongoing story sequence as well.

Cellular Automata (CA) is a topic study in an autonomous self-replicating and self-organization unit with an influence from the nearby neighbours. The common practice for a CA is that it represents an environment as a 2D grid while in each cell hold a state. The stimuli for a state-change behaviour can be triggered from an observation of a surround cell-neighbours. From this concept, CA has a potential for modelling an ecosystem or natural phenomenon on a terrain that describes the relationships between different adjacent areas.

To study for a crisis solution, AI planning is another technique from artificial intelligence study that it aims to describe a problem using a semantic abstract language then find a solution for a suitable sequence of actions from a starting state toward the goal. Majority of planning software support a standard planning language in PDDL version 2.1 but not all features of advanced version are supported in the planner. From our study, Fast Forward planner (FF) and FF-metrics are suitable for a fast prototyping due to its wide-range of PDDL feature support and being one of the fastest planner. The issue on using planner is a scalability since a larger problem requires longer time to execute planning so it may not be suitable to implement a real-time system using only the planning techniques. In an addition, Constraint Satisfaction Problem (CSP) is yet

another technique to search for an assignment of problem variables based on the predefined constraints. The CSP can be applied in AI planning for a crisis management since the resources and actions are generally constrained by different conditions.

Lastly, game engine technology is a collection of software development toolkits commonly used in creating an entertainment game application. Its architecture combines a library of physics engine, rendering framework, networking, sound audio interface, inputs, and, importantly, scripting language. By using a game engine, these foundation components provide a fasten development lifecycle and even let some scripts be reusable across different projects. While there are several existing products both commercial and free in the market, Unity3D and Unreal Game Engine have been standout from the rest in a quality and quantity of provided features. For our current study, the Unity3D has been selected as a based game engine to develop a proposed crisis simulation system due to its high fidelity in graphics, learning curve, and evidence support from a review report as it has been a top choice for a serious application.

In next section, the related literatures for crisis model, simulation framework and implementation technique is presented and discussed in each subsection.

3 RELATED LITERATURES

3.1 Introduction

In this chapter, it emphasizes on related works as a reference in the focus of constructing a generic crisis simulation framework. A related study on defining the crisis model has been presented and discussed for its suitability in the crisis simulation system. A relevant research applying procedural content generation to provide a geological setup is presented to demonstrate a research trend in automated content generation in crisis simulation domain. A research focusing on generation of crisis scenario autonomously with computational method is presented and discussed for advantage and disadvantage that might influence a system specification design. Lastly, Literatures on crisis simulation system are presented with discussion on the important provided features and their test-case results if existed.

3.2 Crisis model

There are several research aiming to giving a structure for crisis modelling. One of them is to formulate a mathematic model of a disaster then generate a scenario with variation on control parameters. By incorporating with the mathematic model, a crisis system can also simulate a disaster behaviour in a finite state transition with probabilities. Another aspect is to use a physic engine for modelling a situation. In this subsection, we introduces some example models imitating a behaviour of natural disasters such as fire, flood, pandemic outbreak, and other related situations. Table 2 shows a comparison of a model target, the test case scenario, and their techniques.

The wild-fire spread (Karafyllidis & Thanailakis, 1997) would be an early example of applying cellular automata (CA) to model a fire disaster. They explain the incorporated of factors affecting a forest fire that are weather condition; wind direction and speeds; the topography of landscape; and the area property resulting in different fire spreading rates. These factors as variable in CA‘s neighbouring algorithm allows generating a variety of different hypothetical scenarios on the environment which is mainly a homogenous and heterogeneous forest varying in fire spreading rate in this work. The simulation still lacks a further involvement of human infrastructure in the geography to describe the impact of the wildfire according to different city establishments.

For an another example of wild-fire simulation, Russo, Vakalis, and Siettos (2013) provides a simulation result of fire-front prediction scenario in Rhodes Island in Greece using CA technique. The research uses a four state representing fire status, which are no forest fuel; contained unignited forest fuel; contained fuel and burning; lastly, a burned down state. The influence from burning cell can ignite a fuelled cell with probability of burning calculated taking into account of vegetation density, vegetation type, landscape slope, and a wind direction. A result shows that the generated fire-front result can produce a similar visualization

comparable the empirical data from the forest fire of Rhodes in 2008 without any fine-tuning parameters. While the research does not take into account of firefighting strategy, it proves that CA is suitable to model a wild-fire situation effectively.

Lava flow simulation model SCIARA (Crisci, Rongo, Di Gregorio, & Spataro, 2004) has been implemented with cellular automata technique with applying a rule to introduce a movement of landslide lava path comparing to the real lava eruption at Etnean eruption in 1991-1993. The model aims to simulate the complexity of phenomenon by describing it in a local transformation of each lava cell using some local cell property such as height, temperature, and inflow-outflow calculation. In addition, the initial setup of global parameter has been applied since it ensure flexibility in configuration of specific event i.e. temperature at vent, CA's clock, Cell side length and physical characteristic such as cooling rate and lava adherence parameter. By comparing the simulation result corresponding to real event data, the research shows that using a hexagon-grid representation is more suitable to better approximate the lava or fluid flow path.

The flood simulation, Sabino and Rodrigues (2009) introduces a simulation of the flooding threat in dam break emergency management scenario. Cellular automata technique has been applied for water mass distribution with constraints for testing and validating agent operation in the simulation. The environments such as road and house are represented as a static segment in the layer of geographical features of the terrain. The model converts the flooding area into spatial representation (grid cell) along with different environment detail of landscape and water blockage for supporting a test of agent decision-making process.

An another example for flood simulation, Mao, Wang, Ni, Xi, and Wang (2017) investigates an impact from a dam-break flooding hazard using two simulation models, which are dam breach water flow and two-dimensional flood evolution model. The research provides an example scenario of the dam-break flood hazard using a setup of upstream reservoir located at the Pihe River in China. The visualization of flood-downstream is generated as a risk-management support material. Using a hydraulics simulation system, MIKE21 (Warren & Bach, 1992), the calculation for a flow of water has been provided by firstly defining a maximum scope of coverage hazard area. Then a grid with an arbitrary shape such as rectangular or triangle has been mapped to divide the terrain into a cell. Lastly, the calculation of water depth, flow direction, and a water velocity has been derived using a suitable hydraulics mathematical model. Considering a flood evolution technique, it is a cell-influencing concept from CA adapted to model a water dynamic characteristic. This work also provided a study result of damage evaluation in an area of affect region with a household and geological damage including human-life loss and health impact.

Epidemiological model, Zhong and Kim (2011) simulates the study of public risk communication toward a pandemic crisis modelled using the influenza outbreak 2009 H1/N1 in Arizona. Their study uses computer simulation to explore a flu epidemic situation progressing to hypothetical communication strategies formulating a crisis plan. The flu dynamics has been developed on a classic Susceptible-Exposed-Infected-Removed (SEIR) model, originated from the early concept from Kermack and McKendrick (1991). The model is predictive for a disease that can be transmitted from human to human by representing a number of human indicating in each state at a particular time. By the nature of different infectious disease, the population between these states can be fluctuated back and forth. In this scenario, the influenza patient from infected period can transit into a recovered state or result in casualty. This research improves on the current classic SIER model with an aspect of patient action during an outbreak called avoidance behaviour into control

parameters of state transition functions. The simulation represents a patient unit as a household then an infection can be spread on the neighbour unit or abstracted boundary area. This type of area state transitioning can be designed in CA rules for a spatial area representation incorporated with the mathematic state transition of SIER.

A slope-based intelligent 3D disaster simulation (Kim, Jung, Kim, & Chung, 2016) offers a simulation of natural and accident disasters such as landslide, rock fall, flood, fire, explosion and geological-object behaviour on a slope terrain using physic engines in 3D virtual world. The system uses a context-ontology interface to store an information of an environment, population, and related characteristic of the disaster nature to create an inference rule for the context-awareness simulation engine. The particle systems in a physic engine can be used to model variety of disaster situation shown from the example in this research. The benefits of using an inference engine is the rule from the context of an environment can use to predict an occurring accident or crisis. For an example, the system shown an example of a vehicle moving on a snowing slope with the vehicle weight and surface friction taking into account of the inference engine, the failure of surface on the snowing day is probable to occur. The simulation using physics engine is useful to observe the impact of disasters but it may not computational efficient for a prediction of hazardous spreading prediction on the regional terrain.

Moreover, some research aimed to model a road-traffic simulation using CA techniques. This situation can be considered as a minor crisis in the urban area. CA based traffic flow simulations create speed update as a function of location on the street network and obstacle in the adjacent cells. These models can be categorized as microscopic and are similar to car-to-car following system, as they take into account only the cell nearby the cell being processed. A very common technique known as the Nagel-Schreckenberg model (i.e. the NS model) (Nagel & Schreckenberg, 1992) and cantered on the Rule 184 (W. Li, 1987) belongs to the class of particle hopping systems. Rule 184 is central to traffic flow modelling with CA. The NS model can be seen as an enhancement of this technique with additional properties such as a discrete speed for vehicle, and an arbitrary propensity to reduce the vehicle speed (Gershenson & Rosenblueth, 2009). These simulations use integer state variables to represent the dynamic features of the system. Traffic flow modelling and simulation is a vast complex topic and the choice of the model to use can be challenging. By implementing CA technique, it provides advantage on this issue as it can produce a large variety of global traffic behaviours using local rules. Each cell of the CA can return individual features and slight alterations in the rules or the cells state can generate important changes (Benjamin, Johnson, & Hui, 1996).

Table 2 Crisis models and their techniques

Model	Test Scenario	Technique
Wild-fire (Karafyllidis & Thanailakis, 1997)	Variation of Hypothetical forest with same and different burning rate	CA with a state that indicates burning status of an area. The local rules infers how the neighbour can receive a burning influence with in a limited time step

Wild-fire (Russo et al., 2013)	Fire-front prediction in Rhodes Island in Greece 2008	CA with a state to define a fuel of vegetation and burning states with transitional burning probability
Lava flow (Crisci et al., 2004)	Lava eruption at Etnean in 1991-1993	CA for simulate a flow movement of lava transformation regarding temperature, terrain height and cooling rate
Flood (Sabino & Rodrigues, 2009)	Aqueva dam, in Portugal.	GIS map as a layers of geographic features to be use in agent decision-making. CA is used to simulate a flood threat over a map
Flood (Mao et al., 2017)	Dam-break on reservoir in Pihe River in China	Dam-leak flow model and using MIKE21 as a tool for a hydraulics simulation (mapping an area into a polygon cell grid as similar to a CA technique)
Epidemiological (Zhong & Kim, 2011)	Influenza outbreak H1/N1 in 2009 in Arizona	SEIR model for influenza outbreak describing a population health state regarding the transition of epidemic evolution. The transition model uses an abstract of one civilian
Traffic (Benjamin et al., 1996)	Hypothetical highway with a vehicle traffic regarding a benefit of deploying a traffic junction	CA model using a ruleset from Nagel and Schrekenberg model with additional rule introducing a "*slow-to-start*" aspect reflecting a real driving behaviour
Multiple disasters (Kim et al., 2016)	A hypothetical landscapes with slope and a variety of disaster situations	Context-sensitive ontology inference for disaster rule and simulation. The particle system has been implemented to create a virtual situation of physical-based phenomenon

From the reviews, crisis simulation framework shared an issue of how to produce effectively a general representation of crisis and its natural behaviour regarding of terrain geography representation. By representing or applying a grid representation to the area, using cellular automata technique can simplify the problem of natural relationship between each neighbouring area of impacted location into a set of rules. While defining a rule incorporating an environmental resource, establishment placement, and global factor of unspecific whether variable, the CA model can be designed for a variety of disaster situations with its own dedicated mathematic models for influencing neighbour. In the contrary, while the physical simulation for the crisis can be benefit to the modelling as the physic models have been well-studied with the knowledge of real-world engineering, it often introduces a computational complexity issue in a large-scale simulation over different region. In fact, the trade-off in computation often causes the modern game application substitute the physic model with an inaccurate approximation but cheaper in computation. For the virtual training, the physic model may be preferable technique since it gives a look and feel of actual environment interaction. But the cellular automata technique has been commonly observed being applied in the simulation of strategic impact of large-scale disaster.

3.3 Automated Content Generation for Geographical Scene

Modern serious games requires 3D models as a main representation content. There are also research, which focus on procedural generation of such content. Related researchers have tackled procedurally generated game environments with varying degrees of success. The most well known and most significant research has been carried out on the CityEngine (Bekins & Aliaga, 2005), a system that is capable of producing realistic and detailed models. The generation algorithms are inspired by the modelling of natural phenomena in string grammars (Greuter, Parker, Stewart, & Leach, 2003) and L-systems are used to construct road networks and buildings. Procedural building generation has focused on the application of grammars to describe structure, like the Shape Grammars original proposed by Johnson et al. (2010). These have been applied in various guises to the construction of building geometry (Lechner et al., 2004).

Müller, Wonka, Haegler, Ulmer, and Van Gool (2006) defined a specific grammar to characterize building structure. This system defines rules to operate on shapes, called shape grammars. It can be used to design a range of detailed buildings in several architectural styles. Recently this approach has been extended to employ imaging techniques to aid in the acquisition of generation rules from existing building façades (Müller, Zeng, Wonka, & Van Gool, 2007; Parish & Müller, 2001). Other approaches include the application of intelligent agents (Prusinkiewicz & Lindenmayer, 1990), real-time frustum filling (Stiny, 1980) and template based generation (Sun, Yu, Baciu, & Green, 2002). In the same direction, (Prusinkiewicz & Lindenmayer, 1990; Wonka, Wimmer, Sillion, & Ribarsky, 2003) propose to simulate the evolution of cities by modelling land use and evolving a city usage map over time that can be used later to create a cityscape. Real-time city generation has also been attempted. Furthermore, (Stiny, 1980; Wonka et al., 2003) have implemented a city generation system that fills the view frustum rapidly with buildings of various shape combinations but their placement is restricted to a road network consisting of a regular grid. Finally, (Sun et al., 2002) proposed the application of templates that encapsulate patterns such as raster radial to generate road networks.

Compared to these techniques our approach introduces novel aspects including as using of agent-based approach for the game world generation, interoperable solution development using Unity3D that is allowing deployment across many platforms including web browsers, and multiple OS. We provide instant in-game generation, rendering and recreation of 3D hexagonal cell-based with organic environments, agent AI Bots to interact with the environment after the generation stage according to other user-set constraints, and structural constraints induced by the game world and crisis scenario generation. Our prototype framework also allows full control of the environment generation by the user, as well as generation options constrained by the crisis generation environment.

3.4 Crisis Scenario Generation

Similarly, procedural generation technique can be applied for the purpose crisis scenario generation since a plausible scenario is often authored by domain experts, which is very time-consuming process. There are two benefits to traditional handcrafted process. The former, Content generation can quickly produce an on demand scenario with setup constraints. The latter, computer generated scenario can be used to supplement human-based scenario quality. In this section, we focus on a crisis management domain. The result scenario have

been observed to aim for producing a sequence of crisis events which emphasized on resource management perspective and stakeholder collaboration in solving emergency. Table 3 descrbies the comparison between crisis scenario generation technique and their drawbacks.

Grois, Hsu, Voloshin, and Wilkins (1998) develops a *SceneGen* algorithm for Navy DCTrain System using Noisy Bayesian Network (NOBNs) to search a sequence of key-event to satisfy a scenario objective empowered by data from knowledge-based in a form of belief network with a penalized likelihood and rejection test to filter non-plausible results. The process aims to provide offline-scenario generation. The authors also mention a content generation technique called Case-based stochastic perturbation (CBSP) that the process is to acquire a seed scenario from experts then apply random distribution to manipulate more variation to the original, which is likely to bring unreasonable or in-plausible result due to random nature without any testing for plausibility. Comparatively, ScenGen's strength lies in its ability to guarantee the "quality" of each and every scenario it generates through a carefully designed selection bias and it claims to be better in overall performance than Manual design by human subject matter experts (MDHE), Naïve random generation (NRG), and Case-based stochastic perturbation (CBSP). The plausible scenario is defined by checking an occurrence of key-events according to a target of learning objective. The learning objectives can result in similar key-events that can be added simultaneously at any time step, thus it provides variation aspect in results. The major drawback of this approach is to require a set of good base human generated scenario as a seed then manipulates them to obtain more variation in the automated results, and, in addition, the offline generation may lack efficiency in dynamic crisis simulation system when the setting is reflecting the complexity situation from real-world problem.

Hullett and Mateas (2009) applies AI planning technique to generate a firefighter rescue-training scenario in the collapsing building area. Their system uses HTNPlanner, which is one variation of AI planning techniques, with building structure data as input. Then it set a goal to create a situation that satisfied crisis final descriptions while a domain knowledge must be defined for AI planner. This domain knowledge contains predicates and actions describing a collapsed building during fire incident to allow physical consistency, thus achieve better plausible solution plan. As a result, the system generates a scenario by filling in content as a sequence of event or activities that are expected to occur and which usually manipulate world environment leading from initial state to the desired goal situation. Domain describe trainees, firefighters, to have a given a role and a set of action-specific skills to perform in the scenario. The variety of generated scenario is delivered in scale of small, medium, and large world setting measured by scenario maximum length in step, and it is argued to be better than random probabilistic distribution of element in case of fire situation and damage propagation due to the usage of well-defined domain knowledge. The main limitation of this system is that the work is tending to encounter a memory shortage during simulating the variation of levels in their proposed medium and large scales.

Martin et al. (2009) proposes an automated scenario generation system that aims to be generic and applicable to any domain specific simulation system by developing a rule for Functional L-system (FL-system) in crisis scenario domain. The authors implement a conceptual mapping approach that is based on 1) training objective parameters; 2) baseline scenario as a pre-defined ideal parameter scenario seed; and 3) scenario vignettes that is a meta-complexity modification of scenario such as weather, light, and related background setup. The main objective and additional data such as weather condition (vignette) setting are composed in an

XML file and generate the scenario variation using scripted FL-system with the syntax to represent scenario elements. This system allows generic automated domain-independent scenario generation with different simulation framework. However, the disadvantage is that it requires extra work on developing a rule for FL-System with can be one-time requirement for new training domain and it is reusable in similar domain setting. The difficulty on generating a rule also become a trial-and-error of learning curve since the rule is often abstract and easily become too complex.

Zook et al. (2012) introduces a combinatorial optimization approach in genetic utility function aiming to deliver the requisite diversity and quality of practice scenarios while tailoring the scenarios to a learner parametric model in military training. The proposed technique can be considered as oppose to AI planning approach as it provides fuzziness probability in matching up key-events for final objective goal set by designer while AI planning often observed as a searching for pattern-like structure in reasoning for suitable events fulfilling final goal condition. In implementation, the main scenario generation bases on a genetic algorithm to search for a best solution; reading in author-specified domain knowledge, the details and the type of possible events and requirement in scenario, and constraint order on events. The process works by considering instant event template for the scenario at random location; mutating the parameter of random chosen scenario; Applying crossover operation, to create new sequence of events to improve the quality of scenario for the next iteration. The authors propose evaluation in an interesting and effective way: 1) quality of solution at run-time; 2.) the diversity of scenario as function of running time, 3) performance of trainee and appropriateness of difficulty level when training on generated scenario. The results from their evaluation are to generate a unique scenario compared to planner generation approach, this technique bases on a combinatorial optimization provides lower-quality solution initially, but explores multiple different regions in the solution space containing high-quality solutions and so refines multiple distinct scenario that meet provided learning objectives rather than explore variation on same high-quality scenario. In practice, AI planner approach is expected to yield a high-quality tailored scenario early and produce several scenarios of roughly equivalent quality that are very similar on a several high-quality scenarios. The authors suggest the requirement of virtual technology for scenario training is vital to incorporates learner attributes and theoretically, lead to more effective training as learner have greater opportunities to train more on relevant scenario. The major drawback is that the generation requires a predefined small element of events (seed independent events) to be tailored into scenario and initial input of learner model for suitable evaluation fitness function for the genetic algorithm.

Table 3 Crisis Scenario Generation Techniques

Authors	Technique	Benefit	Drawback
Grois, Hsu et al. (1998)	Noisy Bayesian Network (NOBNs) to search a sequence of key-event to satisfy a scenario objective	Guarantee the "quality" of each and every scenario it generates through a selection bias and it claims to be better in overall performance than Manual	Offline and require a set of good base human generated scenario as a seed then manipulates them to obtain more variation

Hullett and Mateas (2009)	HTNPlanner, which is one variation of AI planning technique to tailor the scenario environment setup	The variety of generated scenario is delivered in scale of small, medium, and large world setting measured by scenario maximum length in step, and it is argued to be better than seed-based probabilistic approach	The main limitation of this system is that the work is tending to encounter a memory shortage during medium to large simulating
Martin, Schatz et al. (2009)	Functional L-system (FL-system) to define a rule for crisis scenario event generation	generic automated domain-independent scenario generation with different simulation framework	The difficulty on generating a rule also become a trial-and-error of learning curve since the rule is often abstract and easily become too complex
Zook, Lee-Urban et al. (2012)	Combinatorial optimization approach in genetic algorithm	generate a unique scenario compared to planner approach while claimed to have a lower quality at first iteration but better with variation in later loop	The generation requires a predefined small element of events (seed independent events) to be tailored into scenario and initial input of learner model

Overall, different computational approaches have been proposed for scenario generation system. From the reviews, the combinatorial optimization search and AI planning techniques are efficient to produce a variety of quality scenario. The former is generic optimization approach working by evaluating a set of function to determine a necessary event elements of given scenario. It works best for a training aspect since these parameters such as scenario length, constrained sequence structure, and a learner model are available to achieve a relevant and distinct result each time while the drawback is laid in its requirement in pre-computation of event matching resulting in difficulty to deliver real-time solution. The latter is deploying AI planner to fill in between different key events based on the event precondition and constraints. The planner approach is effective to design and control since the knowledge representation is concise using a formal planning domain definition language such as PDDL. The issue on a lack of result variation by using planner is still arguable since there are several example system in digital story telling generation successfully applying such a method to order a story priori based on given setting element of dramatic arc (Porteous et al., 2010). While the seed approaches that blended a human-created element and automated searching process seem to be more effective in generating high-quality result, the lack of variation and time-consuming aspect make this technique less preferable in representing a general crisis simulation framework. For a heuristic approach, the creating of scenario is formed by evaluating a subcomponent of scenario to the given heuristic function. The given subcomponent has been kept in a final scenario if it passes the evaluation threshold. Although this method seems to deliver an effective scenario, it is computing inefficient as it might have to retry combining scenario from every possible events.

Although, there is no distinct best solution on how to generate crisis scenarios, there are desirable components that the crisis automated scenario generation system should provide. First is to allow scenario generation to operate with different simulation platform using a general data representation such as XML and JSON. The cross-platform features allow ease of framework integration into other crisis management systems. Second is an ability to ease of integrating a knowledge of simulation models such as fire, flood, epidemic,

tornado, and earthquake as parameters in equation that would be sufficient to represent the environment of complex real-world problem. This aspect also involves a simulation of human-made crisis such as terrorist, traffic accident and related situations. The tool to ensure the latter aspect is to have a flexible agent architecture that can be modelled into any type of person behaviours. Last is to allow interaction with user during a generation process and provide a configurable model to estimated evaluation of crisis impact of scenario.

For our framework, we employs the AI planning approach due to its simplicity in design for knowledge representation of crisis scenario rather than developing a learner model commonly seen in genetic algorithm approach. We focus our improvement design to develop a representation of agent in PDDL and, thus, it allows us to investigate further a possible outcome of their collaboration in different situations using AI planning in the genetic crisis simulation system.

3.5 Large-scale Crisis Simulation

Since the emergency response process has been described in these four stages: prepardness, response, recovery, and mitigation. Crisis management has two main purpose of averting an upcoming emergency and to control the boundary of ongoing incident. The scale of emergency stiuation can be varied starting from local community incident upto a sesrious and naional catastrope. Gad-el-Hak (2009) gaves a suggestion on classification of disaster severity by using a size in square-meters on geographical area and '*impact'* on both environment and human severity as shown in Table 4. Hence, simulating a crisis scenario can be generally separated into large-scale emergency simulations and smaller local-scope systems. While the small scope of crisis scenario can be better employed a physic virtual simulation, the larger scale crisis require a computational effective model to approximate an ongoing incident with its influence over multiple regions.

Scope I	**Scope II**	**Scope IIII**	**Scope IV**	**Scope V**
Small Disaster	Medium Disaster	Large Disaster	Enormous Disaster	Gargantuan Disaster
< 10 persons	10-100 persons	100-1000 persons	1000– 10,000 persons	>10,000 persons
< 1 km^2	1-10 km^2	10-100 km^2	100-1000 km^2	>1000 km^2

Table 4 Classification of disaster severity (Gad-el-Hak, 2009)

The method of scenario simulation can be performed physically and virtually using computer system. The computer system provides an ability to create a hypothetical situation that could be expensive and physically dangerous. Jain and McLean (2003) gave five main purposes of computer simulation into emergency response applications: 1.) Planning, 2.) Vulnerability analysis, 3.) Identification, 4.) Detection, and 5.) Real-time system support. These applications have a different purpose start from study an impact of event, evaluating hypothesis response, determining a possibility of occurrence, finally, an integration into real-world knowledge base system. In contrast, by the limitation of real-world practice drill, (X. Li & Song, 2018) describes these issues as: 1.) the practice drill is enforced to follow a script so it becomes inflexibility and cannot reflect a fast-pace characteristic of crisis; 2.) Cost for the practice drill is expensive in preparation time

and resources; 3.) Complexity to coordinate with necessary authorities to setup a site of training; lastly, impact of the drill may not catch the attention of the participants after the procedure is finished.

By focusing for the computer simulation, an agent-based modelling simulation (ABMS) has been used consistently with emergency response simulation of real-world disasters. It best suited for modelling on constrained situation of environments and stakeholders attack on transport, attack on crowded place, pandemic human disease and natural disaster such as flooding (Challenger, Clegg, Robinson, & Leigh, 2009). North and Macal (2007) also discussed the advantage of using such a system that it captures the emergent phenomenon; provide a natural description of the system; flexible; and, thus, can be a solution to study an emergency situation.

An agent is often considered as representation of human individual capable with a set of available skills involved in the domain, typically either civilian or rescuer, but sometimes it also represents non-human entity such as vehicles and building. Consequently, design of an intelligent agent is usually aiming to have proactive mind set aiming to achieve some role-specific goal, and it is reactive in response of change in the environment (Wooldridge & Jennings, 1995). The main advantage of using such a model is an ease of observance and flexibility to assign different kind of behaviour for appropriate entities, which regularly represent civilian, rescuer, decision-making department and include non-living entity such as city, vehicle, and building (Wonka et al., 2003; Wooldridge & Jennings, 1995). The decision-making of agents usually implemented based on heuristic rule, finite state machine, utility function, and a decision tree. While a real-time AI planning technique can be useful, the time complexity is a major concern for a real-time application so a hybrid method or problem simplification may be necessary for an increasing object of interests. The agent parameters often depend on type of the simulation.

For the representation of map environment in crisis simulation, the simplest method is to represent it as a grid by dividing an area into a number of abstract cells. It gives a fast perception of position and space equally in every direction. Next, using a vector geographic information system (GIS) is suitable for representing a real-space area as it complies of building information, road network, and other related detail data. These information file sizes are usually large and being required a pre-processing to filter out unnecessary information from the original package such as extracting a real-building polygonal shapes. Lastly, the graph-network is a non-uniform version of grid which each cell or node is representing a connection between key areas of interest. The graph-network is suitable to use in hybrid with both previous approaches since it can represent a higher-layer of data on top of geographical map. The additional concern is that if an incident caused ecological or geographical changes into the environment, the visual presentation should adapt to reflect the situation. The graph network may not be sufficient in the dynamic change of environment but it can be suitable for otherwise such as epidemic outbreak.

A scale of implementation for a large-scale emergency response normally includes greater than a hundred agents within a virtual environment of at least ten-square kilometres (Hawe, Coates, Wilson, & Crouch, 2012). A simulation result also receives an influence from both a number of intelligent units and a map representation detail (Sato & Takahashi, 2010), and it can become a cause for performance complexity. The issue with scalability problem require further study about a trade-off in the level of detail to be adequate for the different scopes of simulation. For a real-time system, the representation should be abstracted thus being grouped up model information into a connecting node of interest using grid or graph-network with

internal model of area control parameters. Otherwise, one such a solution is to address the problem using distributed computation with a shared memory over a cluster systems (Koto & Takeuchi, 2003).

For related systems, one goal of using simulation system is to emulate how the events in the real world unfold and then determine the appropriate optimal response plan for the agents. Several existing systems work around this concept using agent-based simulation (ABS) for large-scale emergency response (Hawe et al., 2012).

Kitano et al. (1999) introduces a Robocup Rescue as a multi-agent simulator system to promote a research of disaster rescue agents. Takeuchi (2004) demonstrated its simulation scenario that is using the 1995 Kobe earthquake as the original test case. The system is aiming to represent the disaster situation sensory information then to incorporate the agent-simulation system to mitigate disaster and encourage large-scale research collaboration by holding an annual competition since 2001. Terrain data is the framework often being delivered in grid spatial 2D representation to simplify road network, obstacles, strategic resource, and civilian positions. The agent response is based on optimization of the design and implementation of better action selection method to maximize the objective function regarding the number of individual, proportion of remaining health to initial health point, and proportion of unburnt area of building. In addition, with the nature of multi-objective in crisis response, a vector-based score has been proposed to compare different responses strategies (Siddhartha, Sarika, & Karlapalem, 2009).

The AROUND (autonomous robots for observation of urban network after disasters) (Boucher et al., 2009) introduces the use of agent-based model to predict the outcomes of possible course of action such as an ambulance response by human rescue teams. The goal of the project is to capture mathematically the agent behaviours for the rescue objective using a learning session with experts to form a parameterized utility function. The result is an optimization of agent selecting a suitable behaviour mimicking a real-world knowledge of responder. The test scenario being used is modelled after an earthquake in the city of Hanoi (Vietnam) from 1935 and 1983. The main difference from the Robocup Rescue is that the agent action selection in AROUND aims to reflect a real-world personal using parametrizing while the latter implementing action selection logic as an agent decision-making logic with optimization on outcome multiple-objective parameters.

PLAN-C (planning with large agent-networks against catastrophes) is an ABS developed to predict the behaviour of individual and collectively in large-scale emergency such as terrorist attack (Mysore, Narzisi, & Mishra, 2006; Narzisi, Mysore, & Mishra, 2006). The simulation scenario is modelled after a possible terrorist attack, which simulates the first 50 hours following a Sarin chemical attack on Manhattan Island. The system represents an area of incident using a spatial GIS map data pre-processed into a graph network with constraints for sub-area transitioning. The emergency response is represented as finding a solution to a multi-objective optimization problem (MOOP) since there is no best single optimum solution in the nature of disaster as the Pareto-optimal trade-off between different parameters is being discovered. An example of the parameters being observed are hospital resource levels (consumable such as drug and medicine; recoverable such as bed and personals). The result from their case scenario is that increasing the number of resources will rapidly reduce the fatalities; however, up on passing some thresholds, the fatalities will not lessen even with more resources available. Furthermore, the system provided an analysis result of civilian behaviour on

attending the nearby hospital regarding the effectiveness of hospital to admit and release the appropriate number of patient to reduce fatalities.

EpiSimS (Barrett, Eubank, & Smith, 2005; Mniszewski, Del Valle, Stroud, Riese, & Sydoriak, 2008) is another system deploying an ABS to study the optimized parameter of resource and procedures in a simulation of SmallPox attack model and influenza outbreak. The analysis results from the simulation is that the rate of death is directly influenced by how quickly the infected patients can isolate themselves from the rest of population (society). Whereas, the study of deploying an official response such as mass evacuation, vaccination and quarantine to control the spread of smallpox is less effective on the rate of death.

IMAMCR (intelligent multi-agent model for crisis response) has been developed to determine an agent optimal response to a pandemic flu in Egypt (Khalil, Abdel-Aziz, Nazmy, & Salem, 2009). The environment representation is based on GISToolkit and developing an agent system based on Cougarr Agent Architecture.

Saoud, Mena, Dugdale, Pavard, and Ahmed (2006) describe a multi-agent model approach for modelling a simulation (SimGenis) to design optimal, efficient, and appropriate rescue strategies, based on the initial state of victims, number of rescuers, and method of communication between rescuers (electronic or paper). They also represent a geography of situation in 2D abstract position of building and unit placements. More precisely, the aim of the research was to determine how the response to a dynamic large-scale emergency depends on the use of a centralized and decentralized collaborative rescue strategy with applying heuristic algorithm on each agent and component of the simulator while testing using seven configurations with 300 agents for victims and rescuers. The optimal response for the simulation is derived from the total number of victims, initial state of the victims and the total number of rescuer (doctor, firefighter, and nurse). The results from the study states that there is no best unique rescue scenario and it is hard to predict depending on the disaster characteristic.

Schoenharl and Madey (2011) introduced a simulation of a multi-agent system WIPER (wireless integrated phone-based emergency response) corresponding to real GIS visualizing geographic terrain in 2D spatial representation with each mobile phone users as an active node having a movement activities tracked by tower cellular segments in the area for a real-time response. A real-time data source provides real-time data regarding cell-phone usage from cell-phone providers. Using a historical data source (a repository of normal cell-phone usage), a detection, and alert system detect possible anomalies in cell-phone usage patterns. The system also operates by taking a batch of different agent movement activities for simulation aiming to mimic the crisis event. The example of such movement are non-guidance flee event where every agent moves away in disturbance, a flock event where agent is trying to move as a mob (grouping), and a jam event where each agent is moving toward their specific goal but is constrained as in traffic jam. Their system can simulate a density of agents as population and measure traffic activity, which represents the actual cell segment for a better understanding of the situation.

Metello, Casanova, and de Carvalho (2008) designs an emergency simulation game based on serious game technique. It aims to represent observable ongoing crisis rather than just a representation of crisis situation in the learning game. Their research introduces a guideline for a content preparation involving pre-incident brief, media response script, constrain and interaction between stakeholders as a context. Furthermore, the system helps to demonstrate a method of how an emergency plan can be tested in finding a

flawed procedure during any given scenario such as oil leakage simulated by cellular automata model. Thus, allowing the trainee to interact with a simulation system which events has been fired to the world environment over time and providing a feedback by rendering the effect on the screen.

3.6 Summary

In this chapter, we study the existing works on crisis model, automated content generation, crisis scenario generation and, finally, a simulation system for large-scale emergency.

From reviews, these researches provided insight on possible core components of general crisis simulation framework. Firstly, a main discussion is how to choose a suitable representation of crisis and corresponding geology with resources location including facilities and specialized crisis agent. Next, it is detailed on an integration of agent-based model to replicate response and method of collaboration between stakeholders, which can be individual citizens, responsible governing unit, specific department and personal using AI planning or rule-based approach. Then a simulation model of crisis event is a necessary component to demonstrate a relationship of agent's action and its outcome for finding an evaluation of assumption behaviour or countermeasure strategy. Several literatures attempted to address the natural phenomenon and epidemics using cellular automata technique since it is flexible to be integrated with a different type of map representation and emergency situations. Moreover, integration of automated content generation on both terrain visualization and crisis scenario are allowing efficient delivery of final application to a user. The test scenario usually models after the large-scale crisis occurred in the local region of the project rather than local practice of equipment. In comparison to the serious game applications, the large-scale systems have deployed for study the optimization of resource and procedure while the serious game approach would focus on a user interaction upon a real-time interactive system alongside with scenario simulation.

So far, in the available literatures, there are several similarities in the development of a crisis simulation framework regarding the preparation stage. The majority of the simulation system aim at representing the real-word knowledge then reproduces the crisis scenario based on historical records for devising a better decision support system and plan. Ultimately, it facilitates training the crisis personal for any upcoming event for a proper response.

During Real-time crisis, simulation framework such as WIPER applies the mobile network GIS information system to provide a realistic and real-time response for decision support simulation. Also for the preparation of study of crisis events by varying the different resource level, system such as PLAN-C can provide an analytical study of improved response on investigating some component of behaviour. In addition, a new behaviour, which is not constrained to be human resemblance, can be investigated as to maximize the evaluation function in Robocup Rescue. The similarities are that each simulation system is deployed using the agent-based model framework to simulate the behaviour of individual such as victim and crisis personal (e.g. firefighter, doctor) then measure the parameter of objective function such as fatality rate as utility. The algorithm is often based on optimization of the objective function to derive a best trade-off practice of parameters on resource and procedures. The representation of world environment is often in a form of grid of cells such as in SimGenis and PLAN-C while there is an argument on the cost of computation to enable GIS

information displayed in real-time multi-disaster or complex simulation model. The compromised has been observed as an offline pre-processed of GIS reference data transforming into simplify grid representation.

As for the persisting issue from the literatures, the development of crisis framework still requires a combined knowledge of multi-disciplinary technique such as environment presentation, crisis modelling, response simulation with agent architecture. There is not yet a generic system that eases a development of crisis disasters with flexibility to accommodate an advancing technology in every component. In addition, the real-time interaction with the user has not been usually mentioned in the existing systems. While some simulation system aims to generate a result offline, the ability to give a direction or changing the overall hypothesis experimentally during the simulation is necessary. It allows the user to explore a possibility that is locked in the preliminary design without repeating a full time-consuming simulation just to find out that it is not showing any promising result. Lastly, the development of crisis framework from modern game technologies are not a common practice. While the modern game technologies provide a suitable starting tools, it is often got neglect as a fast prototyping architecture to consolidate multi-disciplinary knowledge with already built-in of advancing technology such as Graphics render engine, networking, asset management, and, more importantly, an ability to deploy cross-platform.

In the next chapter, it gives a specification of the proposed framework aiming to deliver a novel system for generic crisis scenario design and agent simulation by developing on game engine technology.

4 FRAMEWORK SPECIFICATION

4.1 Introduction

In this chapter, we describe the specification for our proposed framework which are mainly classified into 3 main aspects: an automated environment representation; a crisis scenario generation; an agent architecture including AI planning. An overview of system architecture with high-level components is presented showing its main components' functionality.

4.2 Specifications

From the investigation of related background knowledge and literatures, a development of game-based crisis simulation system can provide advantages for crisis understanding and preparedness. Game technology has been primarily a tool for an entertainment rather than a simulation and training purpose. To incorporate it into a field of serious game, an additional functionality must be designed to integrate state of the art techniques in crisis simulation system such as dedicated agent-based modelling architecture, AI planning, reusable data interface for inter-application connectivity, and an automated map representation with highly flexible for developing a crisis model and visualization from a given hypothesis.

By aiming to overcome a challenge in consolidating different knowledge into a software for crisis scenario generation and simulation, we propose to design and develop a generic framework on top of a game technology called crisis game for scenario design and agent modelling simulation (CGSA-SIM). CGSA-SIM shall facilitate a setting up crisis model, generating a crisis scenario with automated terrain generation functionality, integrating of an agent-based model and AI planning for incident resolution using a modern planner, and, importantly, allowing cross-platform accessibility using standard data format.

The system specification of CGSA-SIM has been discussed by taking into account of benefit and drawbacks of previous related literatures and our research objectives. It focused on consolidating multi-disciplinary knowledge and technology into a unified software system usable by both crisis scenario designer and practicing personal. Then, finally, it has been proposed to deliver a generic game-based crisis simulation prototype including three main aspects: the environment presentation and visualization; the crisis scenario generation and simulation; the agent modelling with AI planning.

4.2.1 The environment presentation and visualization

The development workflow should use a game engine for a provided benefit of existing graphic engine, scene and asset editor, networking, scripting, and deployment. Unity3D would be our main development system for its high fidelity in performance and asset management pipeline. The component-based scripting in game-engine would provide a modularity in developing each key logic, thus ease a process of future extension.

The map representation should be a grid graph with hexagonal shapes. The hexagonal polygon has an equal distance between each region, thus become a flexible representation of landscape. The scope of one hexagonal cell should represent a residing quantity of environment features, agents, and resources. This decision would balance a system performance for visualization and simulation features.

The visualization of system should be in 3D. The immersive metaphor of real-world situation can be achieved using appropriated visual decoration models. Rather than a first-person perception, a classical board-game design from top-view would provide ease of understanding the general information of terrain, placement of resources, and crisis.

In addition, an ability for the user to interact and train with the system in real-time would become an important aspect of serious game application. CGSA-SIM should provide an interactive interface for 3D terrain setup based on key environmental parameters. Its feature would include functions for agent instantiation, agent parameters configuration, and controlling mechanism for simulator parameters. Prospectively, the scenario designer and trainee could use the same software client for testing the study of hypothesis in crisis incident.

4.2.2 The crisis scenario generation and simulation

Procedural generation technique should be deployed on the terrain generation with associated cell parameters. The changing of key environmental quantity would trigger fast visual adjustment in real-time. The scenario designer can work on overview of geological perception that is reflected to a simulation parameter.

A crisis-simulation model shall be design using cellular automata technique as a base environment representation due to it represent a straightforward relationship between area parameters with cascading effect of influences. A CGSA-SIM shall provide an accessible starting class to design any natural large-scaled disasters such as flood, wildfire, landslide, and other area-affected situations.

The time representation shall base on a trigger unit of turn step to provide an overall observation during each scenario development. In addition, an alternative outcome of the current model can be tested using an editor to alter map environmental information or agents before continue the next step.

To accommodate a crisis scenario scripting, the data structure to design an event sequence shall be provided with a suitable data parsing using JSON format for interchangeability between other systems. These events would be reusable for a different map to fasten a scenario prototyping.

4.2.3 The agent modelling with AI planning

An agent-based model should be deployed to represent any participant during the disaster. Each agent would have its own set of goal-objective and resources. The framework should provide a built-in code package for designing and scripting of agent component including a configuration of agent's role, actions, and its decision-making capability.

From the study and experiment of decision-making aspect using AI planning in our preliminary prototype with Jade multi-agent framework, the result indicates an aspect that conditional behaviour script, which can be categorized as simple reflex agent, should be implemented alongside with AI planning agent. The computation complexity for only AI planning agents has not scaled well with increasing environment

objects. Thus, by mixing faster conditional agents with AI planning agent considering with the agent role would prove a better of average performance scalability.

An extension of AI planning pipeline would be developed with open-source planner such as Fast Forward (FF) (Hoffmann, 2001) or Metric-FF (Hoffmann, 2003) due to their fast planning time and wide support for planning syntax languages in PDDL2.1. Moreover, the majority of AI planner in our reviews support PDDL2.1 features, thus it should be selected to be a parsing standard of our application.

Finally, the main core components in the software architecture has been proposed in Figure 4-1 to realize the requirement in the discussed software specification.

4.3 Framework architecture

In Figure 4-1, the design of architecture aims to reflect our specification. It consists of eight main components as the following: 1.) map manager, 2.) simulation manager, 3.) crisis event scheduler, 4.) agents, 5.) planning manager, 6.) game controller, 7.) user interface, and 8.) data manager. A brief description on each component is provided in the following subsections.

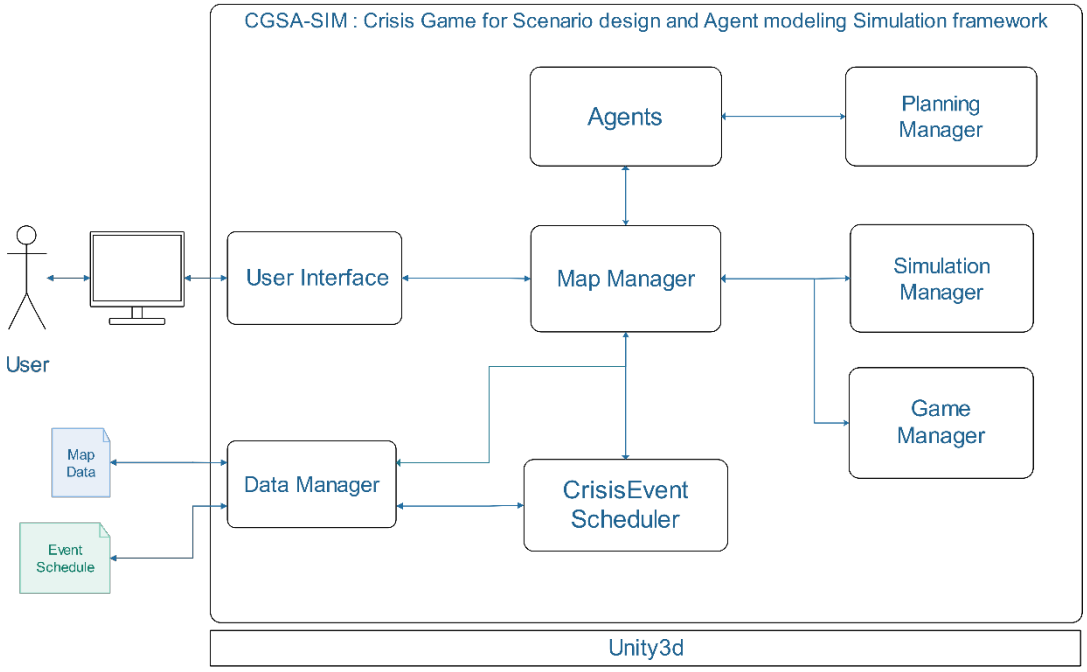


Figure 4-1 Framework Architecture with core components

4.3.1 Map manager component

The map manager has a data structure for a representation of world scene. In general, it is representing geography area in relationship to residing environmental parameters and states. Change in any parameter value and state can be triggered from a result of agent actions or crisis behaviour. In addition, an order of action execution has been centralized by the map manager to avoid conflict from action execution requests. Map manager also maintains monitoring task to change in cell states and adjustment of algorithm parameters

to facilitate a valid visual rendering result. The component provides an interface for data access of cell parameters and affect modification request to others.

4.3.2 Simulation manager component

The simulation manager contains a knowledge on crisis models. It has a logic to simulate crisis state behaviour such as heat transfer, depleting fuel, equalized liquid pressure, and other related data based on cellular automata framework. The control flags and formula modifiers are accessible to customize a progression trend of each crisis model. Additional disaster models can be easily extended from the existing built-in package, as it is a separate logic from map representation. The component provides an interface for cell-state evaluation and disaster state simulation behaviour.

4.3.3 Crisis event scheduler component

The crisis event scheduler maintains a sequence of script event for scenario setup. It assesses the condition of event such as triggered type, location, affected scale, and then apply the parameter changes to a corresponding area, agents in the scene or even global parameters. The framework allows an option to have the event sequence loaded into this component during run-time. Thus, it separates the process of map design, crisis model validation, and scenario scripting. The component also provides an interface to access a list of events via API calls to be triggered in the current time step.

4.3.4 Agents component

The agent subsystem maintains knowledge of agent characteristic including role restriction and a list of agent actions. It also maintains a reference of all agents in the simulation including the creation and termination procedures. For each individual agent, it is equipped with options for decision-making logic. This logic can be accessed via semantic API calls such as sensing, goal selection, and action selection. Practically, these procedures are to be implemented separately and can be reused on agent with similar role. The component provides an interface to access and modify any individual agent information, its behaviour, and containing resources.

4.3.5 Planning manager component

The planning manager contains a knowledge of parsing agent information into problem file formatted in planning domain definition language (PDDL), and then communicate with a AI planner to get a plan. The process is vice versa to parse a solution plan in PDDL's action predicates by mapping them with corresponding agent scripted actions in the framework. A domain file in PDDL for planner is stored as an external text file for ease of modification during run-time. The AI planning is optional to an agent, and it is deployed for a limited number of strategic scope agent to ensure performance scalability. Besides, this manager detects failure and regulate time-out of planning process since it can be unreliable for real-time system. The components provide an interface to request a plan for any agent.

4.3.6 Game manager component

The game manager provides a background logic to control overall game application. It interfaces with smaller subsystems in the framework to facilitate game-based logic of the application such as information query, control parameters, and data structures. The following subcomponents are fundamental to the proposed framework. First, time manager handles the keeping track of simulation step and animation delta time in the virtual system for conveniently synchronizing and calculating the action and effects. Second, game Statistic keeps record of crisis influence and quantity of abstract damage in any generated scenario, hence providing a mean to compare the effectiveness of plan and agent intervention. In this prototype, it is set to record a hypothetical value of collateral damages toward population and environment since they are straightforward measurements common in crisis simulation system. Last, Item system provides a separate function to define a resource object in the global item-definition data structure. It gives interface to query of resource information and its allocation toward agent or area. The game manager component also provides an interface to trigger gameplay state progression such as starting a game; simulate next step; end game; pause an animation; and related gameplay functions.

4.3.7 User interface component

The user interface provides an interactive screen communication with a user to the framework. It utilizes a menu and buttons alongside with screen visualization then accept input from the user to enable gameplay. The component is necessary for any interaction application and it interface with several components to parse user control with correct procedures. Both crisis scenario designer and practicing personal share the main interface in framework. The support tools for setup and modify of map representation, simulation flags, agent information, and file system are integrated with this component.

4.3.8 Data manager component

The data manager provides general-purpose functions to import and export information from the system. It has a parsing knowledge for the specific type of data structure using in the framework such as map data, crisis schedule sequence, and agent configuration. The main file format in the framework utilizes JSON standard due to its fast parsing speed and being generic across different external services especially with web service architecture. The component provides an interface to save and load several data format into the OS file system.

4.4 Summary

In this chapter, we have identified the specifications for the proposed crisis simulation framework developing on Unity3D game engine to provide a generic solution to design, test, and evaluate crisis scenarios.

The specifications are designed for three main aspects. First, the environment presentation and visualization, a grid of hexagonal has been selected for our system. The visualization is based on 3D in real-time to provide immersive strategic understanding of crisis emergency with a map editor tool to access and change any current simulation immediately. Second, the crisis scenario generation and simulation, the crisis scenario is developed from a terrain setup from procedural generation technique alongside with cellular automata mechanic integrated into each area parameters. Each simulation was calculated in the unit of discrete timesteps. Lastly, an agent modelling with AI planner, the crisis simulation has a benefit from implementing

an agent-based model from our general template class for a role representation and its decision-making using an IF-ELSE syntax or AI planning pipeline. These specifications have been reflected into a design of our proposed framework CGSA-SIM architecture containing eight main core components.

Next, the design of core components including their necessary processes are described in detail in software engineering aspect.

5 FRAMEWORK DESIGN

5.1 Introduction

In this chapter, a design explanation of each component is emphasized. The component main processes are presented in details. A map manager design is given with a terrain data structure decision and visualization process. A simulation manager design is explained with its process and time representation for a simulation step. A crisis event scheduler is designed with an analysis of scenario script structure. An agent representation is discussed with a design of agent behaviour component to be developed for serious crisis game on game engine technology. A planning manager and its AI planning process are explained for it provides an alternative decision-making capability to the agent. A game manager and its subcomponents designed has been emphasized as these modules provide a utility interface for the overall system. A user interface component and its use-case design are given. A data manager has been explained in component detail as it provides pre-processing and post-processing data interface for the framework. Lastly, an overall class diagram of CGSA-SIM is presented combining all the above.

5.2 Component Designs

5.2.1 Map manager

This subsection provides a process design of map manager. It provides an interface for a map-initialization and map-editing procedures. Moreover, the map manager oversees the visualization process from parsing relevant cell parameters into polygonal surface information.

5.2.1.1 Component design

To accommodate a grid of cells with local environmental parameters, A class UML of the map manager is shown in Figure 5-1. The map manager contains a cell information in a terrain grid. In each cell, the local environmental parameters have been stored separately from cellular automata state. This design enables modularity to extend a cell representation and simulation data with less conflict. The 3D assets are directly accessed by the map manager during a visualization and map editing process.

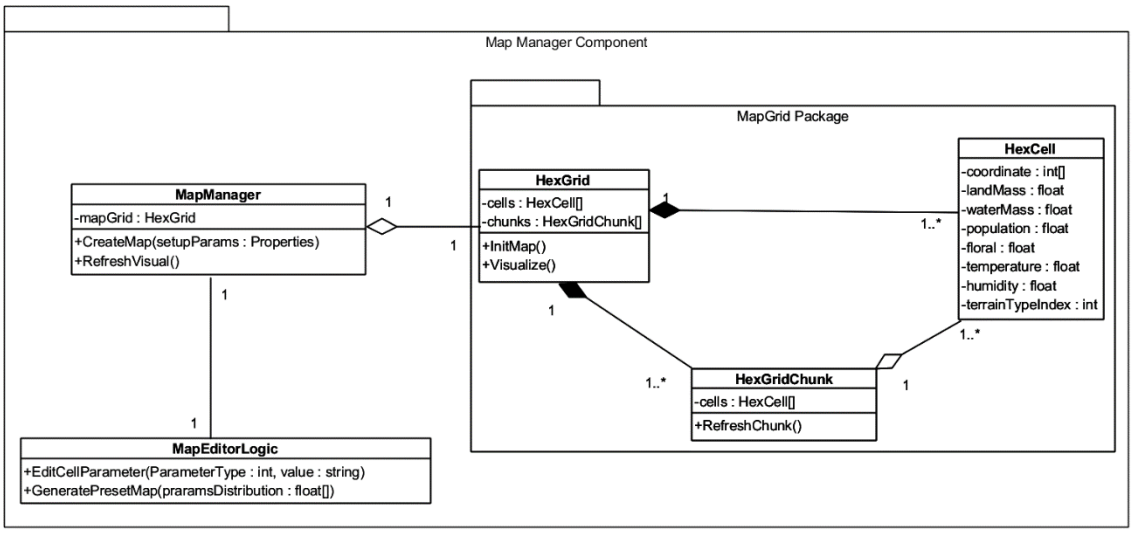


Figure 5-1 Class UML of Map manager

5.2.1.2 Map initialization and Map editing process

Figure 5-2 shows that the process is to generate a data structure of cell in a map grid for each cell contains an abstract set of environmental parameters. Next, the process performs an initial value assignment, and then convert a cell information into visualization data.

Figure 5-2 Map initialization process

In addition, the initialization of map environment is integrated with procedural content generation technique and can use a configurable formula of value distribution for each parameter in a cell. The map creation procedure has built-in user interface to influence an initial outcome of map generation algorithm, thus mitigates a burden of scenario designer in starting to draft a terrain shape by hand. For an example, the assignment of cell parameters can be programmatically defined for a specific proportion of topology details of forest to urban population area. The height of landscape in proportion to sea level can be balanced using a similar technique.

In this regard, the system is designed to allow modification on the map in any stage of gameplay to influence a result of simulation process. Map editor is a module from user-interfaced component to coordinate with the map manager for an access of world grid data. The modification of cell parameters is being reflected in a next visualization update loop. Figure 5-3 shows a map editing process by receiving an input from a user to modify a world map information.

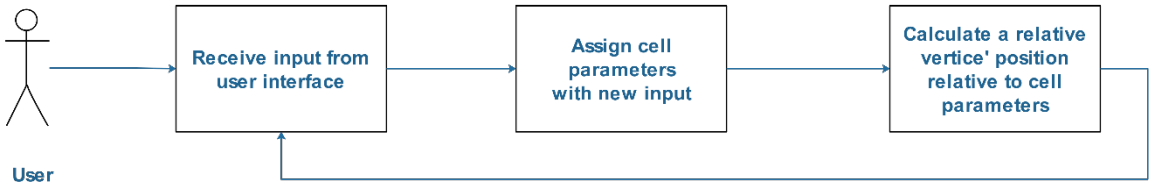


Figure 5-3 Map editing process

5.2.1.3 Visualization Process

The visualization is a process to parse existing cell's parameter into visual geometric surface and a selection of 3D assets to be instantiated.

Map has a grid of area cell that is holding parameters of resources, terrain topology and as a location concept of residual stakeholders. First, the ground terrain topology will be processed from each cell's coordinates and elevation for deriving a set of representative vertices in mesh triangulation process. Second, the measurement of area features is parsed in to visual level of population, floral, water mass for the selection of suitable 3D model representing these parameters can be placed on top of the terrain in town, plantation, specific landmark, and sea respectively. This step also includes an evaluation of disaster situation then select suitable special effects accordingly, such as fire and smoke particle on fire cell, rain on heavy rain event cell. Finally, the placement of agent representation that abstractly stayed inside the area is processed. 3D models representing each agent are placed in their corresponding cell's position. Figure 5-4 demonstrates the process of visualization in our framework. The level of details 3D assets can be setup using built-in script from Uniy3D. Then they can be used in our visualization process seamlessly.

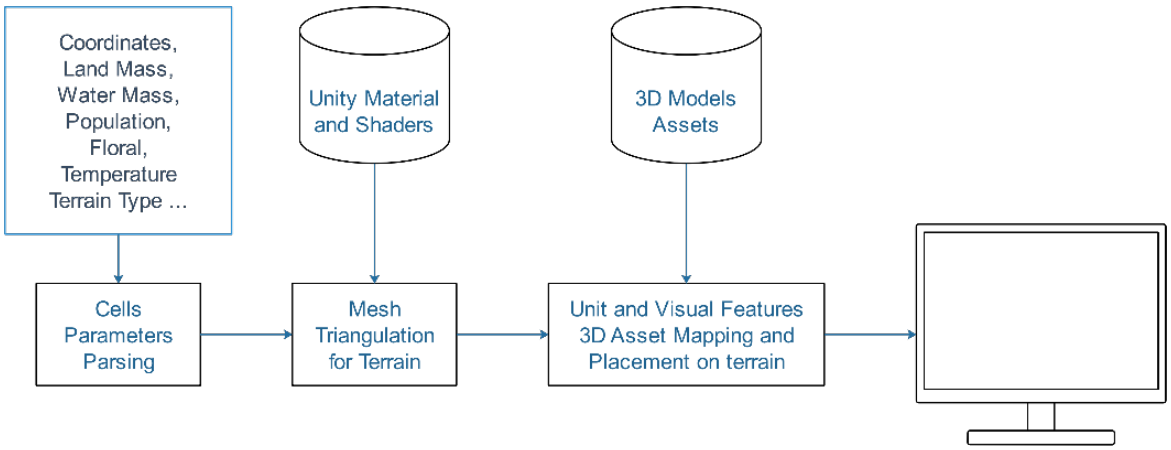


Figure 5-4 The process of terrain mesh generation; placement of 3D models and visual effects; Placement of agent's represented model in each area cell

5.2.2 Simulation manager

This subsection focuses on the design for simulation of crisis model using cellular automata. As the framework is being developed in Unity3D, the game engine allows us to have multiple separated scripts to be attached to each Game Object as many as necessary.

5.2.2.1 Component design

For each cell data structure, which normally keeps only area parameters, can be attached with an extension component to support state evaluation by invoking an evaluation logic from the simulation manager. The identifying logic in this simulation manager performs calculation test for crisis model state transition using a parameter such as fuel, water mass, land mass, and other related value to determine a trigger condition. In addition, the simulation manager also has a task to emulate disaster behaviour with its nearby area if applicable.

For an example, the ongoing fire situation would propagate heat toward its neighbour and possibly starting another cascading fire incident.

Figure 5-5 shows a class UML of the simulation manager. The simulation manager is designed to keep track of cell with an ongoing disaster state. The key feature for the manager is to have a target list and, in opposite, ignored list of cell to be excluded from any influence of CA behaviour. It gives a map designer ability to constrain an influence area of crisis formula on demand with ease by using a map editor interface.

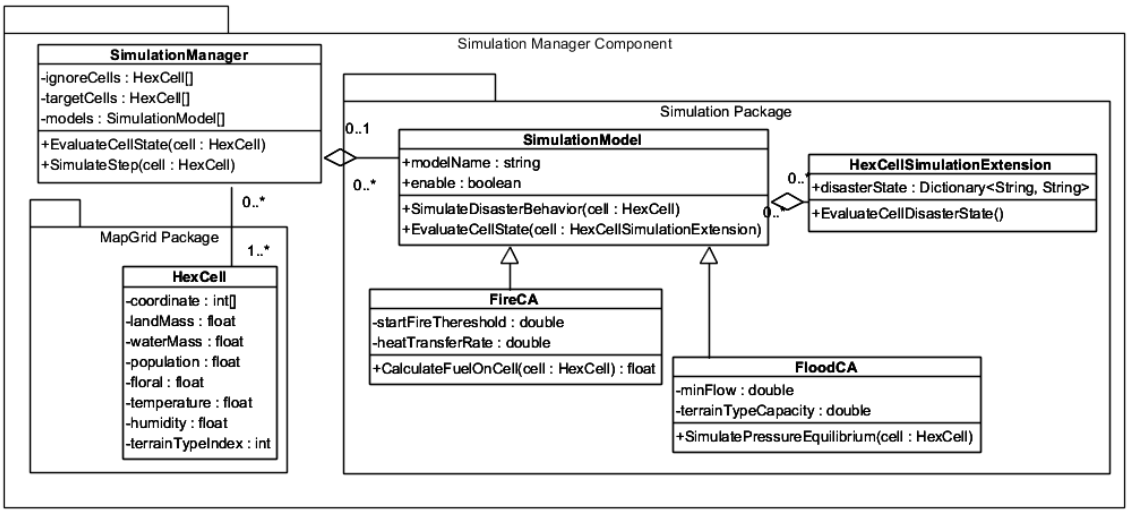


Figure 5-5 Class UML of Simulation manager

5.2.2.2 Scenario Simulation Process

The scenario simulation is started from map initialization with a map file or by a user using editor tool. In the main simulation process shown in Figure 5-6, with cellular automata technique, each cell invokes an evaluation of crisis state then rules for each state are applied. Finally, an agent can intervene in the situation by executing an action. During each loop, step 2 and step 3 are being updated with the changes from each step occasionally.

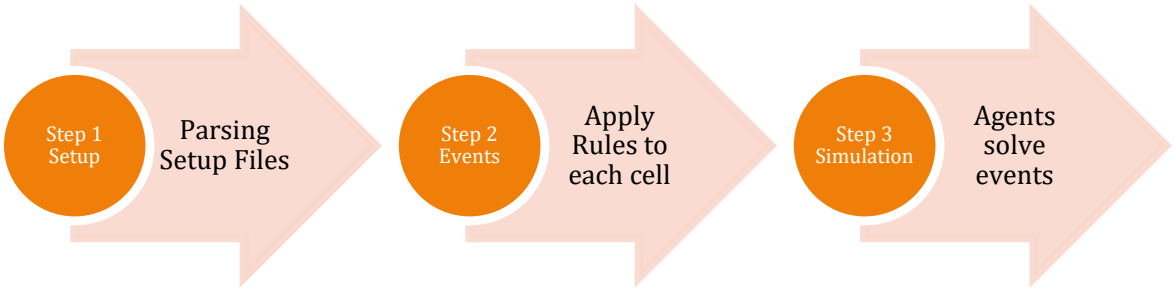


Figure 5-6 Crisis Scenario simulation process

5.2.2.3 Time as a turn-based design for simulation step

Time in our current framework has been constrained into a turn similar to turn-based strategy game. The main reason is this method synergizes well with cellular automata that requires every cell to finish its state evaluation prior any further action to avoid unsynchronized behaviour.

Figure 5-7 shows a flow of each turn function in sequence respectively. On each turn, first, it is starting with event scheduler fire a set of events having same starting time as current turn number. Next, simulation manager performs evaluation of disaster state from each cell's parameters. Second, cells with disaster state are processed for its related behaviour whether spreading temperature, simulate water pressure equilibrium or using other simulation model while including adjust cells anomaly parameters back to the global setup by steering. Third, unit movement and action parameters that are being spent during previous turn are replenished. Finally, the game logic then allows player or automated agent to take control freely in the current turn. The turn logic is being repeated until end of the simulation or close of application.

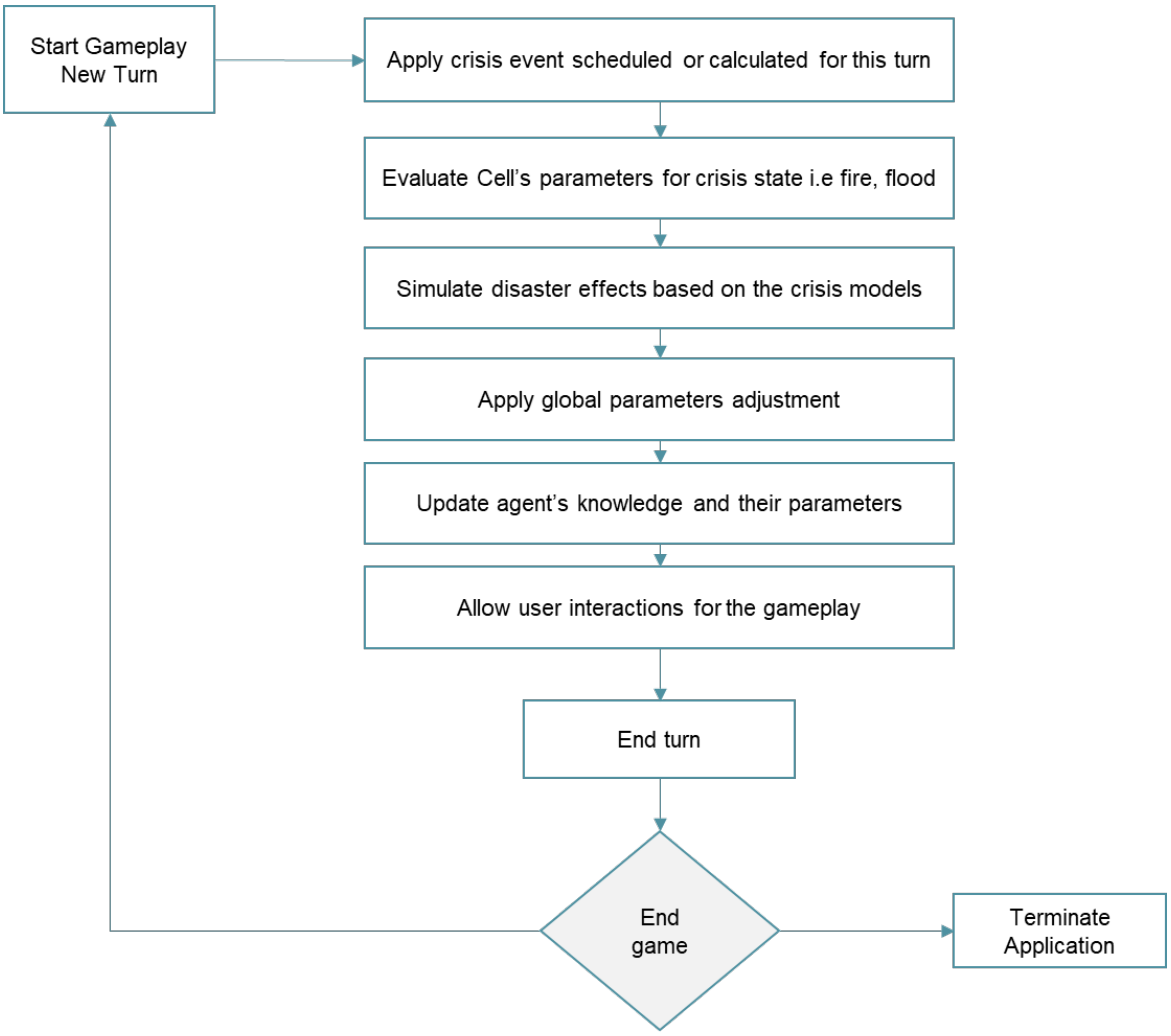


Figure 5-7 Turn logic during simulation gameplay

5.2.3 Crisis event scheduler

In this subsection, we emphasize on the design of crisis event scheduler to provide method in designing a crisis scenario.

5.2.3.1 Component design

With the requirement to maintain knowledge of each crisis event, the component has a crisis event template stored as a building block. By parsing crisis scenario script into the module, script keywords can be mapped with the template to generate a system object of crisis event. These valid events shall be assigned into the

event queue sorting by a triggering-turn time and starting condition. Moreover, the design of crisis event and its disaster situation also include option to define an unexpected event with a probability model. This design gives a freedom in generating a variety of crisis scenario with reusable aspect of common crisis events and situations' behaviour. Figure 5-8 shows a class UML of the crisis event scheduler. The event scheduler keeps all registered scenario in the active list during run-time then provides API calls for a game manager to regulate its evaluation of event activation.

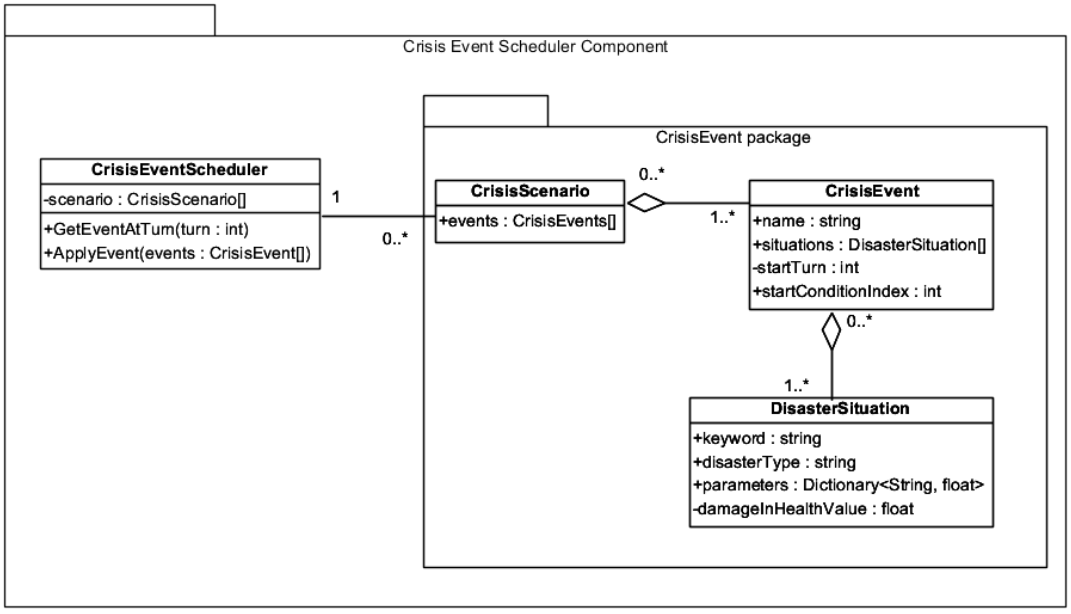


Figure 5-8 Class UML of Crisis event scheduler

5.2.3.2 Crisis Scenario structure

In the general definition, crisis scenario is a set of narrative events tailoring a condition and sequence of ongoing situation for interested area and stakeholders.

The context of scenario is described in map parameter data while crisis that is a sequence of ongoing anomaly or political impact has been scripted for chronological schedule separately. Figure 5-9 displays crisis scenario structure. As being noted, the main task for event scheduler component is to maintain a temporal sequence of crisis event to be applied corresponding with its conditions.

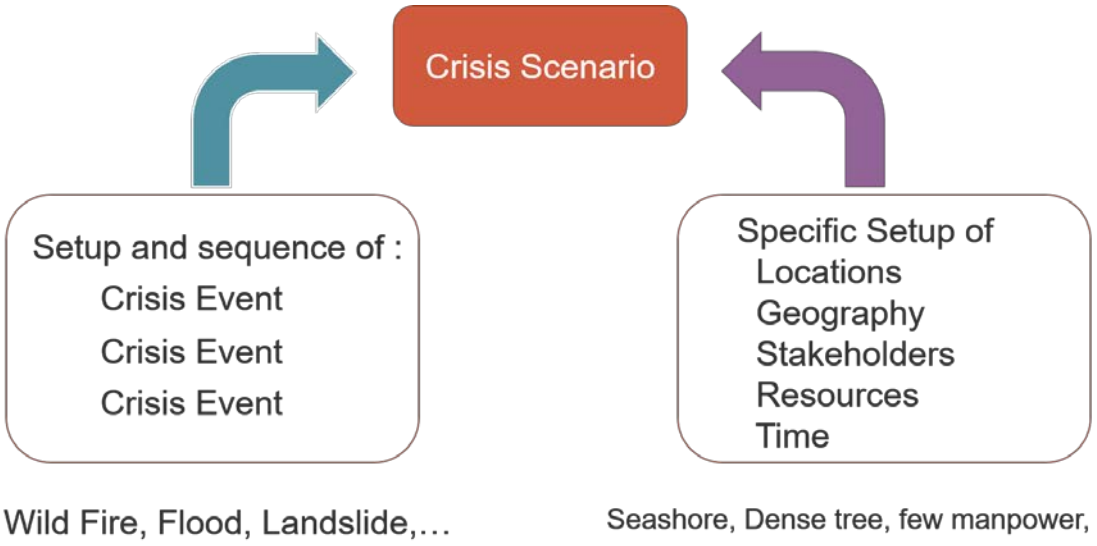


Figure 5-9 Structure of crisis scenario

To provide a basic structure in detail, crisis scenario context is described as a set of Crisis Events. For each event, it consists of a set of Disaster Situations and starting time in the simulation clock called simulation step in which one step includes calculation of ongoing disaster flag such as fire and flood and also evaluation of simulation parameters for state-changing.

Disaster Situation is considered as template for any anomaly of raise or decrease of simulation parameters similar to real-world problem. This allows us to describe the element of situation specifically in parameters and being reusable for similar context. For example, the "*Heat surge*" situation raises the temperature parameter of specific area with 30-50 degrees and, therefore, allowing Fire Simulation Behaviour to trigger a burning state if the condition is reaching the starting fire temperature threshold. Another example is the "*Water level rising*" situation which globally raises the water mass quantity of impact area, which can be shore line cells, and allowing Water Simulation Properties to trigger a flood state if the water mass is calculated to exceed the elevation of impact cell.

Crisis Event is considered to be an abstract event combining different situations altogether. The event usually consists of delay in simulation step, which controls the timing to be active state. Next, the active crisis event applies different disaster situations on designated cells. From the representation of crisis event, it gives us an ability to represent a single crisis event as a specific event template that is reusable similar to disaster situations.

Lastly, Crisis Scenario is a high-level narrative of sequenced crisis events. With the collective representation of crisis events and disaster situations, the specific scenario can be reproduced with flexibility of alteration such as a random starting time of crisis event and range of disaster situation impact on parameter based on assigned scale modification level. For an instance, general *Wild Fire* crisis scenario can be modified to have a faster starting fire event or more density of heat surge on area. This alteration produces more variation details for the same scenario.

5.2.4 Agents

This subsection focuses on how the CGSA-SIM can integrate an agent behaviour model for representation of scenario participant and yield observation result for analysis.

5.2.4.1 Component design

Each agent represents an entity in the environment mainly for a crisis personal and civilian. The class UML for agent component is shown in Figure 5-10. We design the system to have a unit as a general representation of agent common properties. Each agent can hold on a resource using a local InventorySystem while an AgentBehavior module is responsible for a decision-making process. This decision-making process is to have an agent evaluates the current map environment with its self-evaluation of how to interact with ongoing disasters. The simple decision-making design can have a conditional syntax to let agent take action with its available capability and holding resources. More advanced decision-making design directs the agent to gain insight on wider scale of situations and its possible cascading effects. The drawback for advanced agent is that it probably requires higher computation time than the conditional logic. Thus, in our design, the agent component takes into account to accommodate this decision-making variation. Any agent can be equipped with different logic module then can communicate to each other with ease by a common knowledge on implemented unit action.

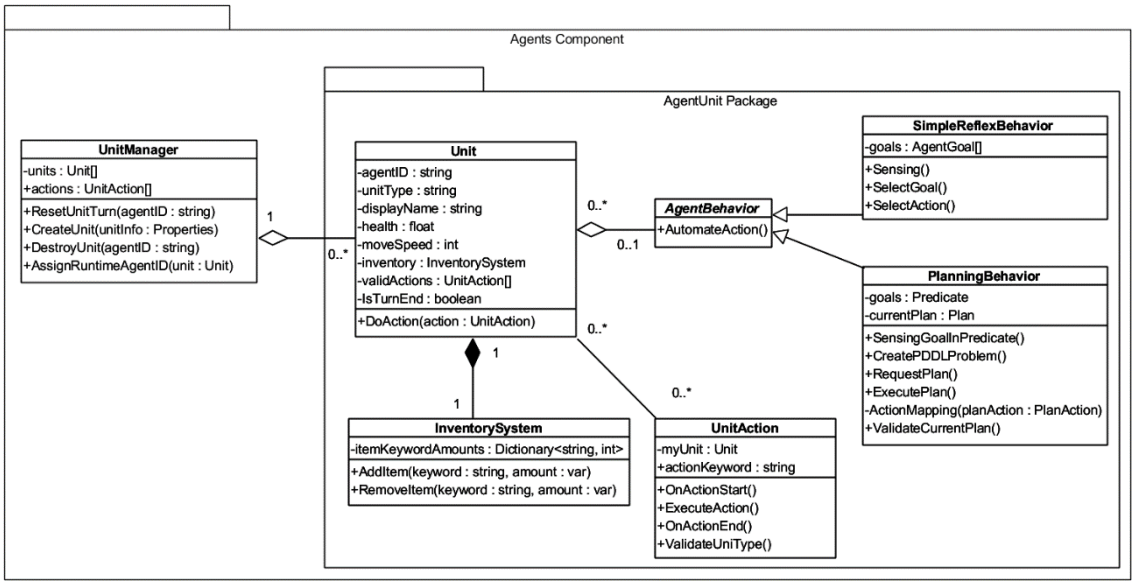


Figure 5-10 Class UML of Agents

5.2.4.2 Agent Behaviour design

From the unit class description, an agent can be modelled into a crisis stakeholder with different roles such as facilities, firefighter, police, and civilian. Each unit has to perform a suitable action according to the current incident and their role. These actions can be varied in the objective of resource allocation, giving a directive for minor unit, and use a tool to perform mitigation process.

In regarding our agent behaviour component, actual implementation of decision-making requires further discussion. As mentioned from background topic, there are several prospective techniques to develop the action selection process. On such an example is to use an if-else reflex agent using sequence of condition

testing to determine a priority of executed action. Another approach is to parse world-environment information into AI planning language then use a planner component for tailoring a plan, sequence of actions to accomplish a given goal. Both technique have its own benefit and drawbacks.

Fundamentally, the if-else reflex agent is a common implementation due to its simplicity and performance effective but it lacks of deeper knowledge representation capability in AI planning technique. From this reason, it is a design question for why our framework should not fully based its agent decision-making with AI planning. To test our open assumption, we have investigated a performance of whether completely an agent relied on planning is applicable for real-time crisis simulation system. Our test platform has been employing with JADE framework, as it is a JAVA middleware providing an agent-oriented programming environment. It provide flexible structure for developing an on-top extension. The main feature is to provide a core run-time environment implementing a life-cycle approach of agent (Bellifemine et al., 2007). Currently JADE is being distributed as an open-source project under LGPL license.

The result from our study has indicated that increasing the number of agents will require more because they generate more computational time from a collision in action execution in the environment such as trying to execute an action on the same object. Only one agent would success while another fails triggering replanning process. This situation leads to studying of opportunistic competition between each agent and invention of more efficient planning and replanning algorithm for multi-agent system but, currently, also it is not a focus of our research. Furthermore, the increasing of problem size has posed an issue on time required for planning, thus, it becomes a challenge to develop real-time application with only planning decision-making approach. Emphasizing on the entity count, this test system on JADE allows not just an area to be triggered with On-fire as the decorative objects can be set to trigger fire state. Considering the issue with our crisis model, we believe cellular automata with simplified representation of area in each grid cell can reduce a problem size significantly but still not small enough for executing real-time planning solution of every agent. From the drawback, we have made a decision on agent architecture so that the crisis personals or lower-tier agent should initially developed its behaviour with if-else reflex approach for performance purpose. While planning behaviour can be extended to only decision-making logic on medium and higher tiers. Still, this design trade-off has a benefit that reflex agent is simpler to design and test since AI planning required more time to script the domain definition of each roles and extensive test for ongoing-development crisis model for validity.

Designing in this regard, our CGSA-SIM has a code template as a component called Agent Behavior to be equipped as an extension on any unit. The behaviour component serves as an abstract representation of automated agent architecture allowing an implementation via a provided interface. Figure 5-11 shows a design of agent behaviour component equipped on any stakeholder unit with the API interfaces, which is involving sensing an environment, goal selection, and action selection to perform an interaction with the map environment. The trigger to execute all these actions can be invoked in a common function call such as “DoAutomateAction” so that both agent decision-making component can be integrated seamlessly in the gameplay loop.

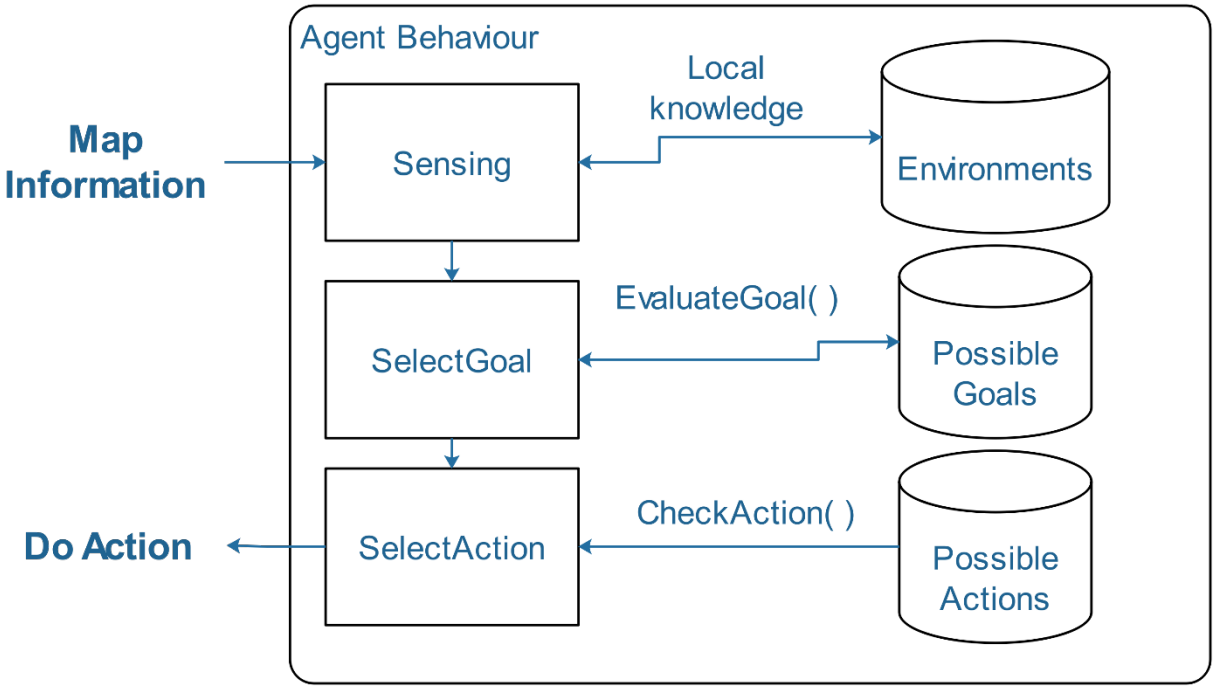


Figure 5-11 Agent Behaviour component in the Stakeholder Agent

5.2.5 Planning Manager

This subsection explains a design of planning manager component. It acts as an interface system between Unity3D application and AI planner which is implemented in the separated web service architecture.

5.2.5.1 Component Design

The planning manager invokes the execution of AI planner. Then the planner allows the generation of a solution plan consisting of sequenced actions related to a given world states, allocated resources, and possible actions. Normally, this world environment information can be retrieved from agent's local perspective and must be pre-processed into PDDL format so that it is being stored in the local KBS of the planner. Figure 5-12 shows a class UML of a planning manager. It provides a data structure which can be used to represent a knowledge of entity and action in the predicate format corresponding to PDDL standard.

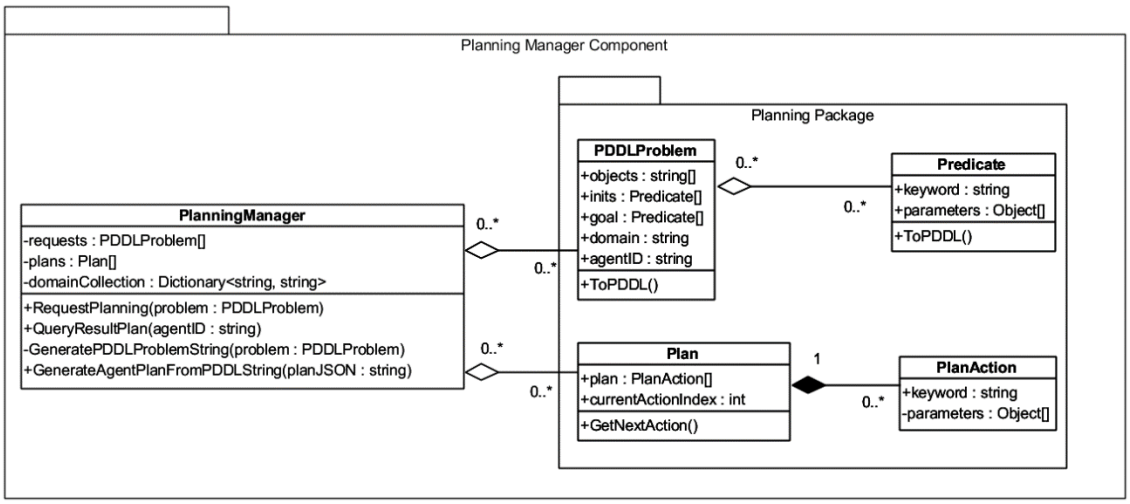


Figure 5-12 Class UML of Planning manager

5.2.5.2 AI planning process

The process of AI planning is described systematically in Figure 5-13. The input parameters are domain and problem file in PDDL to update a local knowledge of planner. The invoked planner evaluates a goal condition from problem whether it is reachable with current initial setup and action from the domain file. Reachable goal means that a solution plan from the current state of world representation toward a desired state can be found via AI planning process. The calculation for reachable goal is time-consuming process since it is to explored with possible action represented in the problem domain file until the conclusion can be made. If the goal is reachable, the plan can be generated into a list of action in an execution-order sequence. Otherwise, it generates a planning failure report as a result. Moreover, while executing the plan, the world environment state has not exclusively manipulated by only one agent so the next action in the sequence might not be suitable. In this case, the replanning process which forces the agent to discard its current plan then recalculates anew is being called. Figure 5-14 describes a monitoring process that observes the result from execution of action then decide whether the agent require the replanning process or not. In the beginning, the agent monitors its current executing action from the solution plan. In the event that action's result is failure, next action in the current plan is tested whether the next action in the list is reachable or not. If it is unreachable, new plan toward agents' goal is generated. If everything still results in failure, the agents reports its (failure)result back to the higher-scope agent or game controller. Then the configuration of failure planning can be varied on the scenario designer. The alternative decision-making logic such as IF-ELSE conditional syntax can substitute the current AI planning mechanic.

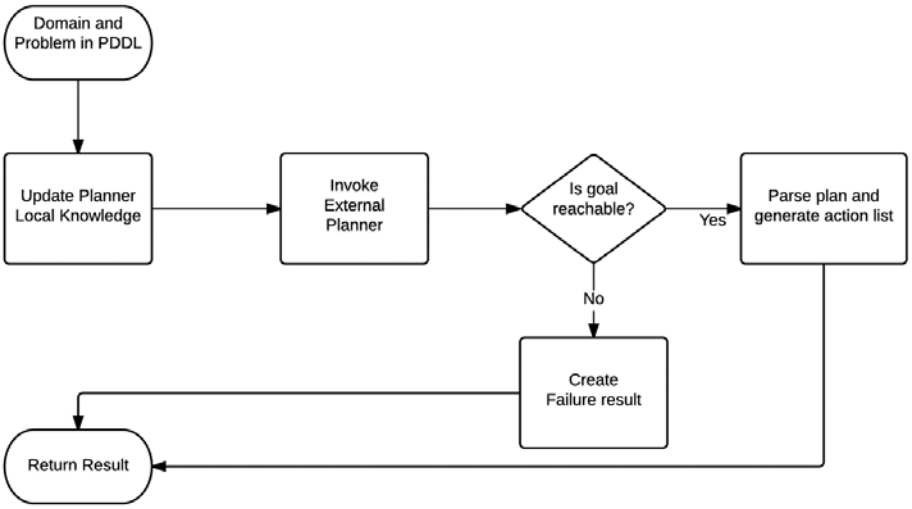


Figure 5-13 Planning Process flowchart

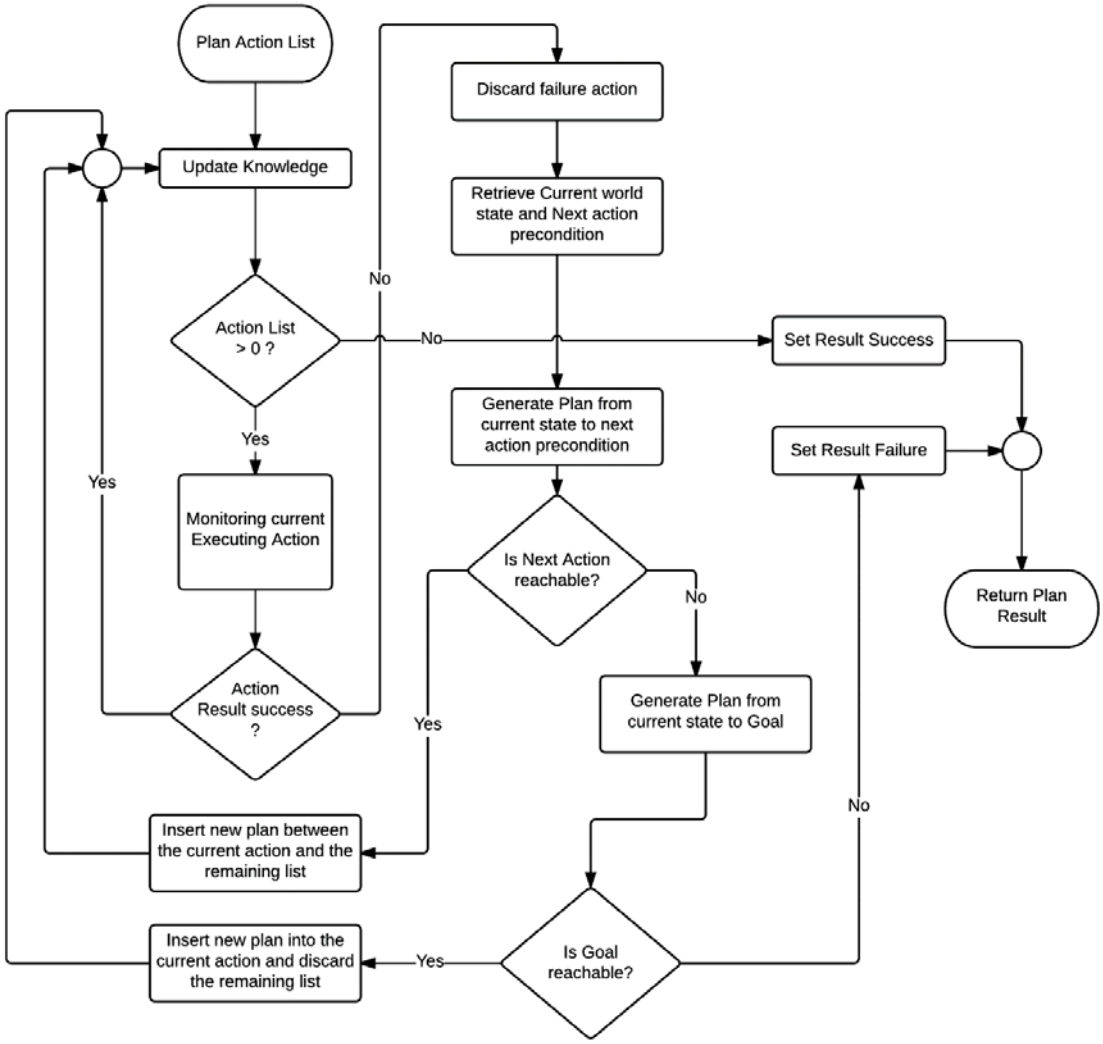


Figure 5-14 Monitoring and Replanning Process flowchart

Crisis agents in the framework are described with scope-levelled of responsibilities form High to Low. The AI planning process is aimed to be equipped in the agent with management task. This type of agent requires global awareness of disaster situations, available resource, and deployment of operational personals. In contrary, the operational personals only receive a directive for the site of crisis then act accordingly to a predefined procedure. Figure 5-15 demonstrates how the scope-levelled agents can be generalized into 3 types. Decision-making agent oversees the high-level planning for each group of control agent. Then the control agents usually represent as headquarters of local operation use given direction to assign a specific goal for their field agents. Finally, field agents straightforwardly resolve the given goal with their given procedures. Our approach enables balancing between longer computation of AI planning and faster conditional syntax of field agents.

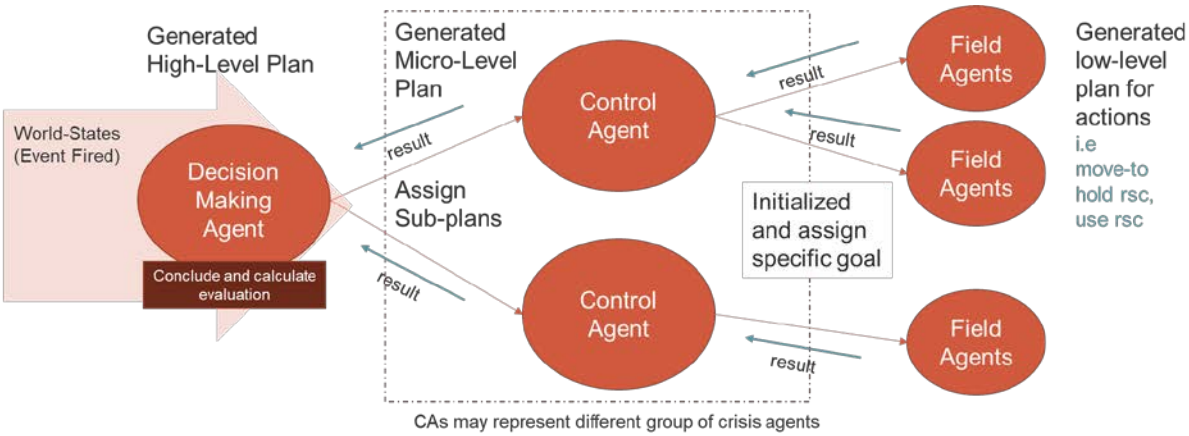


Figure 5-15 Agent decision-making designed by scope-levelled of responsibilities

5.2.6 Game Manager

This subsection provides an explanation on game controller component. Its main tasks are to control gameplay and triggering of suitable sequence function from the user interface component. Importantly, game controller has time manager, statistic manager, and item manager as subcomponents.

5.2.6.1 Component design

The game manager overseers state of CGSA-SIM application. It has a collection of subsystems mainly for a query reference of animation time, gameplay metrics, statistics, and item database. Game state is a finite-state machine variable to transit the current gameplay from start game, pause game, play game, and into the end game. The input to control game state transition is received from a user interface component. Figure 5-16 show a class UML of game manager and its subsystems.

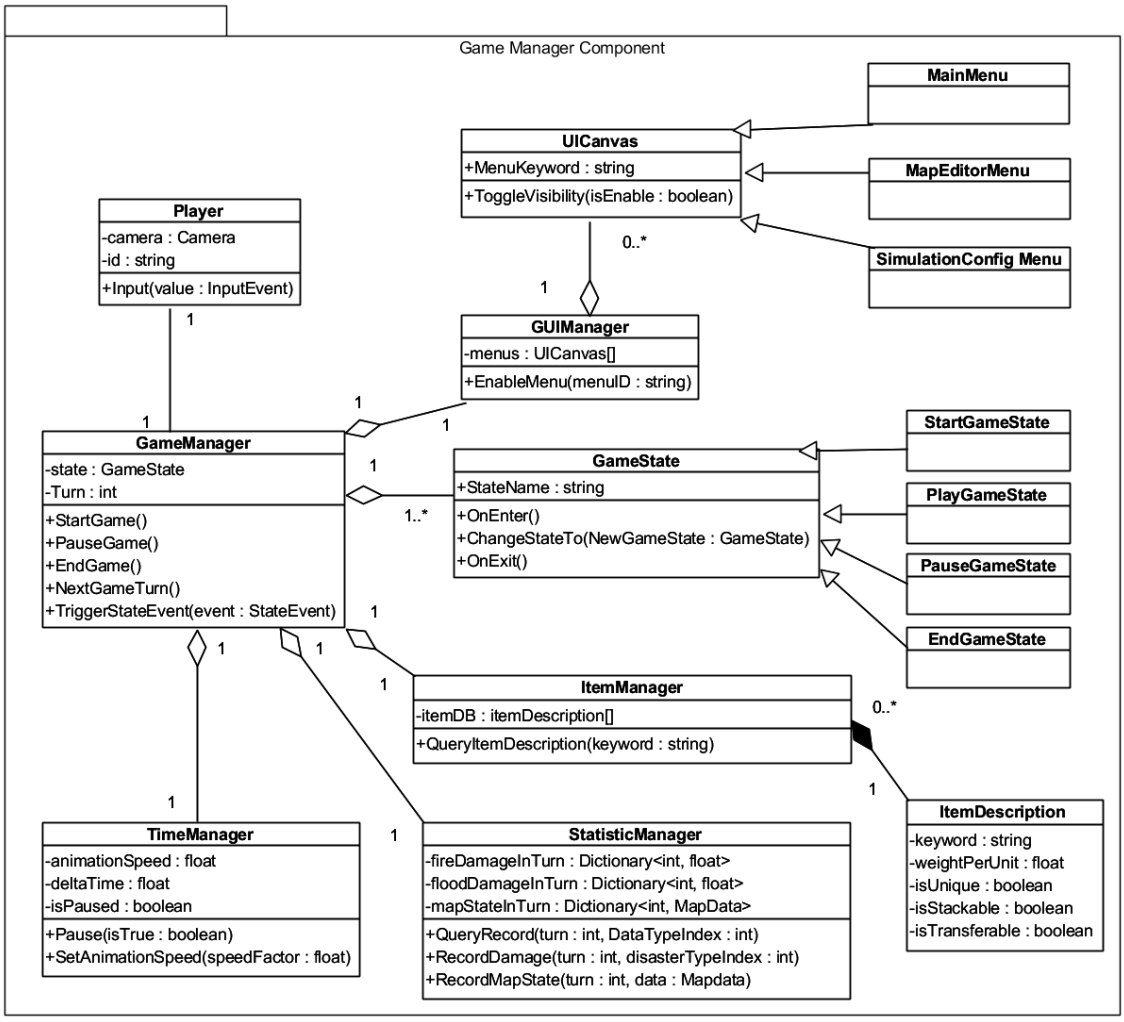


Figure 5-16 Class UML of Game manager

5.2.6.2 Statistic Manager

The subcomponent is designed to hold an information from simulation of crisis behaviour. For the starting stage, our framework provides a building block to record influenced damage from a cell with disaster state flagged such as on-fire and on-flood. The abstract damage could be recorded from a simulation manager logic directly while a user interface can query a result to process for information analysis. Besides, this system holds a run-time information statistically for a purpose of performance evaluation in different perspective such as framerate, simulation time for each turn, and agent run-time for decision-making and AI planning.

5.2.6.3 Item Manager

Game controller provides an item system subcomponent to keep a reference for item management in the gameplay. Item is an abstract representation of resources in our framework. It has the building block to define a dictionary of item definition. Item's parameters can be configured by editing a set of keywords such as quantity min-max constrains, weight per unit, role-specific requirement, and other related information. In addition, it keeps tracks of local item database in each agent during gameplay.

5.2.7 User Interface

This subsection gives more explanation on the user interface design. The components receive an input from the user to perform gameplay state transition and configuring on each system components.

5.2.7.1 Component design

The user interface provides visual feedback to the user including a toolset to influence other subsystems. It communicates a command as a function call directing to necessary object. The component requires a collection of suitable graphical assets and function to display screen menu. Figure 5-17 shows the design of user interface component. A design of our user interface functionality for our framework is based on the use case from Figure 5-18. The user can perform five main tasks using a toolset provided by this component: Map initializing configuration, Terrain editing, Trigger simulation turn, adjust planner system, and generate export file. The communication between the graphical user interface and the game manager has been included as a part of code design in the game manager package.

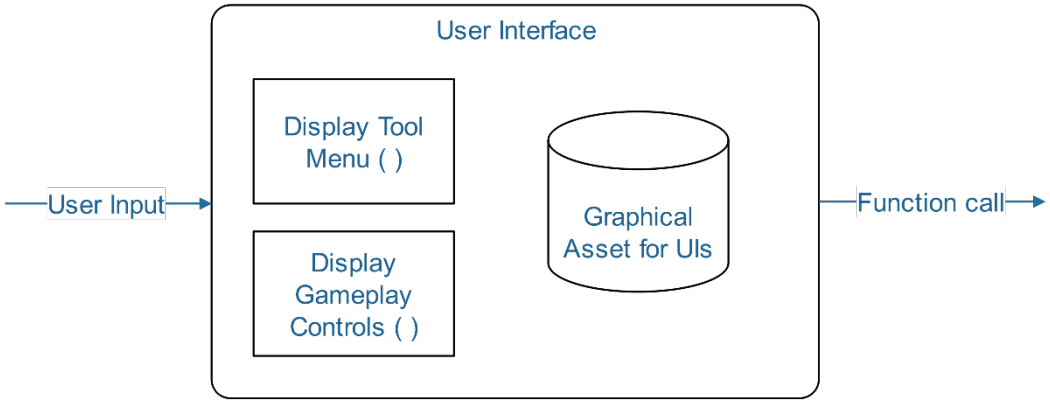


Figure 5-17 User Interface component

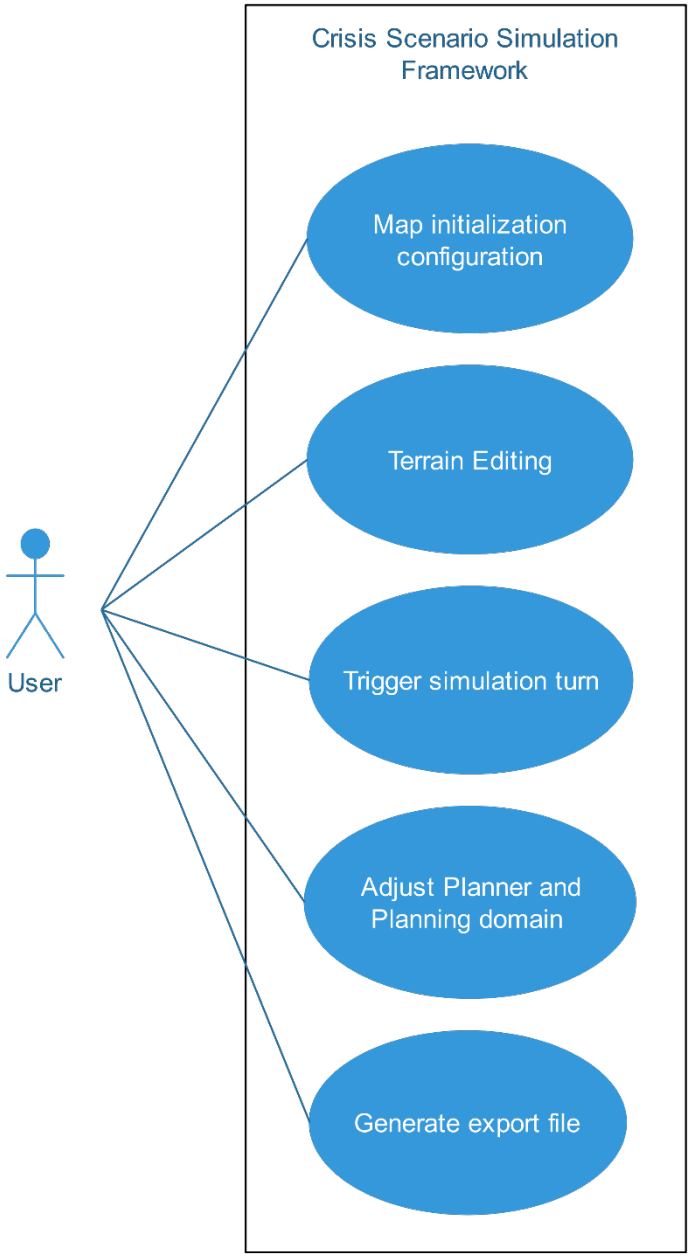


Figure 5-18 System Use Case

5.2.8 Data Manager

This subsection provides a general design of file manager component. It performs file parsing procedure during import and export process of a framework information.

5.2.8.1 Component Design

The parsing format for import and export process is based on JSON. The run-time gameplay can be exported by generating an associated data structure for each type of information. The component then provides a general interface to write a data into text file. In vice versa, the import file must be identified from its header to trigger a specific parsing procedure to convert them into the suitable data structure. Figure 5-19 shows a class UML for the data manager with its utility data class. The different types of system data are extending the

JSOnSerializableObject to implement a common interface of generating its own JSON and loading procedure. Then the data manager regulates the overall process of accessing an operating file system.

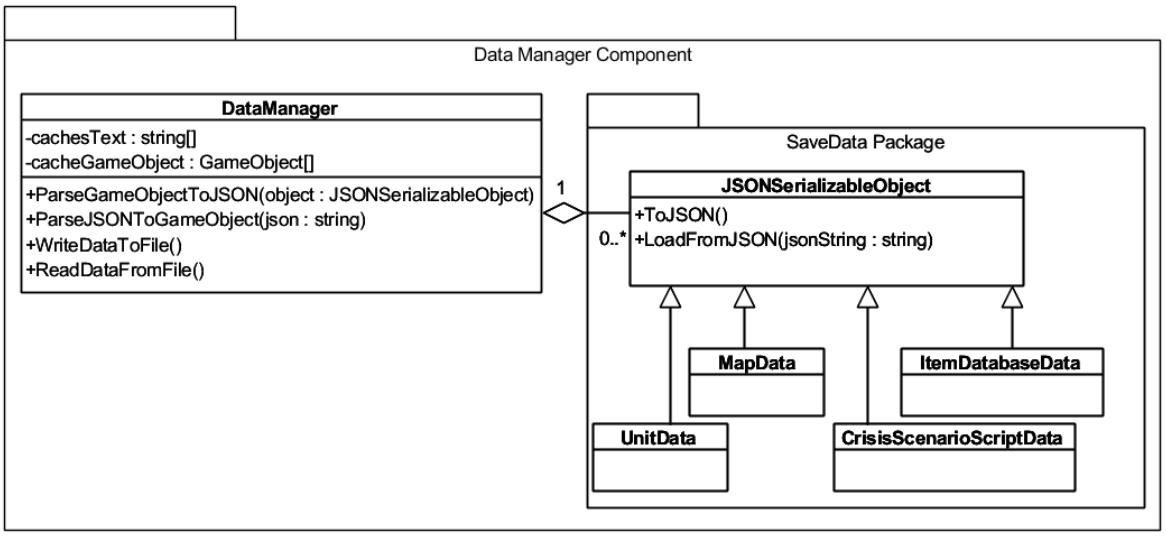


Figure 5-19 Class UML of Data manager

5.3 CGSA-SIM class diagram

The design on each main components of CGSA-SIM (crisis game for scenario design and agent modelling simulation) framework has been presented in this chapter. Figure 5-20 shows an overview class diagram of the framework.

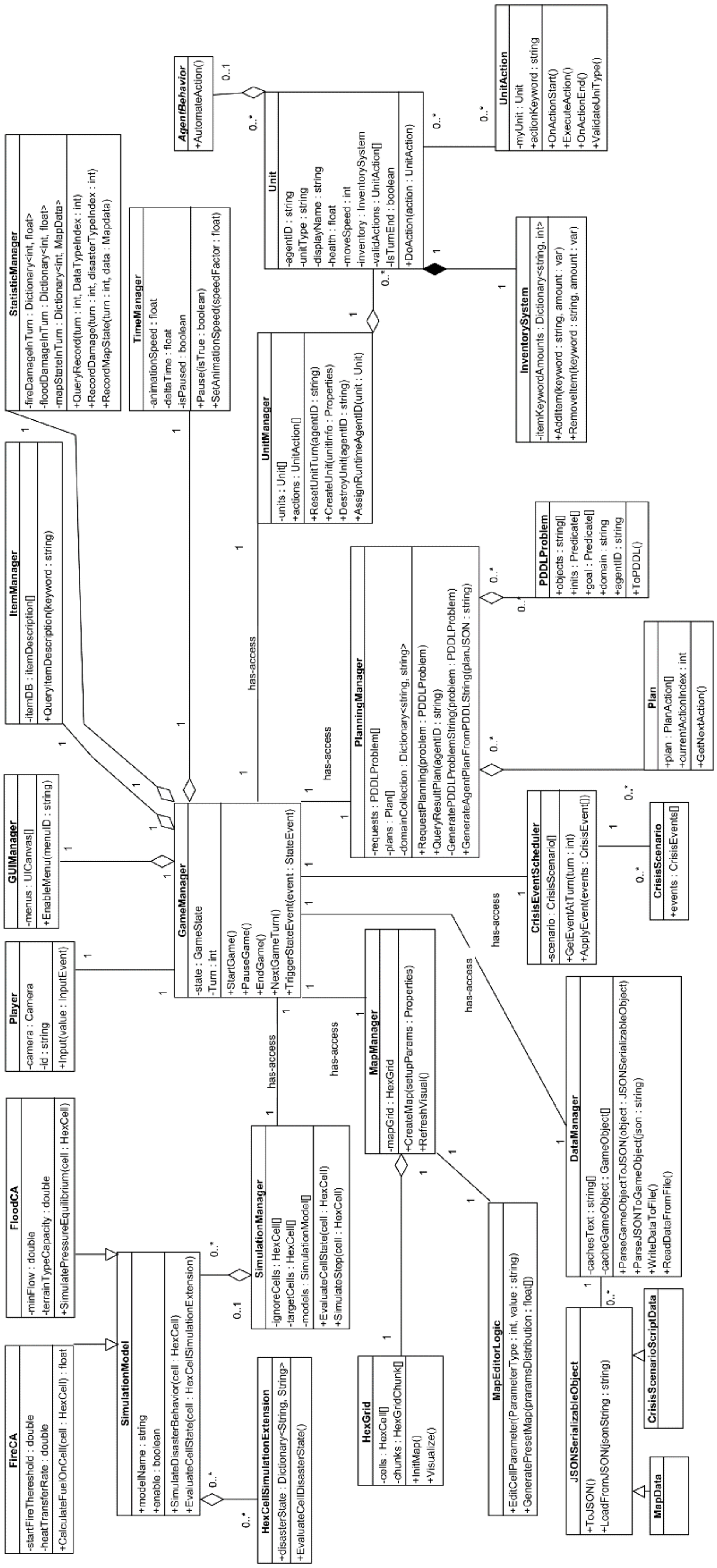


Figure 5-20 CGSA-SIM for the framework development plan

5.4 Summary

In this chapter, the design detail of CGSA-SIM main components has been described with a providing of its class diagrams and explanation in each core processes.

CGSA-SIM consists of eight main components that are map manager, simulation manager, crisis event scheduler, agents, planning manager, game manager, user interface, and data manager. The development of each component aims to deliver a generic crisis modelling and simulation system based on the game engine Unity3D. The map manager addresses an environment initialization while provide an accessible interface for the other components to interact. Next, simulation manager accesses the information from the map to perform a state-change calculation in regard of registered disaster model using an approach of cellular automata technique. In addition, the narrative of emergency event can be controlled using a script structure in JSON then parsed into the crisis event scheduler component to trigger an environment and disaster stage flag for the area. Then, the agents are a template class to represent stakeholders such as firefighter team or civilian in the ongoing emergency. Each agent consists of dedicated role and its behaviour design which influence a selection of suitable action to reach each agent goal. The planning and conditional syntax agents can be applied together to collaboratively work toward solving a crisis problem. While AI planning may be time-consuming in a large problem set, the design can mitigate this issue by assigning only a few individual agents with complex reasoning in opposite to a simple action execution in a civilian agent logics. Only an agent with managerial role require strategic understanding of crisis event so they have a planning decision-making module equipped. Furthermore, the planner manager allows a planning-type agent behaviour to parse their world understanding into a standard PDDL2.1 predicate syntax then communicate for a request for plan with a planner. Then the rest of agent receives a directive from this planning agent as a goal. This approach averages the computation necessity in the framework. Lastly, the game manager cooperates with user interface and data manager to facilitate the basic API calls and gameplay logic.

In the next chapter, the implementation features a technical knowledge to realize our final application prototype.

6 IMPLEMENTATION

6.1 Introduction

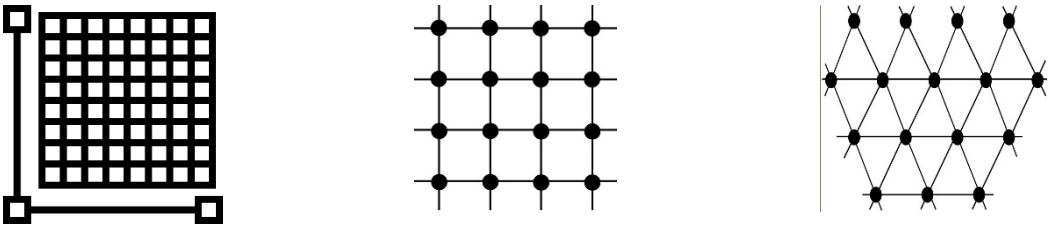
In this chapter, we emphasize on an implementation technique for a framework. A map representation is discussed with the usage of procedural terrain generation technique. The development of crisis disaster model is explained with preliminary experiments and its verification using test cases. The integration of the experiment finding into a simulation manager in the framework is presented with example crisis models. An implementation of agent as a game unit is explained. An extension of agent behaviour component for conditional reflex and AI planning approach is discussed.

6.2 Map representation with Procedural terrain generation process

From literature in simulation system, the structure of map representation has been an important decision that must be identified early since it can make huge different in the performance and pathfinding quality.

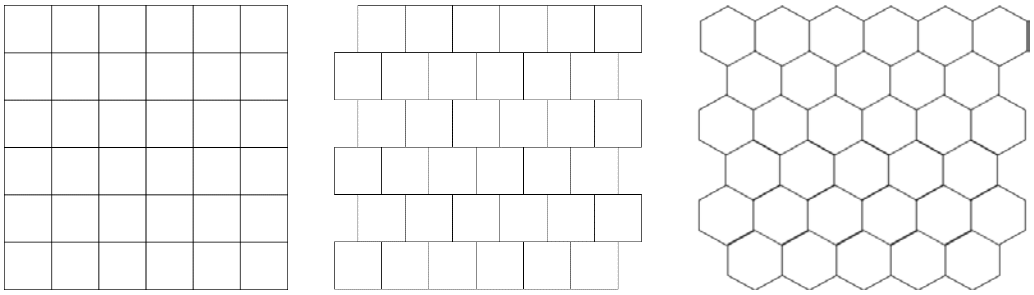
Most games and research have been using a grid map, which is often called as lattice graph, due to its simple and easy to understand. The characteristic of grid is a uniform subdivision of area into a small resemble shape to each other, sometimes, called *Tile* as demonstrated in Figure 6-1 (a), and it can be represent as an equivalent of regular interval-distance graph nodes in Figure 6-1 (b). The common shape in grid using in many systems are square, triangular, and hexagonal. In contrast, more sophisticated alternative to grid is to use a non-uniform graph that each node representing a key area of location or obstacles. The pathfinding in non-uniform graph uses different approaches from 2D grid since it is considering a free-movement around possible represented space. From this point of view, we decided to choose a 2D grid lattice structure since it is straightforward to present an understandable map. Another advantage comes from its affinity with cellular automata technique, which each tile can dynamically trigger self-state change regarding on current parameter in terrain characteristic such as elevation and water level.

In addition, our framework was presenting this grid in uniform hexagonal cell structure. Each tile, called Hex Cell, had consisted of six cell neighbours, and represented cube coordinates defined as *Cell(X, Y, Z)*. The process of transforming grid from traditional 2D grid into abstract Hexagonal grid was briefly demonstrated in Figure 6-2. The distinct advantage from choosing hex cell over the traditional square cell lattice was that the distance between each direction to centre of the cell would be symmetrical and, therefore, provided more realistic in traversing between coordinates shown in Figure 6-3.



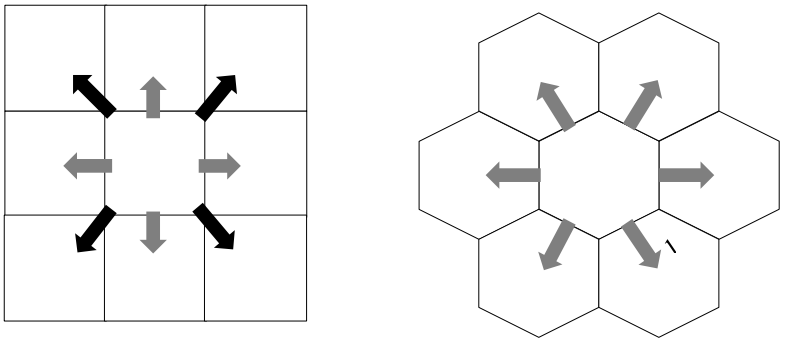
a.) b.) c.)

Figure 6-1 Type of grids (a) 2D lattice grid (b) Square lattice graph (c) Triangle lattice



a.) b.) c.)

Figure 6-2 Generation of Hexagon Grid structure (a) base 2D-Square grid (b) offsets cell's centre by rows (c) transforming square rendering into hexagons



a.) b.)

Figure 6-3 Grid representation and its distance to neighbours (a) Square cell with 8 neighbours (b) Hexagon cell with 6 neighbours

In assistance of map terrain creation and design, we implemented the procedural generation technique to generate 3D coordinates of each cell area combining into triangle mesh for visualization in Unity3D mesh rendering component. From the centre vertex, we generated relatively six directional corner coordinates in order of northeast, east, southeast, southwest, west, northwest respectively clockwise, each draw a triangle mesh representing a direction from centre. Positioning each cell's centre was offset with pre-processed result from Figure 6-2 (c.), we constructed a base mesh for a terrain, and each cell can be identified with its 3-index number shown in Figure 6-4. From this example, we emphasized the terrain topology by assigning a colour variation such as green for grassland and blue for sea surface.

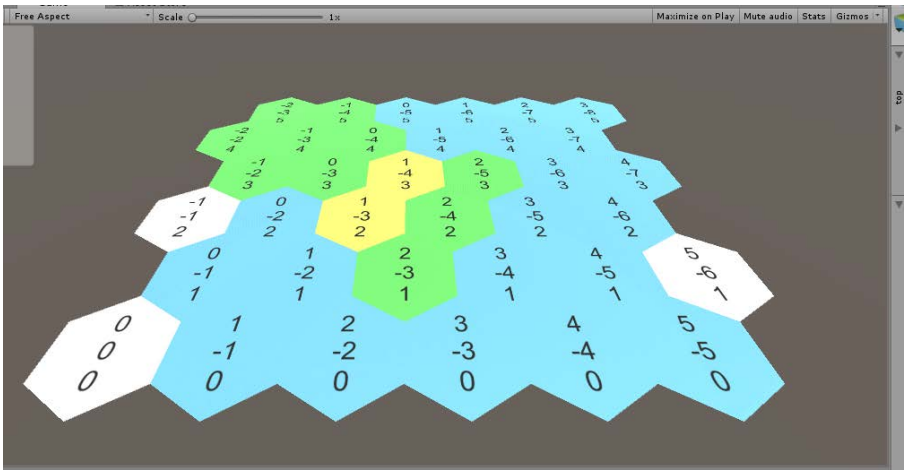


Figure 6-4 Base mesh surface generated from 2D Hexagonal coordinates of grid

To create subtler colour transition between each cell's terrain colours, we created a boundary area between each hexagon by reducing its radius then bridged the gap between each another with transition meshes. Figure 6-5 illustrates a shrunk hexagon cell being connected to each other by adding bridge surface on northeast direction from bottom-to-top order. Next, another bridge had been connected on east direction shown in Figure 6-6 while the only remaining triangle area was filled separately with equally influence from three adjacent edges in Figure 6-7.

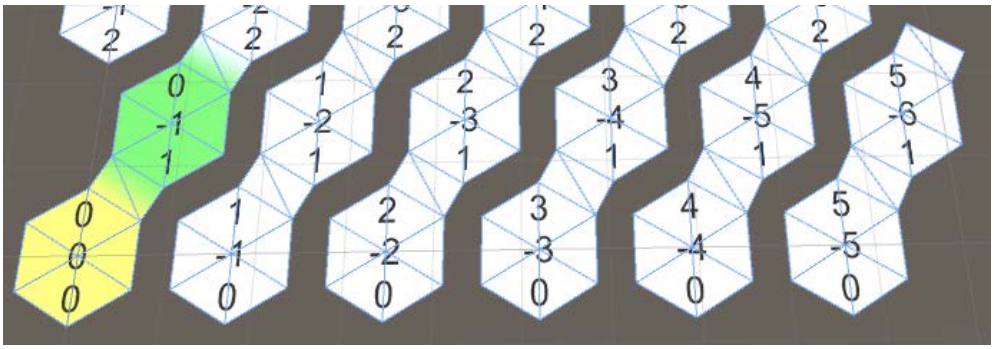


Figure 6-5 Bridging a northeast direction of each cell

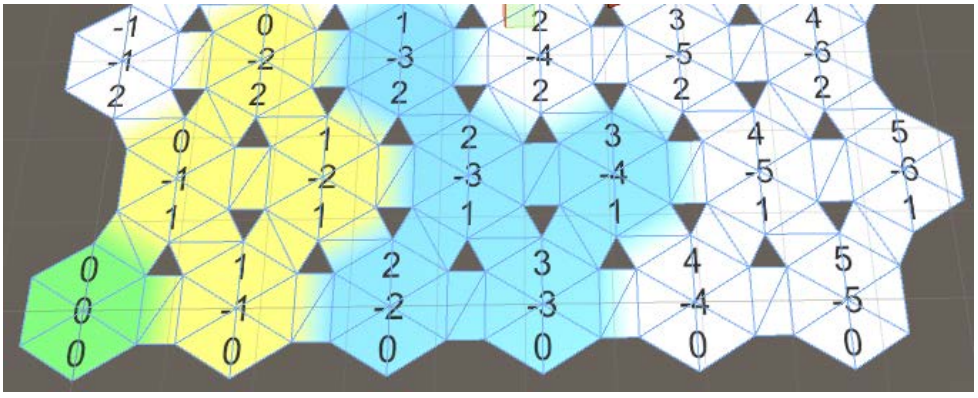


Figure 6-6 Bridging an east direction of each cell

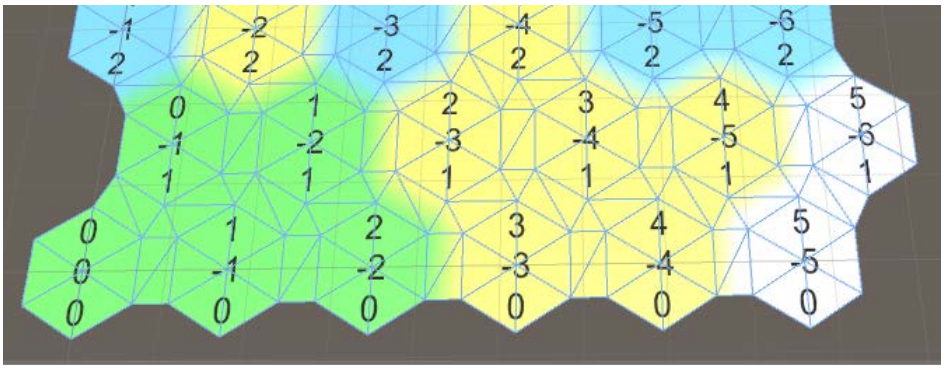


Figure 6-7 Fill in triangle between three adjacent hex cells

Finally, a landmass parameter had been introduced to the cell's data, which it was used for calculation of discrete elevation level on generated surface. By multiplying this elevation level with height scaling value and y-axis value of cell's coordinates, Figure 6-8 shows a final result introducing more realistic perception of terrain geography with elevation offset. Moreover, complex version of connecting surface had been represented in sub-interval terrace of height-difference hex cell shown in Figure 6-9. Although elevation-terrace method provided more visual aesthetic features it introduced extra adjustments for further upcoming stages, therefore, our framework had progressed with basic surface model with additional subdivision step shown in Figure 6-10.

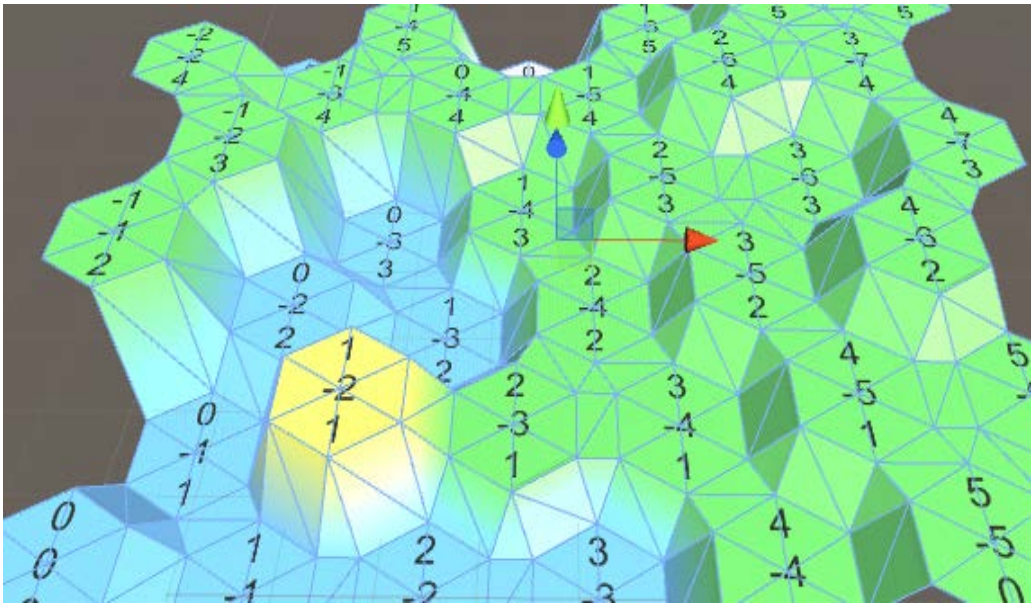


Figure 6-8 Terrain with elevation offset implemented

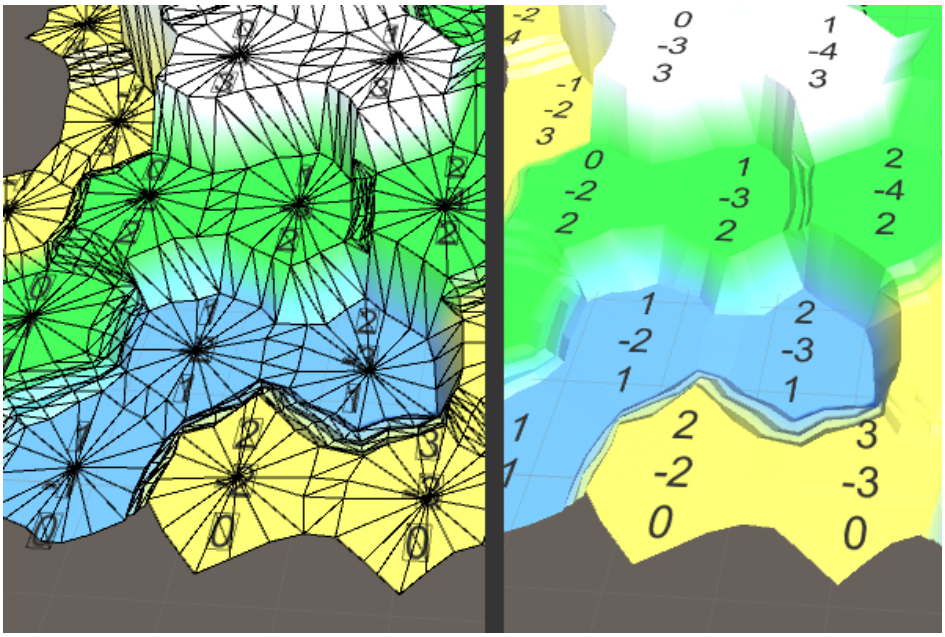


Figure 6-9 Generated terrain with sub-elevation terrace

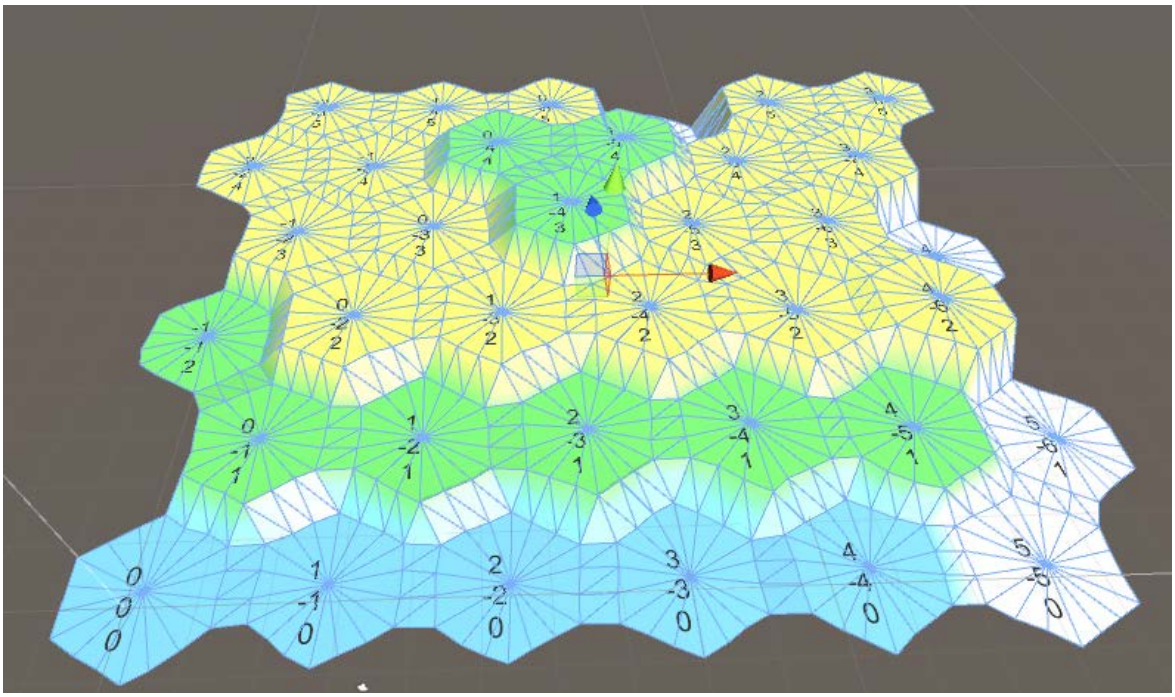


Figure 6-10 Hex Grid with additional subdivision step in each cell

Before we continue, our current design faced a limitation from unity3D mesh data structure that the maximum number of vertices had been limited to 65535 coordinates. To accommodate larger map details, map had been partitioned into a fixed size rectangular shape of a chunk, which each chunk held all vertices of assigned cell using mesh component of Unity3D. From this solution, it enabled our framework to generate a larger map. In addition, the major benefit from chunk was it provided a localized group of cells, thus, improving performance of any updating procedure for hex-cell redrawing since only a local mesh structure had to be regenerated again instead of whole map terrain. Comparatively, Figure 6-11 displays the total vertices that

must be recalculated for position for any single cell parameter modification while, in Figure 6-12, shows that only a small subset of grid in cell's parent chunk is required.

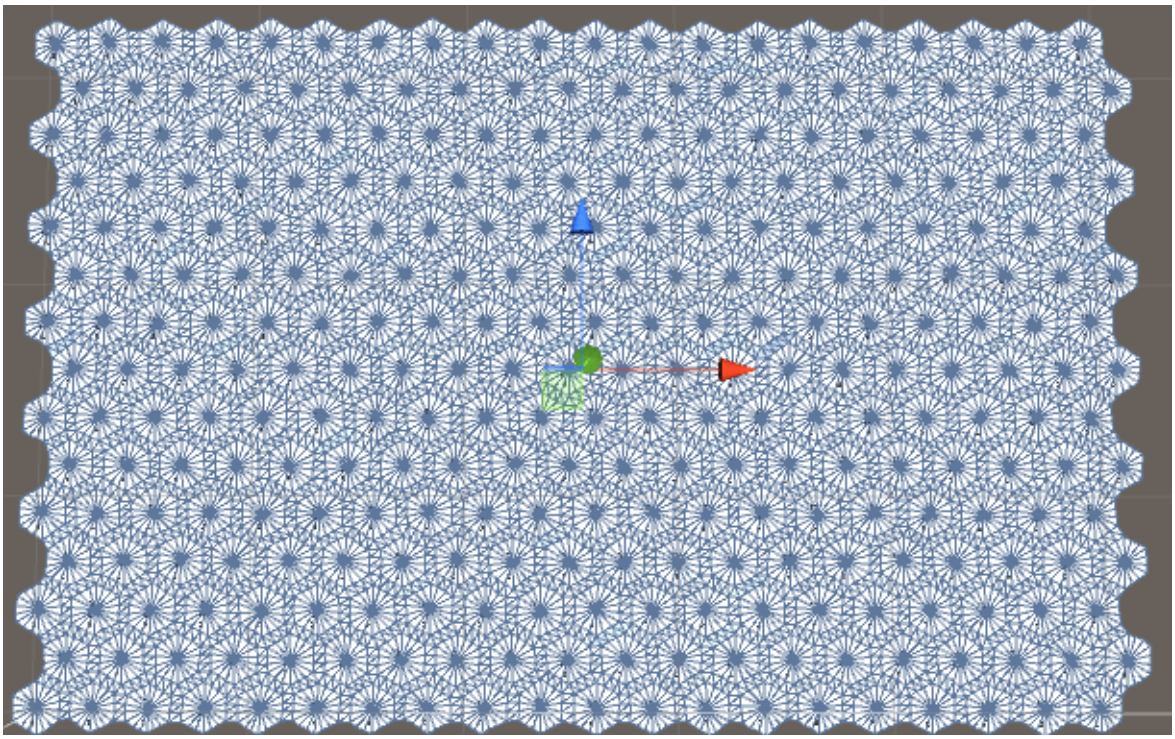


Figure 6-11 Hexagon Grid highlighting all vertices without partitioning into chunk

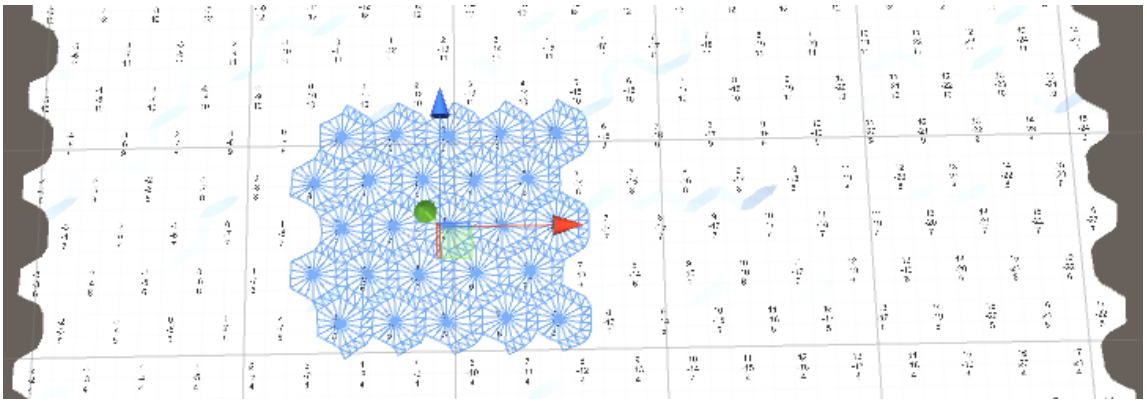


Figure 6-12 Hexagon grid with highlighting all vertices in a local chunk

From this stage, additional terrain parameters such as river, road and water mass had been implemented respectively by modifying previous cell surface generation procedure. Figure 6-13 demonstrates an altered surface structure from recalculating the related vertices for river. In Figure 6-14, the direction of river between cells influenced how generation procedure reacted in final surface and, finally, its water surface had been added on top with moving noise texture.

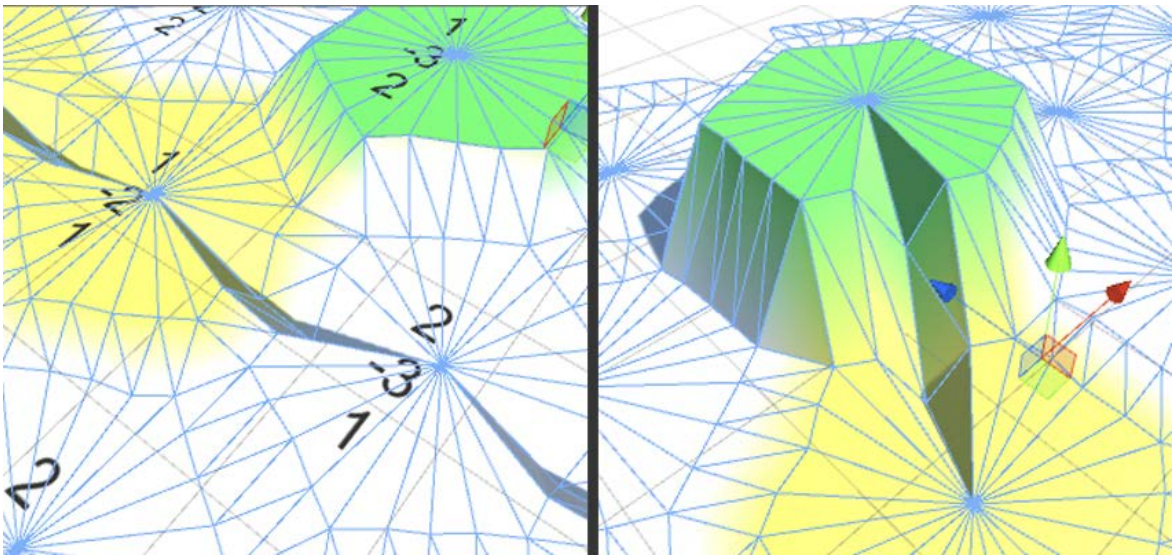


Figure 6-13 River canal coordinates generation: (left) on plain surface (right) on different elevation surface

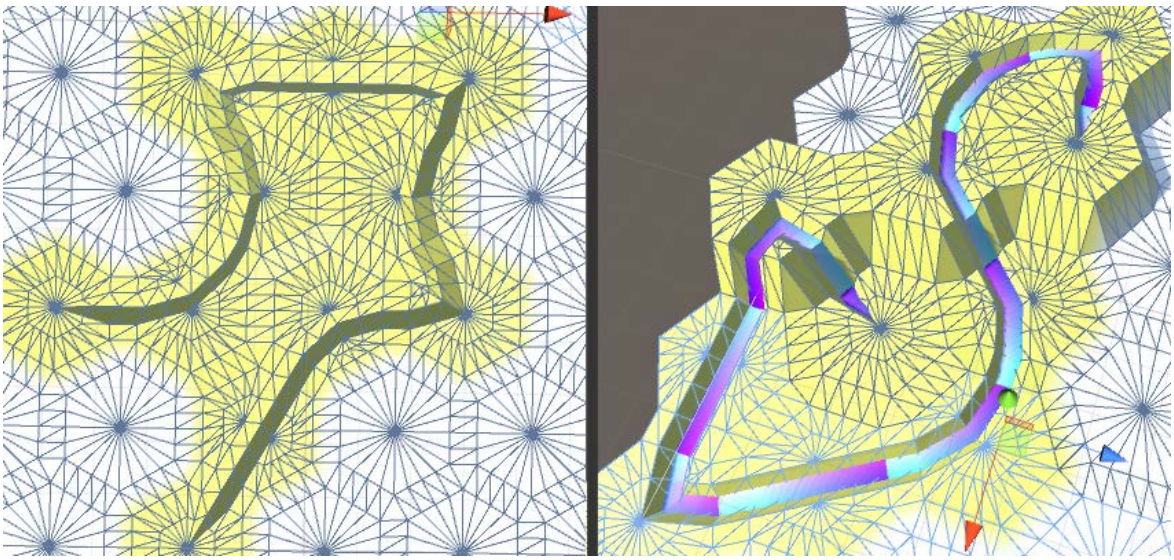


Figure 6-14 Example of surface mesh with different river direction (left) and the generation of river water flow mesh on top of the canal (right)

In contrast, road had been introduced by parsing its direction parameter in each cell then the drawing procedure only have to assign a road texture to the corresponding uv-coordinates in its path. Figure 6-15 shows the assignment of vertices with road texture on its path along slide with ongoing river. Generating a river resulting in terrain mesh change and it would interfere with road generation process on the same direction of cell, our system constrained that there would be no road in the same direction with river on one cell. The combined result from both river and road was displayed in Figure 6-16. From our data structure design, the vertex information forming a water surface of river and uv-coordinates for road textures had been kept in their own separated chunk layers on top of the original landscape mesh. It provided an ease of update drawing technique and textures.

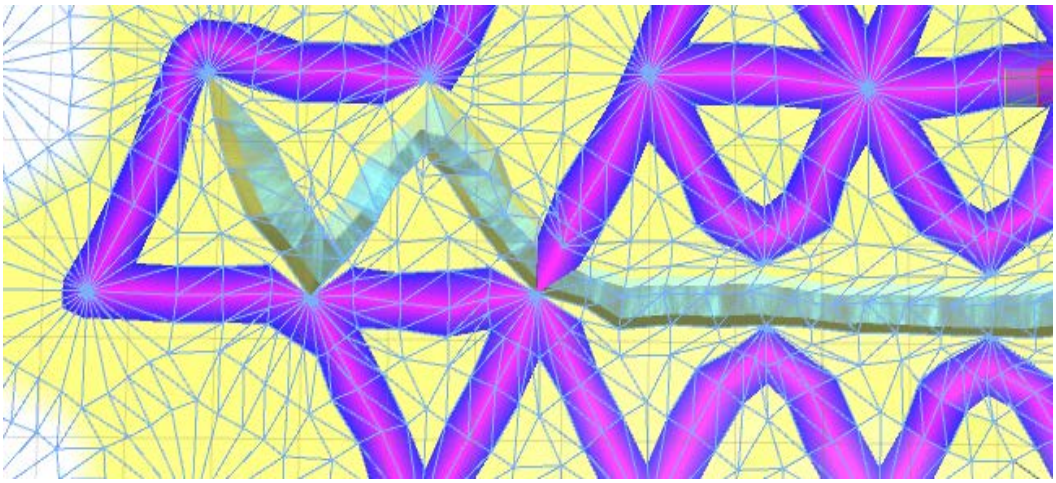


Figure 6-15 Assignment of road texture along with the river

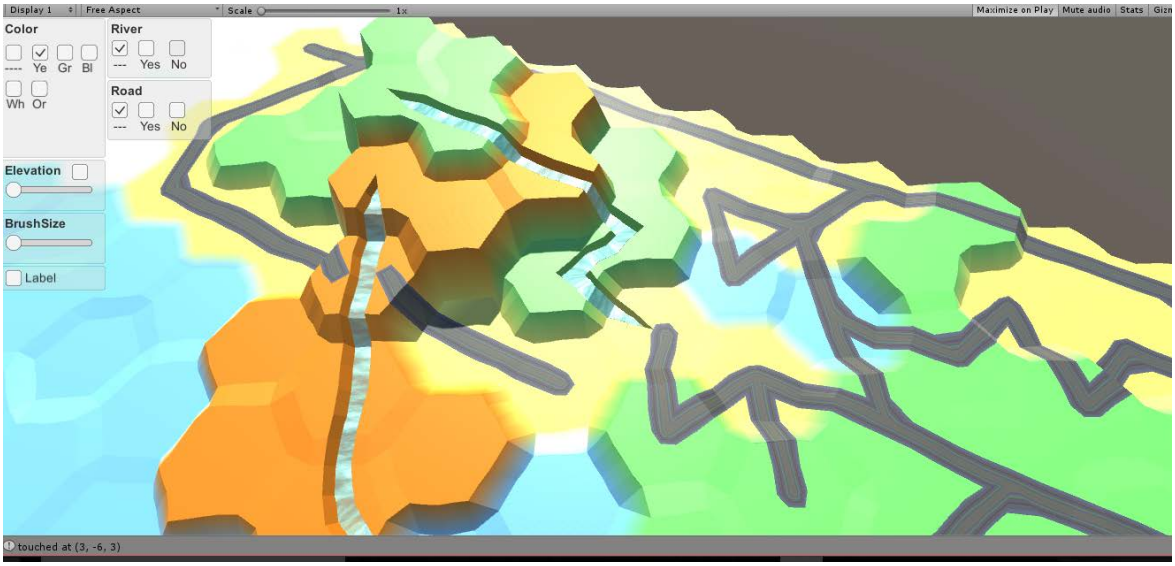


Figure 6-16 The rendered version of terrain with river and road textures

Next, water surface had been added on a cell with higher water mass than its land mass, which is being calculated as water level similar to elevation. Figure 6-17 (left) demonstrates how a water surface had been generated with similar method. The lid mesh required lower resolution except a shoreline perimeter where water surface of underneath cell connected to neighbour with higher elevation. Finally, a water texture was assigned procedurally by computing a blue colour with a noise function to add a form-like characteristic resulting in Figure 6-17 (right).

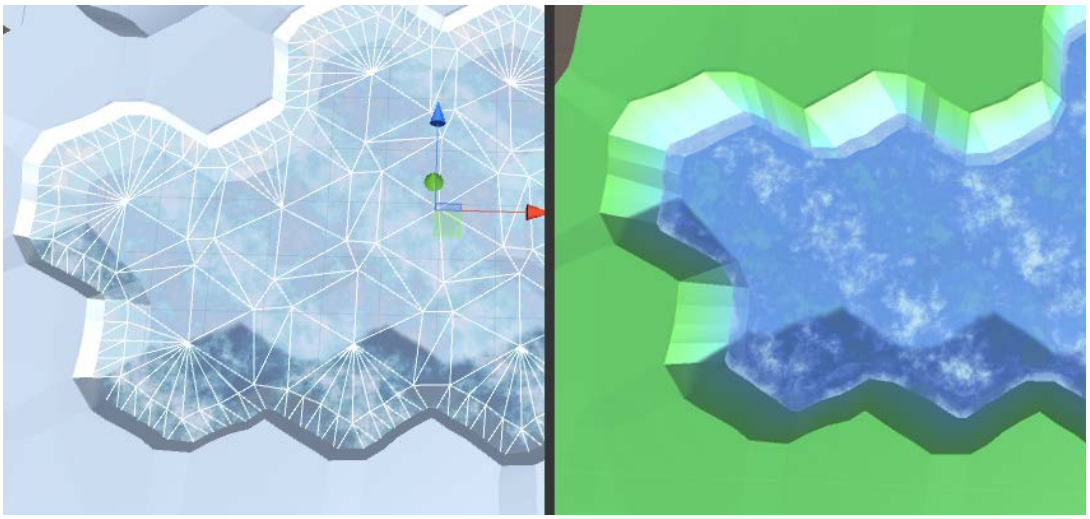


Figure 6-17 New water surface mesh has been calculated to fit the lid of underneath cell and nearby shoreline neighbours (Left) and its final rendering result (Right)

Furthermore, we represented a quantity of population, floral, and facilities in cell's parameters. Starting on a process of selecting for suitable 3D assets, these cell's parameters had to be calculated into levels, similarly with elevation and water surface. While considering a possible placement position from our design in Figure 6-18 (left), there were roughly limited to six possible instance, one in each direction. Then the visible asset had been selected with random selection that higher level of parameters had a better probability weight and likely to be shown than another. Figure 6-18 shows an example of 3D model for floral on these six placements while making an exception on the river direction, which also includes a road as well for aesthetic purpose. Note that the middle area of cell was appropriated for displaying important representation separated from the decorative features, thus it was reserved for agent and facility 3D model placement

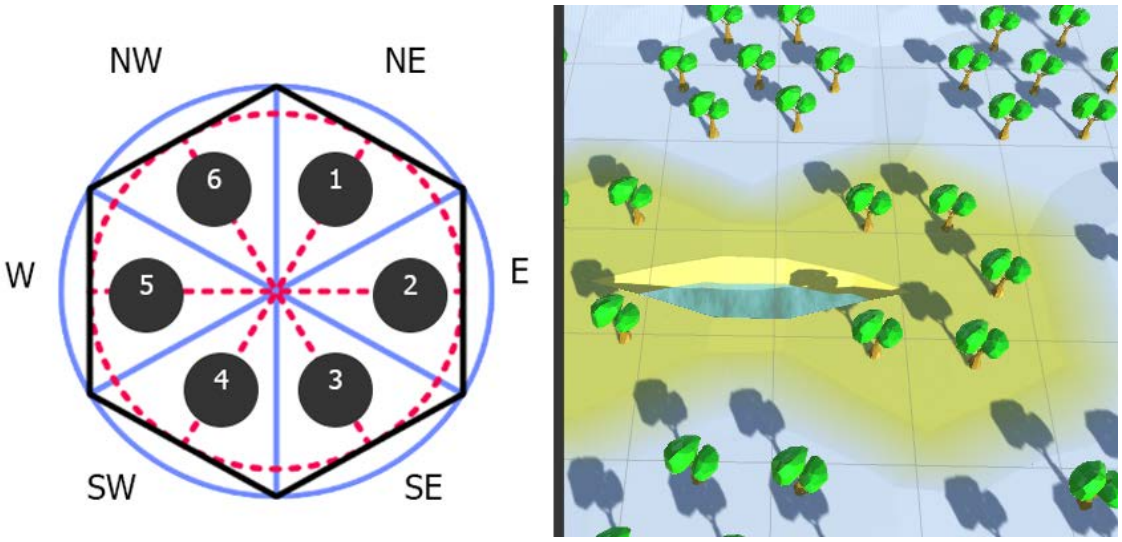


Figure 6-18 Possible 6 placement for 3D decorative models in a hexagonal cell while centre position is reserved for agent and facility model (Left) and the example of 3D model on those positions with an exception in river path (Right)

As a result, we concluded our framework based terrain generation procedure. All vertices data were kept in map manager component then being passed into Unity3D engine for each rendering update. By solving

new vertex coordinates procedurally on any change in terrain geography, it allowed real-time assignment of cell parameters from map editor interface, hence yielded flexibility for scenario-context designing process, and enabled integration of mathematic terrain generation model in the future extension. Figure 6-19 is an example of rendered result from all previous stages. As map size continued to expand, the selection of low vertex-count 3D assets was necessary to ensure real-time rendering application. In this regard, the post-processing filters also had an effect on a framerate for it requires additional render iteration loops. We experimented with several post-processing such as a Gaussian blur and tone mapping. These filters reduced framerate of the prototype by roughly 3-5 fps. For only simulation purpose, the post-processing was unnecessary aside form aesthetic purpose to create an immersive environment visualization. Therefore, the evaluation and scalability of the system were apparently dependent on trade-off in aesthetic layer during run-time. Figure 6-20 shows alternative result assigned with more vertex count 3D assets.

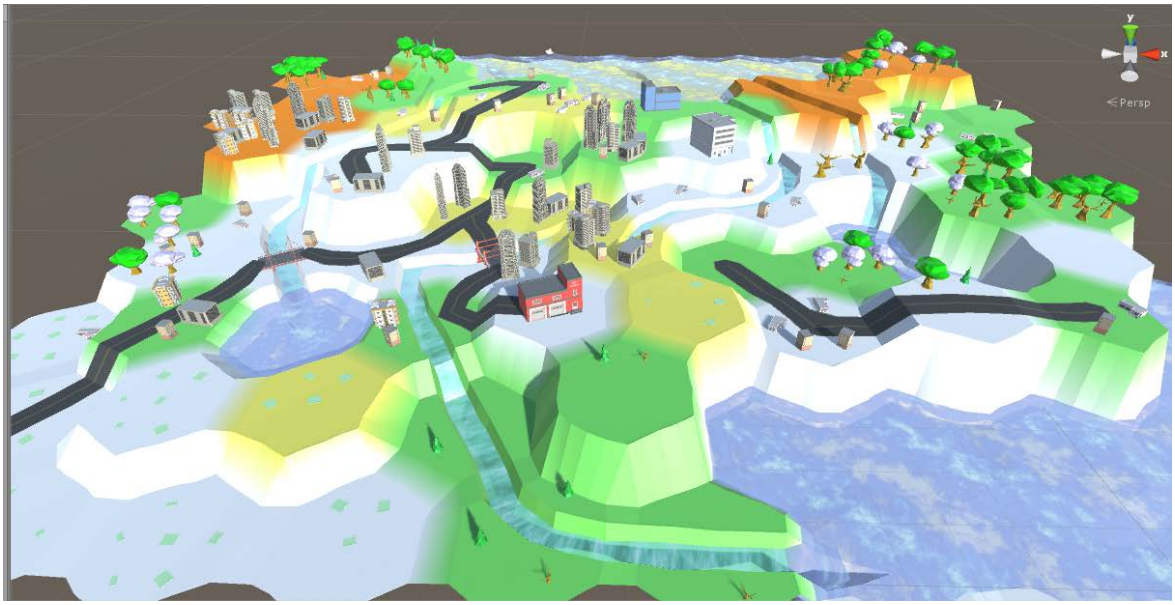


Figure 6-19 the combined result from terrain generation process: surface, elevation, water, river, road, and 3D decorative features.

Figure 6-20 Generated scene with details 3D asset (Left) and Large map replicate world region from far-zoom (Right)

6.3 Crisis simulation model development

In this section, we focus on experimentation to represent a general model for any large-scale natural disaster while developing a specific-disaster test model for the framework.

Fundamentally, major large-scale disaster models had based their approach based on classic elemental cellular automata (CA), which the basic property was described using a number of spatial dimensions, width and height of each side, a count of neighbourhood model for each cell, possible cell states, and, finally, a local rule representing as a transition function (R). In addition, the involvement of global parameters mentioned in previous work was another necessary step to achieve more realistic and flexible simulation configuration. To map between disaster models with CA, the desired event had to be parameterized for producing a relationship rule between each cell. Iterative rule testing was also possible for applying multiple state changes for different disasters at the same time step.

From study of an example in CA model, one portion of our literates had applied 2D square-grid representation with separated ruleset for each single disaster model. In general, for crisis scenario simulation, the rule design concerned not to produce a very unrealistic behaviour especially a fast-pace oscillating between patterns mention in Figure 2-8, which was created from a basic rule of dead-or-alive observed in Game of Life example. In the opposite, a ruleset producing a pattern that enabled state change moving in momentum of increasing or decreasing parameters had been a suitable model for natural phenomenon such as fire, water, lava, landside and, even, contingent disease outbreak. In consequence, it was advisable to experiment running CA finding a reasonable ruleset for different crisis.

6.3.1 Experimentation using cellular automata as crisis model for fire incident

To demonstrate how ruleset can be identified and applied into our simulation manager, we had developed a test case, which was applied with our CA ruleset for fire crisis. While rule was defined for abstraction a physical model of phenomenon, the goal of a usable design was to not over parameterize the model than necessary since it increased the processing complexity exponentially.

A pre-processing of real-world data required to map an abstracted size of space into CA grid cell. As for the state representation, a progressive state design had to produce a logical description relevant to ongoing situation for each individual cell property. In our experiment, we initially defined an environment with 2D square grid, which each state in each cell had possible value of Normal, On-fire, and Burnt-down (Out of fuel) respectively, to capture a progressing of fire situation. In addition, a substate representing geological type such as plain land and water terrain of cell had been introduced for better description regarding infrastructure property. Finally, we represented a concept of fuel as available burnable sustenance for ongoing fire and burning rate in the cell. These parameters allowed a customizable calculation of live-step duration on a cell that is On-fire without external influenced action from player or agent. Thus, each cell had a description as:

$$Cell(\mathrm{x},\mathrm{y}) = \{Type, State, Fc, Br\}$$

Where:

- x and y represented the coordinate of square-grid cell position
- Type indicated landscape terrain property of the cell

- State identified the current situation regarding fire crisis
- Fc represented the calculated abstract capacity of fuel residing in cell
- Br represented the burning rate of fuel in current cell

From our test system, local interaction between each cell was assumed to be corresponding within all neighbours of the current cell. For example, the square-grid cell coordinate (x, y) had eight neighbours (*Moores neighbourhood* (Moore, 1962)) such that:

$$Neighbor(x,y) = \{(x-1,y+1),(x,y+1),(x+1,y+1),(x-1,y),(x+1,y),(x-1,y-1),(x,y-1),(x+1,y+1)\}$$

With this setup, we included the information of infrastructure and geography to be mapped and pre-processed into cell's substate property. Depending on the pre-processing formula, we initially interpreted them the assigned each cell a fuel capacity. Furthermore, to simplify the control of fire spreading direction, we set terrain type of water to have a characteristic of zero fuel capacity. At each simulation step, rules for state transition had been applied to calculate the next state of each cell according to the previous mentioned model as shown in the Algorithm 1.

Algorithm 1 Wildfire CA (Simple)

```
1: procedure Simulate Fire State
2:     if cell(x,y)'s Type = Land then Continue
3:     if cell(x,y)'s State = On-Fire & Fc > 0 then
4:         Fc = Fc - Br
5:     if Fc <= 0 then
6:         cell (x, y)'s State = Burnt-down
7:     if cell(x, y) 's Fc > 0 then
8:         cell (x, y)'s State = Normal
9:     N ← Count(On-fire State) in neighbors(x, y)
10:    if N > 2 & N < 5 & (cell(x, y)'s State) = Normal then
11:        cell(x, y)'s State = On-fire
```

Algorithm 1 Wildfire CA (Simple) algorithm

With these basic ruleset, the behaviour of fire was easily simulated while it reflected on the concept of burning rate and fuel capacity. The additional calculation could be inserted to create sophisticated movement of fire direction based on how the fuel is plotted in the grid. With manipulation variable in step four, the extension controlling velocity of fire propagation could be extended. Still, we focused the result on this present ruleset rather than introducing over-complicated experimentation model. Moreover, our test had included a basic design of agent, which was a firefighter in this case, with a property of location existing in a valid cell (x, y).

$$Firefighter(x,y) = \{State, Health\ point(HP)\}$$

For each firefighter, an agent selected to execute one suitable action corresponding to its related position with fire using prescript if-else conditions in each simulation step. The following explained agent's actions in detail:

- **Extinguish** (x, y): Agent can extinguish any fire within one unit distance to any Neighbours(x, y) if destination cell's state is On-fire. This signature action represents possible interaction between firefighter and fire scenario. It allows observation of crisis outcome between firefighter efforts to put down a fire while fire continue to spread out on any terrain with fuel capacity value higher than zero.
- **MoveTo** (x, y): Agent can move one unit distance to any Neighbours(x, y) if destination cell's type is Land Terrain, thus Water Terrain will not be considered a valid destination, and it represents basic obstacle for both fire and firefighter simulation.

While the firefighter is not limited to move to cell with state is On-fire, it requires another ruleset condition to control his decision-making not to blindly walk into fire himself, or move randomly away from fire. Therefore, in each simulation, these following rules is applied for firefighter state transition shown in Algorithm 2.

Algorithm 2 Fire Fighter Behaviour

```
1: procedure Simulate Firefighter State
2:     if Firefighter(x,y)'s State = Alive then Continue
3:     N ← Count(On-Fire state from neighbor(x, y))
4:     if N > 4 then
5:         Firefighter(x,y)'s HP --
6:     if Firefighter(x,y)'s HP <= 0 then
7:         Firefighterell (x, y)'s State = Dead
```

Algorithm 2 Firefighter Behaviour algorithm

From the firefighter's state transition rule, the decision-making condition in each agent should avoid moving into a cell consisting of more than four On-fire neighbours since it would lead to his dead.

Since the calculation of agent's behaviour and fire simulation were ongoing in each frame, applying A* pathfinding from agent to fire cell would consume too much computation time, the implementation of influence map was a cheaper alternative for optimization of firefighter decision-making logic and motion for this test model.

Influence map was a buffer data storing information regarding a specific location it was represented. Its data structure could be varied from spatial square grid to area graph. Map was calculated for different stakeholder since it stored with various critical information for agent to process and use to make a suitable action. The first step was to set a spatial partition of environment space into corresponding same-size or greater data object, the relationship between each cell was then calculated and being added into the influence variable which allowed a single floating number or a vector of parameters. For our application, each cell (x, y) had an Influence (x, y) as the following:

$$Influence(x, y) = \{D, Nf\}$$

Where:

- D specifies distance from the current cell to nearest cell that is On-fire using Manhattan distance.
- Nf is count of Neighbours (x, y) having On-fire state including cell (x, y) itself

Agent took advantage of this information for their pathfinding steering function so it moved to a cell with less distance to fire and with high count of On-fire neighbours resulting in efficient movement to extinguish the

most On-fire cell. We also designed that agent would avoid a cell that has too high Nf since it was likely to incapacitate agent himself. Figure 6-21 illustrates the possible influence value in each cell. We controlled an overall behaviour of CA and agents by allowing involvement of global parameters. These values could be an initial value of agent's health point, a number of maximum firefighter randomly inserted into the scene, base number of burning rate, modification of count threshold in ruleset of cell and agent to cause progressive fire state transition and inflicting damage, and scale factor of damage to agent.

The implementation of this experiment was initially designed to give proof a potential of CA to mode a fire crisis. We selected Processing2D as lightweight prototype visualization framework written Java is suitable for this purpose as a developing tool. The simulation of environment in CA and agent behaviour was executed in limited update loop of simulation step. Figure 6-22 describes the colour reference representing state of cell and positioning of firefighter.

Figure 6-21 Influence map variable (D, Nf) on each cell corresponding to its location

land	water	fire	burnt	firefighter

Figure 6-22 Colour reference in 2D representation of fire CA simulation result

To demonstrate how agent acquire information regarding a destination cell with most fire neighbours, displays a visualization mapping between map and its corresponding influence map. Black spot indicated that the cell had less or close to zero influence while white cells was important and required immediate priority.

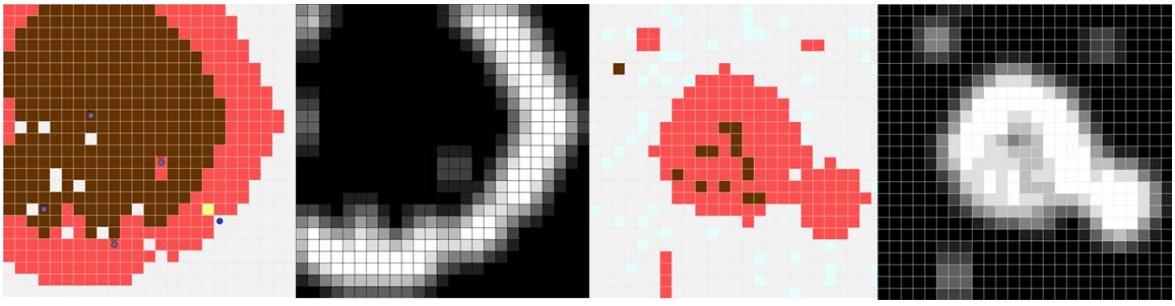


Figure 6-23 Fire CA simulations and their normalized influence map

6.3.1.1 Test Case 1, No agent intervention

Shown in Figure 6-24, we demonstrated the situation of fire starting in four different patterns across the area without intervention of any firefighter. Within early 13 steps, the circular fire front could be observed in the

middle of map area whist its innermost starting fire cells had already been out of fuel. The generated result reflected a natural behaviour of unstoppable wildfire spreading, thus, it give a small validation on using CA as a base technique for simulating fire behaviour. In addition, as it was a preliminary model to demonstrate the given of applicable rule, some fire had not trigger its state transition due to the lack of enough fire neighbours. Finally, at step 103, most cells were burnt down hence ended our first simulation test

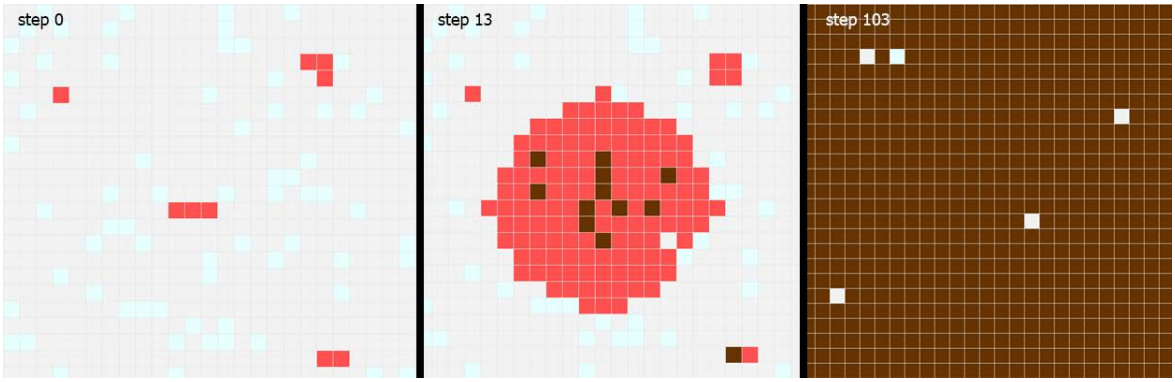


Figure 6-24 Basic fire spreading without agents intervention: FPS =60, Total generation step is 103, Burnt down cells are counted as 620 units

6.3.1.2 Test Case 2, Sufficient agent intervention

Shown in Figure 6-25, the starting fire was in a small setup under perimeter of well-positioned firefighters. In step 7, all agents collaboratively cleared fire cell with minor damage of two burnt-down areas. The burnt-down cell state occurred from enduring a few step of On-fire state due to small capacity of starting fuel. From the test result, it had shown a resolution of fire incident within minimal simulation steps, thus this small test case represented an ideal situation where well-timely placed resources could contribute to damage reduction during ongoing crisis.

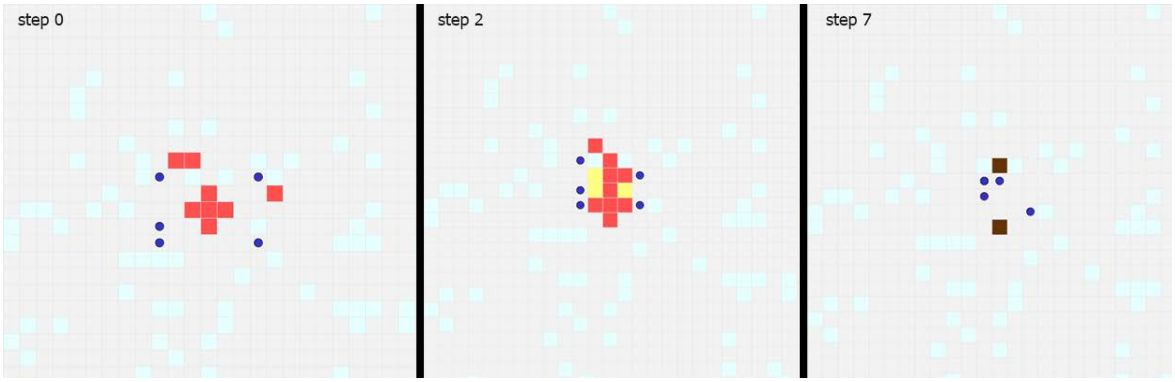


Figure 6-25 Simulation setup where fire is surrounded by sufficient firefighters: FPS = 60, Total simulation step is 7, Burnt-down area is 2 units. Extinguish actioned 24 times.

6.3.1.3 Test Case 3, Insufficient agent intervention

Shown in Figure 6-26, the initial fire location were separated into two starting areas, and the assignment number of firefighters had been different for both sites. This experiment would produce a variation of results sometimes due to randomness of agent's decision so we decided to emphasis on one such possible outcome. Noted that our map was designed to be able traverse from left corner into right corner connectively. From step

30, we observed the disappearance of fire cell at the lower right corner due to sufficient deployment of firefighters while the influence of fire on the rest were getting out of control from the upper left corner of the map then spread horizontally across to the its connected right-sided sections. In the end at step 115, the situation left only catastrophic damage of everything almost become burnt-down state and most agents become dead.

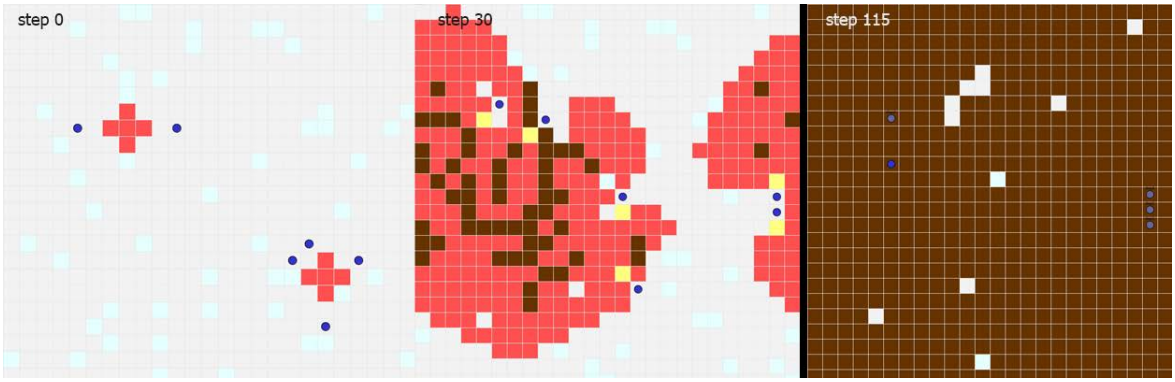


Figure 6-26 Setup scene with two location of ongoing fire sites with agent assigned in different numbers: FPS = 60, Total generation step = 115, Burnt-down area = 614. Extinguish actioned 221 times. Dead fireman = 5 and Alive = 1

From our technique, we introduced then demonstrated one possible design of crisis model using cellular automata (CA) for fire incident creation, simulation, and resolution. The system shown potential to simulate a behaviour of natural crisis such as wildfire with an input of abstract representation of terrain types with calculated parameter of fuel capacity. It also included a simulation of crisis agent deployment such as firefighters to resolve the emergency event. The sequence of simulation can be observed clearly with appropriate visualization framework such as Processing2D.

Therefore, the fire CA ruleset would be improved to be integrated into simulation manager in our proposed framework on Unity3D game engine. As a consequence, the final prototype system had been implemented and support fire crisis model. We concluded that cellular automata was an efficient technique to develop the modelling and simulating natural crisis situation involving a variety of environment setup for many possible outcomes. In addition, the given model would be used to study the fire propagation, the success of the firefighters to control the situation, and the risks involved to the firefighter health and life in case there was no enough resources allocated for the ongoing event. These aspect provided a quantitative parameters to evaluate the outcome for testing a hypothesis on crisis scenario.

6.3.2 Crisis model implementation for the framework in Unity3D

Evidently, our previous experiment crisis model was a compatible design with our map representation grid. In this subsection, we focused to extend further investigation the usage of CA for crisis scenario generation with the proposed framework.

6.3.2.1 Wildfire model

Extending from previous design, we introduced more parameters into the design such as temperature, humidity, global wind direction, new fuel capacity formula regarding existing quantity of

population and flora. For an example, rather than transiting On-fire state directly to an adjacent cell with ruleset for fire occurrence count, we had taken an aspect of heat accumulation. The final value was simplified into a tracking of temperature parameter until it reached a threshold to ignite On-fire state. Figure 6-27 shows a propagation of accumulated heat in each cell. Cells neighboured to fire incident received an increasing in its own temperature every timestep then got ignited when reached the threshold of 300 Celsius. Similarly, agent's action such as extinguish reduced this accumulated temperature of target cell. When the value of temperature had been reduced under the threshold, it was resulting in state transiting back to normal.

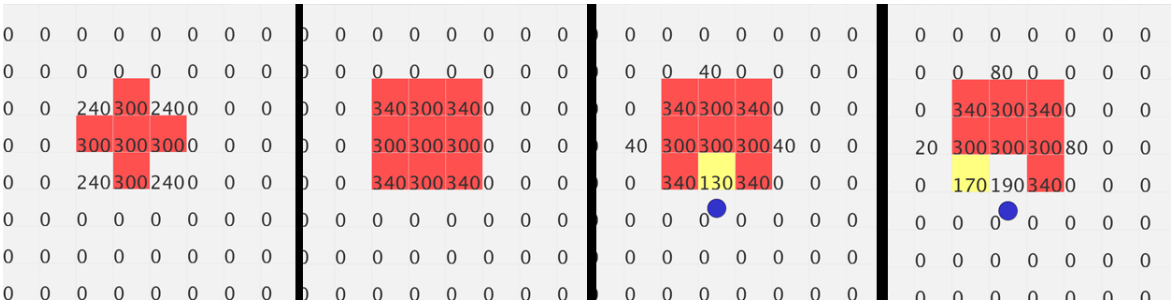


Figure 6-27 Heat parameters for triggering On-fire state and effect from agent extinguish action.

Integrating this crisis model into our framework, we had developed a fundamental component of CA model in simulation logic to acquire the information from map representation layer. With hexagonal cell grid, a transition function included a maximum of six neighbours from adjacent cells in each direction:

$$Neighbor(x,y,z) = \begin{Bmatrix} Cell_{xyz}(NE), Cell_{xyz}(E), Cell_{xyz}(SE), \\ Cell_{xyz}(SW), Cell_{xyz}(W), Cell_{xyz}(NW) \end{Bmatrix}$$

In collaborating with our 3D terrain editor with this heat transfer CA model, alteration of temperature from the editor caused a similar observation in fire cascading effect, which was visualized as fire state mapping corresponding particle system, shown in Figure 6-28.

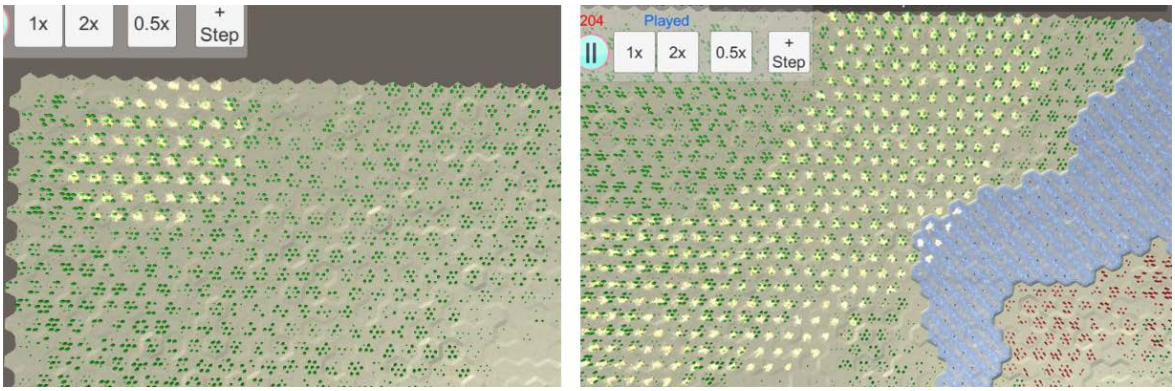


Figure 6-28 Fire crisis simulation tested with prototype framework in Unity3D

Next, we had further improved current crisis model with additional parameters. It enabled better realistic conditions for verifying an On-fire state. For instance, we had set a condition for any cell to trigger On-fire state in a condition that its accumulated temperature was exceeding starting fire threshold. We also checked that fire state would occur if there was less humidity to stop a fire progressing. Furthermore, cell was constrained that it would have a fire state under an influence of flooding or shown as underwater state. Importantly, the remaining fuel must more than zero for the fire state to continue over timestep.

$$\text{Cell(x,y)'s FireState} = (Temperature \geq fire\ Threshold)\ \boldsymbol{AND}\ (humidity < Stop\ Humidity\ Threshold)\ \boldsymbol{AND}\ (cell's\ underwater = false)\ \boldsymbol{AND}\ (Fc > 0)$$

With this new condition, the fuel capacity was designed to reflect an accumulated quantity of population and floral in the cell while both were depleting by constant damage percentage if the cell was On-fire state.

$$Cell(x,y)'s\ Fc = Population + Floral$$

$$Depleted\ fuel\ per\ step = (Population + Floral) * OnFireDamageFactor$$

Furthermore, we simulated the On-fire cell behaviour with this new following rule shown in Algorithm 3.

Algorithm 3 Wildfire CA (Advance)

```
1:  procedure Simulate Fire State
2:      if cell(x,y)'s Underwater = True then Skip the  calculation
3:  transfer heat loop:
4:      if cell(x,y)'s State! = On-Fire then
5:      goto self-evaluate.
6:      for Each cell(x,y)'s neighbor in each cell adjcent directions do
7:              Nb ← Neighbor cell in D direction
8:      if Nb's Fc > 0 then
9:              Nb's Temperature+ = Heat Transfer Rate
10:     else
11:             Nb's Temperature+ = Heat Transfer Rate for empty fuel cell
12:     if Nb's Direction = Global Wind Direction then
13:             Nb's Temperature+ = Wind Speed * Heat Transfer Rate for Wind
14: self-evaluate:
15:     if cell(x,y)'s State = On-Fire & Fc > 0 then
16:             Fc = Fc - Br
17:     if Fc <= 0 then
18:             cell (x, y)'s State = Burnt-down
19:     if cell(x, y)'s Temperature >= Ignite Fire Temperature then
20:             cell(x, y)'s State = On-fire
21:     if cell(x, y)'s Temperature < Ignite Fire Temperature &
            cell(x, y)'s Fc > 0 then
22:             cell(x, y)'s State = Normal
```

Algorithm 3 Heat Transfer Fire CA Algorithm

From the modified formula, we took into account from all related existing cell parameters including a global parameter such as wind direction and wind speed. While still relied on CA structure, the change in state transition condition had been enhanced significantly from early primitive model into complex representation of fire crisis.

6.3.2.2 Flood model

In the same fashion, our current crisis CA logic can become a foundation for development of other large-scale disaster model such as flooding. Extending from previous cell evaluation logic, we have inserted procedure to simulate pressure distribution model of liquid using the following rule shown in Algorithm 4.

Algorithm 4 Flood CA

```
1:  procedure Simulate Water Pressure
2:      if cell(x,y)'s Elevation Level > cell(x,y)'s Water Level then
3:              Skip the calculation
4:  Equilibrium  Calculation:
5:      Nout = 0
6:      SumMassOut = 0
7:      Nlist = Empty list of cell
8:      for Each cell(x,y)'s neighbor in each cell adjcent directions do
9:              Nb ← Neighbor cell in D direction
10:             if cell(x, y)'s water mass - Nb's water mass  >=
                  Min flow Threshold(MinTh) then
11:                     Nout ← Nout + 1
12:                     SumMassOut ← SumMassOut + Nbs water mass
13:                     NList ← NList + Nb
14: Flow calculation:
15:     if Nout > 0 then
16:     Meq = (cell (x, y)s water mass + SumMassOut) / ( 1 +Nout )
17:     for Each cell Nb in Nlist do
18:             flow ← Meq - Nbs water mass
19:             if flow >= MinTh then
20:                     Nbs water mass = Nbs water mass + flow
21:                     cell(x,y)s water mass = cell(x,y)s water mass - flow
22:             if cell(x, y) Water Level > cell(x, y) Elevation Level then
23:                     cell (x, y)'s Underwater = True
```

Algorithm 4 Flood CA algorithm

The sequence of crisis behaviours had been synchronized so it ensured no interfere with other existing CA procedures. In our prototype framework, the water pressure simulation was triggered after heat transfer behaviour from fire crisis model, and the fire state would be converted back to normal state if cell was underwater. Moreover, there was no fuel parameter calculated in the flood model but water mass, ongoing flood event would add consistent percentage damage to any population and floral cell's parameters in similar formula to On-fire state. The assumption to reduce a population and environmental parameters was to ensure a reflection in visual representation change. We tracked these parameters for evaluation of hypothesis scenario.

While the developing another crisis model could work on using an existing or extended cell parameters in the map, an extra effort for modifying the visualization pipeline in procedural generation procedure was inevitable. In the framework prototype, adding a flooding crisis model meant that the corresponding visual representation had to be developed on top the previous terrain generation procedure. Figure 6-29 (Left) displays an adaption of terrain generation procedure in our framework to facilitate 3D visual representation of water volume in flooding crisis model. Finally, in Figure 6-29 (Right), our framework demonstrated its capability to simulate flood crisis model and its visualization.

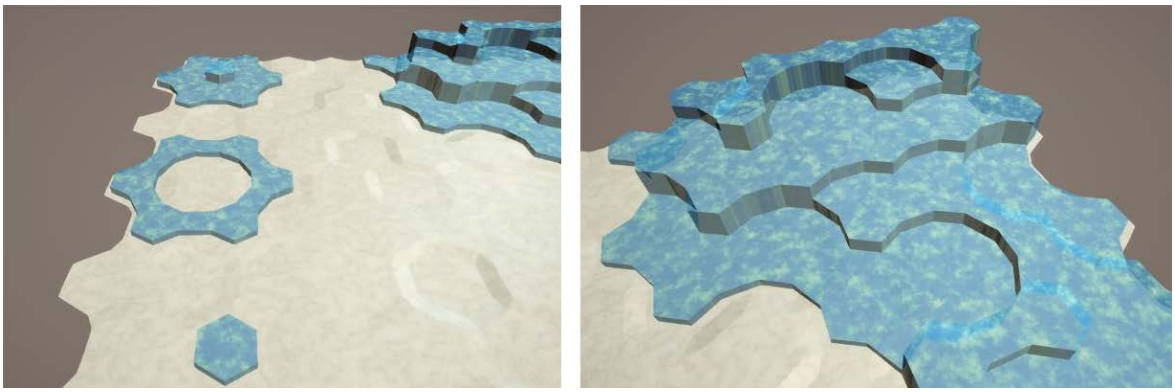


Figure 6-29 The procedural generation for visual mesh in flood crisis: Different type of water volume visually represented in on the terrain (Left), Stacking water volume visually represented on the terrain (Right)

As a result, we introduced a disaster model with its testing technique. Then we converted the finding into ruleset algorithms in the simulation manager component. Next, the generation of crisis scenario required a functionality to tailor crisis situations.

6.4 Scenario scripting for crisis event scheduling

Our framework provided a basic structure for creating a new crisis event template that its behaviour involved a change in parameters of area and related stakeholder agents in the simulation step. Then, each simulation step included a calculation of CA state evaluation, and re-update visualization procedure, mentioned in Figure 5-7. In Figure 6-30, it shows an example of how to extend a provided abstract crisis event class into new Heat Surge and Heavy Rain events. Importantly, parameters in an extended class should emphasize only a necessary factor for the state transition ruleset. For an example, the Heat Surge extends crisis event class, which would raise the area cell‘s temperature was instantiated with surgedAmount parameter of 50. It had been named with keyword “heatsure-50” with the effect of increasing cell temperature for 50 Celsius. Next, another “heatsurge-80” was created similarly using a template but with different applied effect. Furthermore, these created events would be registered as valid event templates in the crisis event scheduler during run-time. It allowed any upcoming schedule to be specify by template keyword, starting time and coverage area of events. By generating a set of event templates, they could be exported and reimported into another running client seamlessly using JSON parsing from the data manager. Then, we implemented our crisis model by extending event scheduler for a fire and flood situations. our framework provide a method to script crisis event behaviour and create a new instance of desired parameters to be registered in scheduling component as a template with ease as shown in Figure 6-31.

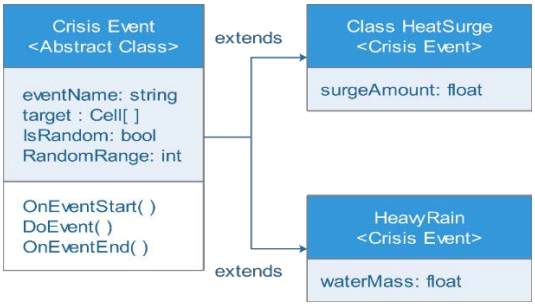


Figure 6-30 Crisis Event Template and its extension

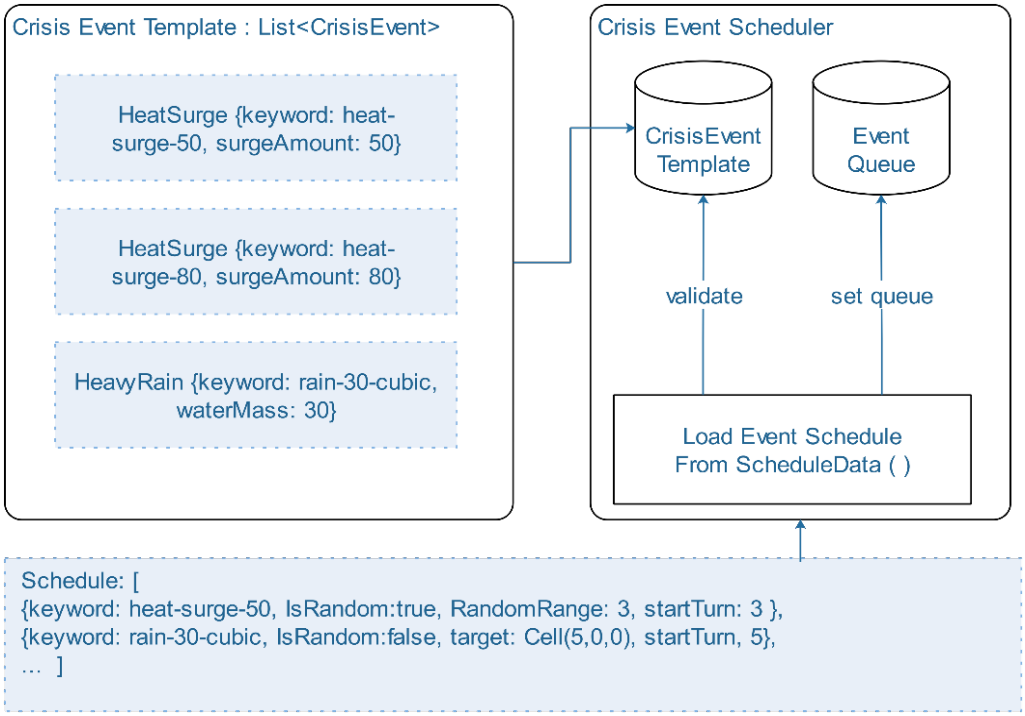


Figure 6-31 Crisis Event Scheduler Component

As a result, crisis scenario designer had a capacity to schedule a narrative sequence of crisis events with collection of alternated disaster situations. These set of predefined events could have a random starting affected area and specific occurring time. Furthermore, the crisis event scheduler was also capable of reuse already script scenario in different map context by loading an exported crisis sequence into the new event queue. Thus, the component allowed an effective way to test similar situations without reassigning a new schedule again.

6.5 Agent Implementation

6.5.1 Unit and unit action

The implementation of agent' actions in our framework was visualized in Figure 6-32. Unit was a general class for each agent while UnitAction was an abstract class representing doing an action from the agent. By implementing UnitAction class into a specific action, agent's behaviour could be constructed by tailoring a decision-making logic to execute these actions.

The Unit class owned common game-based parameters for representing its capabilities while the distinct factor between each setup was residing in unit type and a list of valid actions. In practice, developer and crisis scenario designer had to communicate how to convert the real-wold knowledge into corrected action extended the template of Unit Action. The examples of these actions were any specific manipulation of unit's parameters, related cell area, or another unit while some would require a prerequisite for suitable world state or agent's role. While the framework allowed us to predefine our own implementation of interaction, these implemented actions could have a common usage over different unit types. Figure 6-33 demonstrates a variation in unit setup to represent game actors with separated set of available actions.

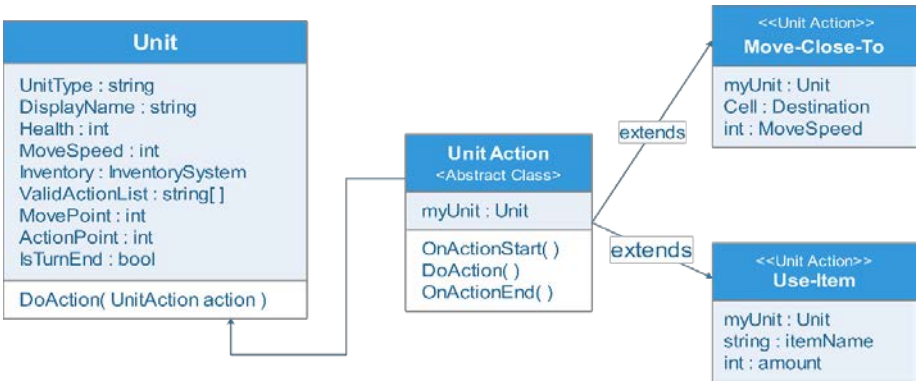


Figure 6-32 Unit and Unit Action class template an example of extended unit actions

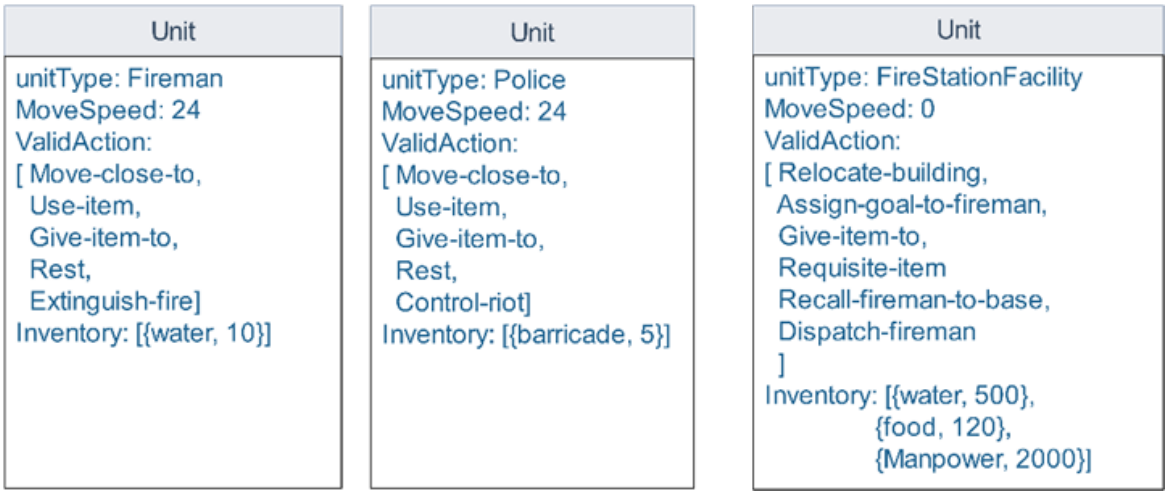


Figure 6-33 Example of Unit parameters for different unit actors

6.5.2 Agent behaviour component

From our design, the agents had an agent behaviour component as an abstract interface. It provided a general access to decision-making process of automated agent via a common API calls as AutomateDoAction(). The game manager or simulation manager could simply trigger the automate decision-making then waited for an agent suitable response regardless of its type.

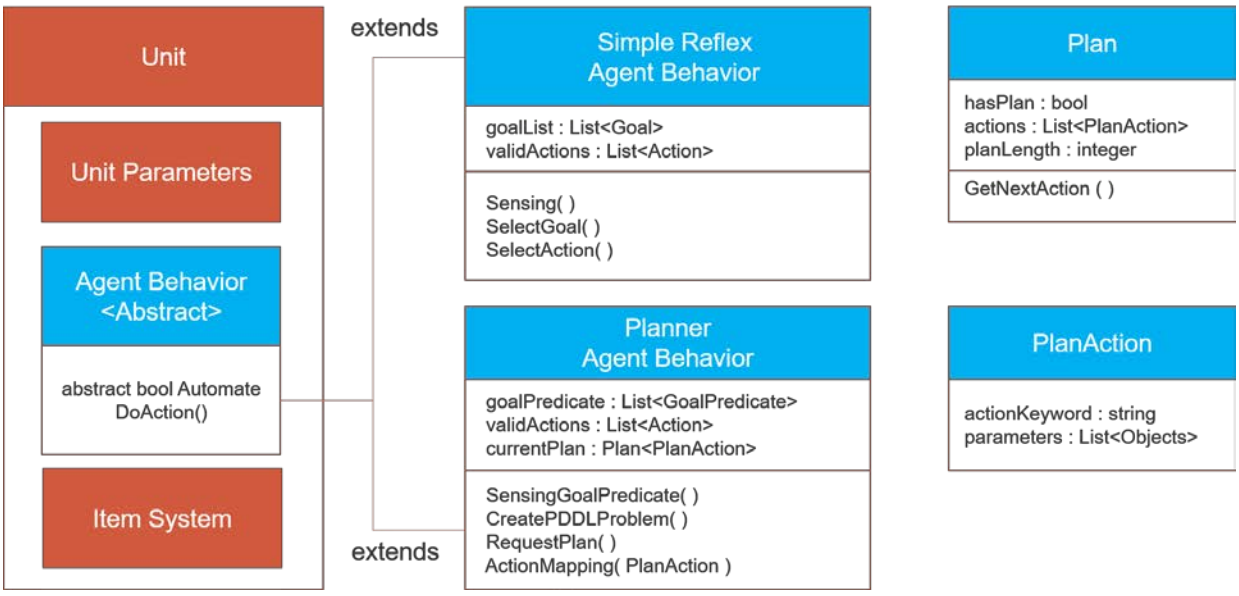


Figure 6-34 Agent behaviour extension for AI planning

To provide more details, the planner agent was working on a different concept of IF-ELSE conditional syntax. It parsed the word environment information, requested a solution plan, and executed action referred in plan action list. The variable for AI planning had to be represented in predicates. As a note, predicate was a semantic state representation in STRIP and PDDL so a keyword was mapped with related any number of parameters such as object name. Figure 6-35 shows a class diagram to represent predicate and problem in our framework that was implemented for a fire incident domain. Moreover, the plan action could be derived from a predicate representing action as a solution from planner so the planner agent requires performing an action mapping to its corresponding unit action script implemented in Unity3D.

Figure 6-36 shows an overall process of plan generation for planner agent behaviour in our implementation phase. The communication between Unity3D's client and web service were implemented using an interchangeable JSON format. We had separated the planning into external web service for it gave flexibility of changing executable AI planner and tracking its result without recompile a framework in Unity3d. To begin a planning procedure, agent planner behaviour had to generate its problem file in PDDL then used an API calls from a planner manager to create a HTTP request toward our planner web service. Next, a PHP interface processed any incoming plan request in JSON format then parsed the keyword for related variables and its requested domain file. In generating a plan, the PHP script invoked our implemented Fast Forward Planner (FF) wrapped in Java language (JavaFF) to find a solution. Parsing the result from JavaFF output, the planner service created a response *Plan.json*, which, sometimes, could contains zero length plan. The meaning of zero size plan was meant to be that the solution was not found. Moreover, the planner manager logic had to process the HTTP response into framework's Plan object in Unity3D data structure before transferred the plan back to agent planner behaviour. From this step, the agent mapped the action keyword in the plan with its own registered unit action to create a same execution order as the plan.

With web service extension, the verification process for requested problem and planning execution result could be tracked for every individual agent on a the web application. In our implementation, the framework allowed a firefighter agent as a lowest tier stakeholder an ability to take advantage of AI planning architecture. Likewise, it could be implemented for other higher tiers with different problem generation

algorithm as the goal would involve strategic manoeuvre of resources. Reminding the performance issue with using AI planning, while it was possible to equip every agent in the crisis model with planning agent component, the HTTP request procedure could become a bottleneck for real-time application. Another issue was how agent could generate a problem with minimal required predicates efficiently. Unfortunately, the optimization of agent planning algorithm was not in the scope of the thesis. With the given implemented AI planning process and agent behaviour model, it was still suitable to use planning agent as only for small group of strategic decision-making units such as city governing unit.

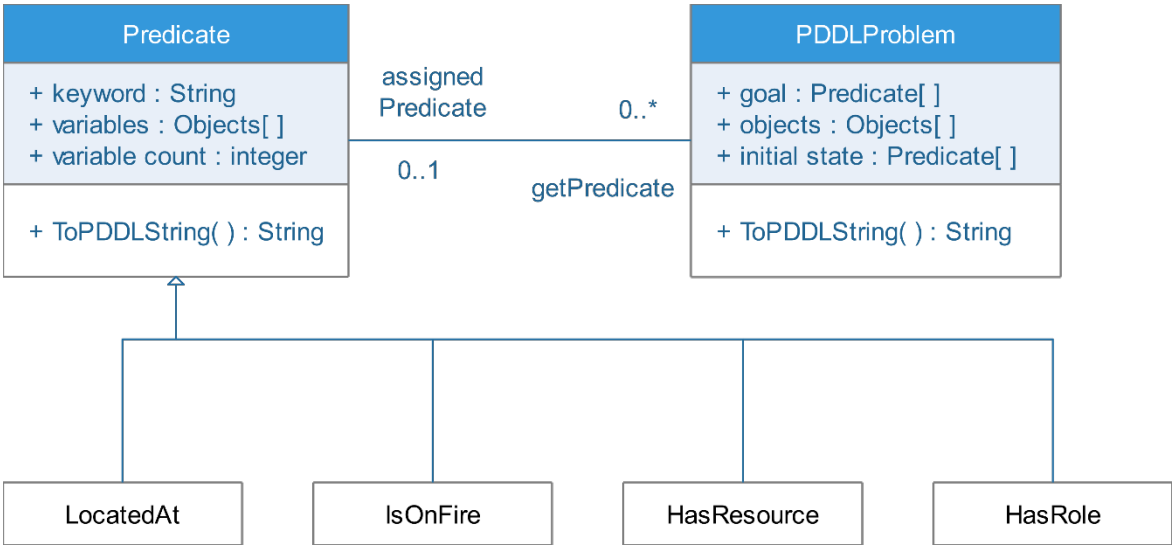


Figure 6-35 Predicate class diagram

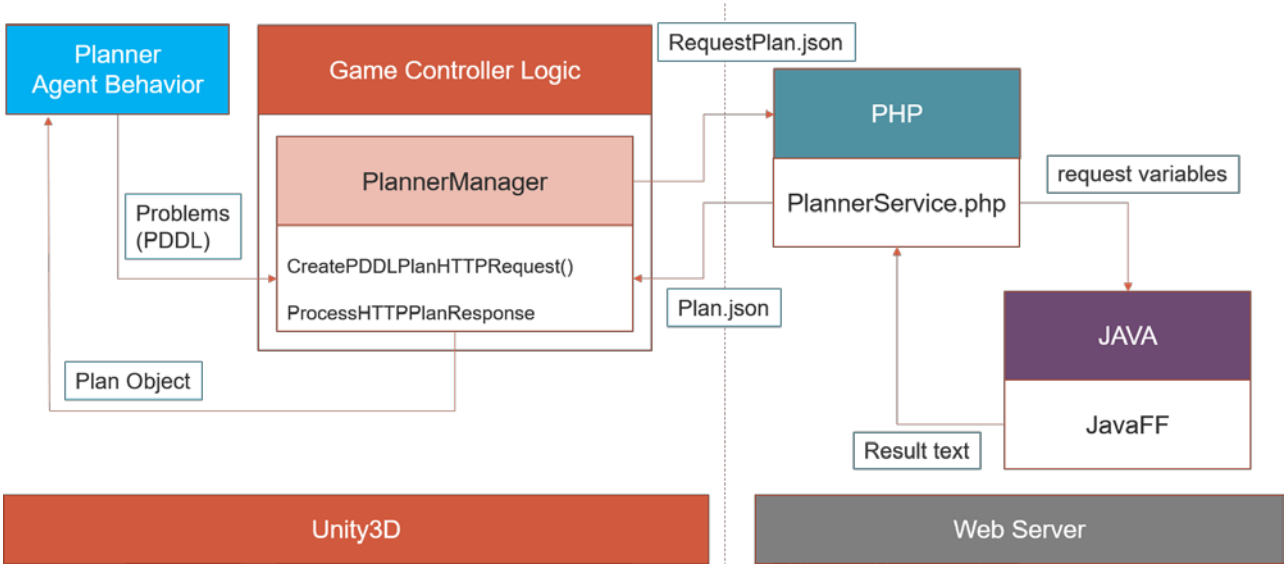


Figure 6-36 Process of request plan from Unity3D to web server

In conclusion, we demonstrated an extension of agent component for AI planning behaviour using web service to enhance flexibility in assigning new domain, changing planner and tracking the result from every agent. The pipeline operated seamlessly with a unit equipped with agent behaviour component in Unity3D. As a result, crisis scenario designer could start testing their model from implementing an IF-ELSE as simple reflex agent then empowered their simulation by adding planning agent for manager-type agent.

6.6 Summary

In this chapter, we explain how each main component in the CGSA-SIM was implemented according to the given specification and design.

First, the map representation has been implemented in a hexagon grid data structure using Unit3D game engine. The coordinates representing an area location has been generated procedurally for vertices to form a hexagon shape and its volume. By adjusting a visualization procedure to reflect a level of each corresponding cell parameters such as land mass to height, water mass to sea surface level, floral to tree feature placement, the addition of more relevance crisis parameters has been easily applied with the similar process. While the procedural generation of terrain is controlled by script, changing a visual feature is simply by configuring on parameters of area topology then the visualization pipeline reflects the parameter changes into a new terrain.

Next, the crisis simulation model for the framework has been developed from an early experiment of our fire incident using a simple ruleset in 2D Cellular automata. The preliminary results from a Processing2D prototype shows that crisis incident could be modelled and simulated for an observable impact with agents effectively. Then we demonstrate the design process for more sophisticated models to our framework. As a result, wildfire simulation using additional parameters of heat transfer and wind direction has been added alongside with flood hydraulics model based on the liquid pressure equilibrium concept with an adjustment on water-volume visualization procedures.

Moreover, we explain how a narrative sequence for crisis scenario can be structured then parsed into our event scheduler. The interface for parsing and reusing a crisis script allows fast experimentation with different disaster script to test the impact on different generated environments without redone from the scratch.

Lastly, the implementation of agent as a stakeholder in crisis situation has been explained. The role of agent is made explicit by assigning a different list of valid actions and role parameters. By focusing on the reusability, the implementation of agent's action allows being reused for any agent in the system as long as it is compatible with action's constrains. Importantly, we demonstrate how a planning agent could use a provided library of predicate to generate a PDDL version of their goal logic then communicated with a planner service externally from the Unity3D. With our AI planning process outside of Uniy3D application, developing a new domain and verify a plan solution have become more flexible without requirement to recompile a whole project.

In the next chapter, the results from the final application are presented. they include a performance evaluation and a testing on crisis scenario generation.

7 EVALUATIONS AND RESULTS

7.1 Introduction

In this chapter, the run-time results from a framework are presented. We provide a performance report for overall system and its related scenario example. The evaluations are separated into two main tests. First, Scalability evaluation has been measured for system performance with real-time aspect. Second, the scenario test has been conducted to observe the granularity of crisis model and its agents' integration.

7.2 Testing and Evaluation

We discuss the testing methodology and evaluate the results. To evaluate our system quantitatively, the scalability test has been performed for game-based application aspects consisting of visualization, game initialization, simulation time, and agent decision-making time. To test whether system is suitable for modelling and testing crisis model, a test case on crisis scenario has been conducted.

7.2.1 Scalability Evaluation

The report focused on a result of terrain generation and visualization processes while tracking an agent response time. The framework experiments were run in a Laptop with Core i7-5500U @2.40GHz, 2 Cores with RAM 16GB (10GB Available), and Window 10 with GeForce 840m model.

Figure 7-1 shows the overall visualization performance in framerate resulting from parsing terrain information on (a.) standard gameplay camera-zoom and (b.) custom panning camera trying to cover as much as possible map contents regarding the different size of terrain setups. The framerate measurement performed with the assumptions of limiting framerate to 60 on rendered screen size of 1280 x 800 pixels while the map had proportion of 20% plantation, 20% city and 20% water area equally on all map sizes. In the standard gameplay camera-zoom, the framerate was stable and could maintain its playability as close to target 60 FPS with minor drop to an average of 50 in 100 x100 map setup, total 10,000 cells. In far-pan camera mode, small map size of 10x10 and 20x20 was able to operate in maximum 60 framerates. From 30x30, the performance was starting to decrease linearly from 40 FPS until an average of 15 FPS at 50x50, total 2500 cells. Lastly, the framerate had been maintaining its average on 12 FPS until the biggest map size in our evaluation. From a reference standard to industrial game defacto, the average playable gameplay should have a framerate locked at a stable of 30 FPS with a minor circumstance when it had to reduce this lock into average of 15 FPS for a dense are of 3D assets. The reason that it provided a stable user-experience in gameplay while unlocked framerates would result in a jitter visualization as image may being ripped. While the visualization performance for Max-Zoom-Out Pan camera in our prototype performed poorly at the average of 12 FPS, it didn't mean that the actual crisis scenario simulation would suffer the unacceptable gameplay. In contrast, it indicated the requirement to have visual asset filtering process to ensure appropriate population of 3D assets

during a simulation. The prototype was playable in normal gameplay with maximum visual assets until the map size reach further 30 x 30 FPS. After the threshold, the necessary algorithm to create a level of detail for visualization could be applied but it was out of the scope of the thesis. Hence, this evaluation demonstrated that our framework provided acceptable gameplay performance with a map size range to 30 x30 with all visual assets available. The result can be improved by adding a level of details for 3D model assets with lower vertex-count or create a substitution process which applied cheaper meaningful representation to these 3D objects.

Next, the time requirement to create and initialize map was measured and displayed in Figure 7-2. The graph shown linear increasing in computation time requirement regarding to increasing in map size total cells with an average of 0.15 milliseconds per cell. Thus, it confirmed that our framework provides efficient stable map generation process in linear trend. The procedural map data structure generation was less than 2 seconds for 100 x 100, total of 10,000 cells.

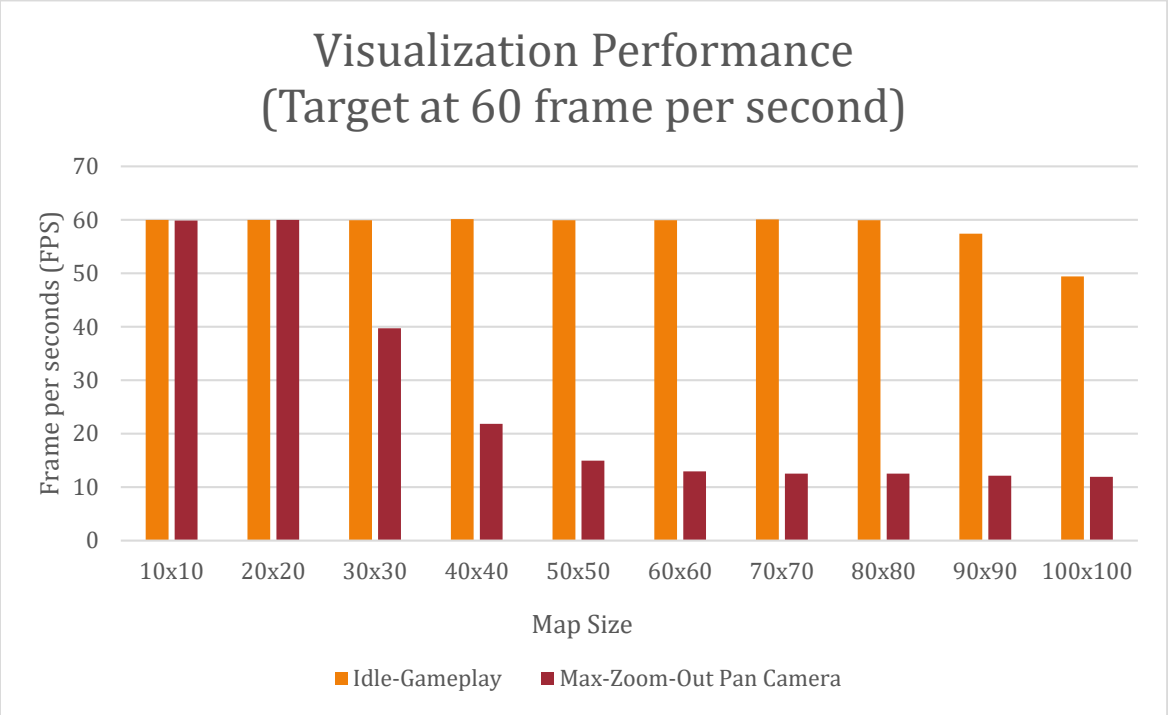


Figure 7-1 Visualization performance for standard gameplay and maximum-pan zoom

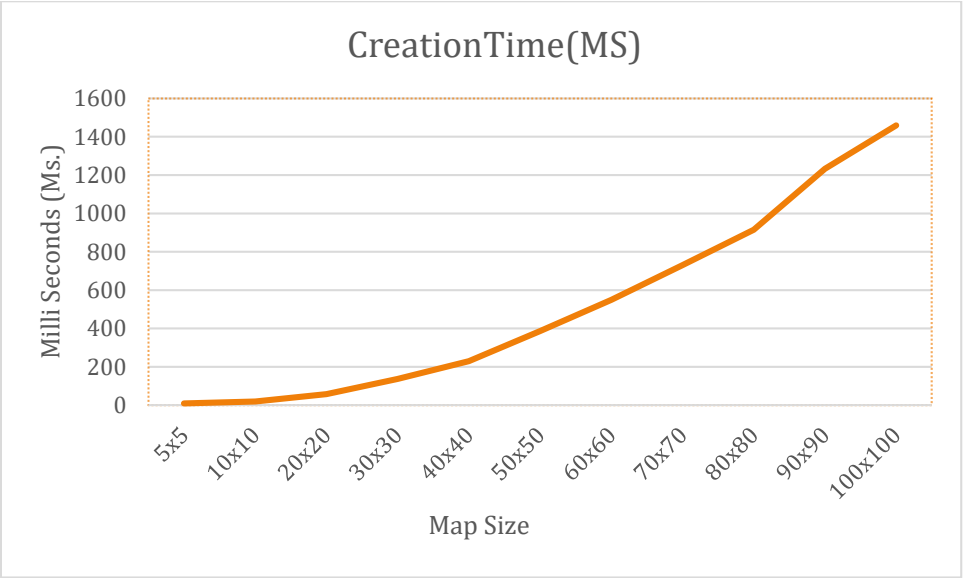


Figure 7-2 Map creation and initializing performance

Furthermore, to test on our built-in cellular automata algorithms on fire and flood disasters, we setup a test map with 30% fire and 30% flood situation areas focusing to evaluate time requirement of the process of disaster assessment and simulating its behaviour, which were heat radiation and water pressure equilibrium maintaining respectively. The result was presenting in Figure 7-3 highlighting these two processes regarding an increasing of total disaster areas. In small range of total disaster cells, the state evaluation and disaster behaviour simulation required similar computing time on average of less than 5 milliseconds until reaching total 400 cells. Then the distinct trend had been observed with an exponentially significant increase on state evaluation process in contrast to smaller linear changes regarding computation time of disaster behaviour simulation process. At total 1600 cells, the state evaluation stage required 11 times longer computation time than disaster behaviour simulation, which was finally at 1.5 seconds. Therefore, the performance of our simulation had strongly depended on complexity of the process in disaster state evaluation model as enabling or disabling parameters in the crisis evaluation allowed developer to have control over simulation performance. The propagation of disaster effect had a linear influence on system scalability and it took less than 20 milliseconds for 40x40 map size, total of 1600 cells.

Finally, Figure 7-4 displays the performance in responding time of single agent on different map sizes measured with an assumption of map setup with 50% forest areas and 10% of them are under fire situation. Agents in our experiment was equipped with global knowledge of every disaster location and its computation was abstractly separated into three stages: (a.) environment sensing, (b.) goal evaluation and selection, (c.) action selections. From the result, an increasing map size with proportionally increasing in total cell with crisis situation had influenced the exponentially increasing computation time of goal evaluation and selection stage closing to 800 milliseconds in 50x50 map while, in the opposite, sensing and action selection shared similarity in requiring less than an average of 14 milliseconds to complete. This contrast trends could be explained as the goal evaluation required logic that was more complex for assessing and sorting for the best possible goal candidate for agent. With an increasing of map size, this stage consumed more time whereas the sensing and action selection operates on only local agent environment perception and decision-making logic. The result suggested that we could improve the performance by setting up few agents having higher

environment perception scope with a task of assigning specific priority goal to lower scope agents. This approach, therefore, demonstrated a chain of command practice while it would reduce system computation cost. The example test scene for CA performance measurement and agent response time were shown in Figure 7-5 and Figure 7-6 respectively.

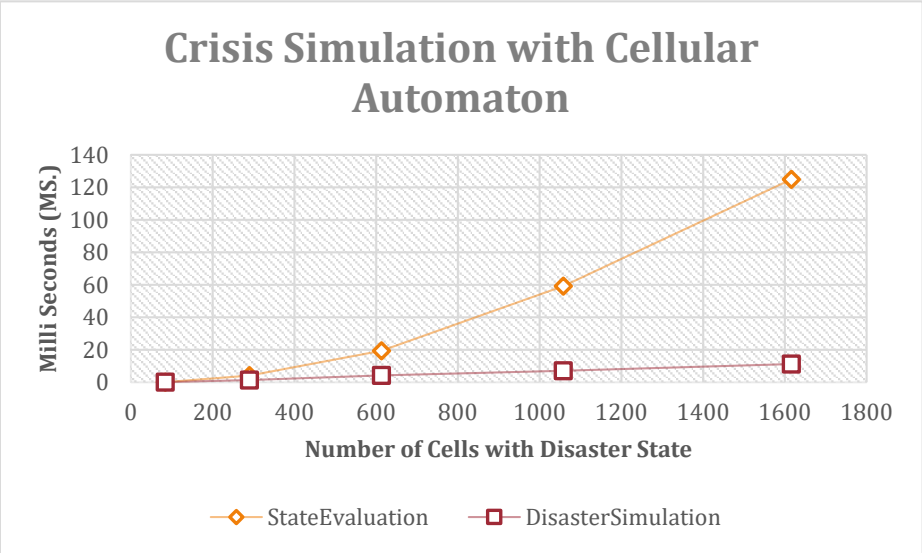
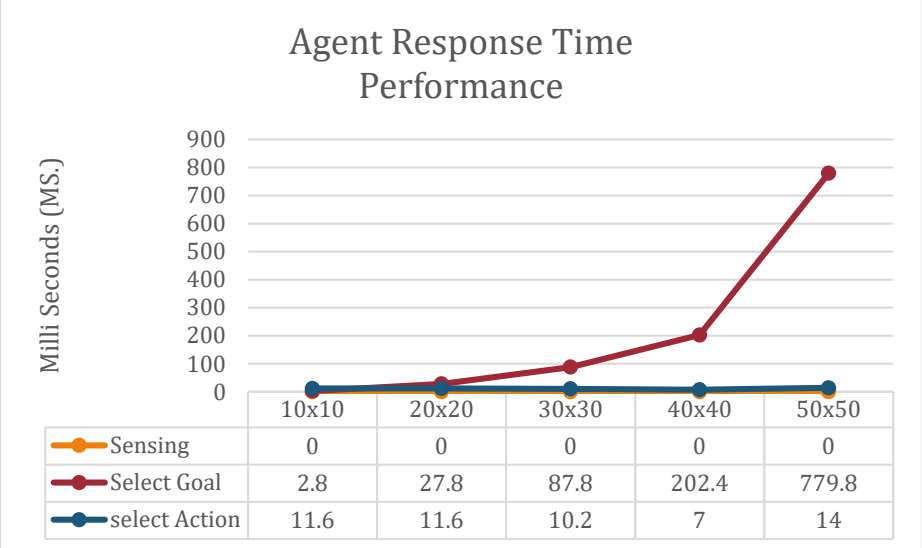


Figure 7-3 Crisis simulation performance with cellular automata model



	10x10	20x20	30x30	40x40	50x50
Sensing	0	0	0	0	0
Select Goal	2.8	27.8	87.8	202.4	779.8
select Action	11.6	11.6	10.2	7	14

Figure 7-4 Agent response-time performance

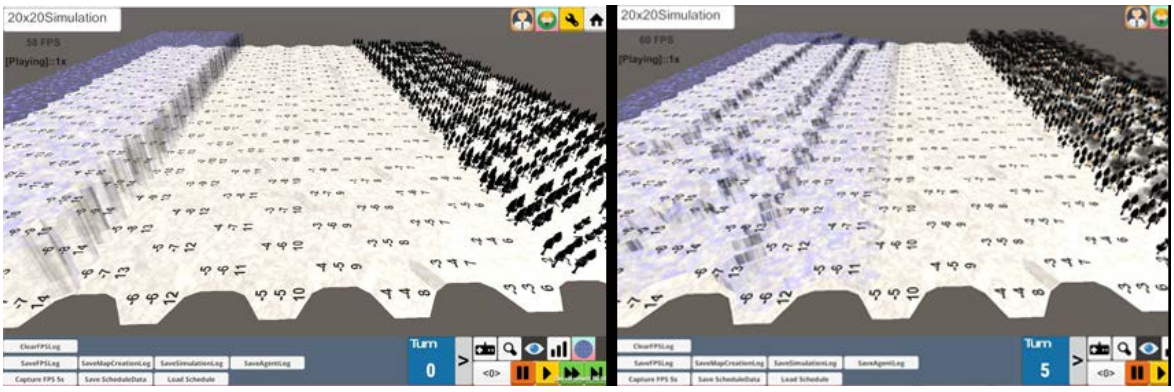


Figure 7-5 Example of test scene for CA model using fire and flood crisis.

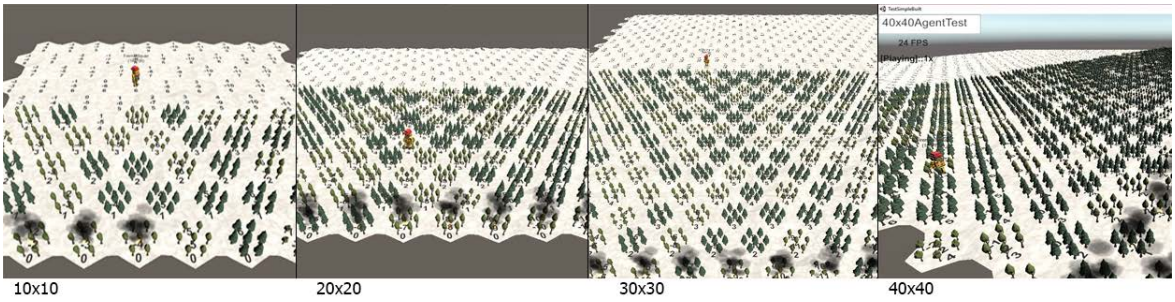


Figure 7-6 Example test scene for agent response time in an increasing map size

Figure 7-5 shows a simulation test for flooding and wild fire incidents in the example 20 x 20 map size, total 400 cells. The distribution of water mass generated a dispersion of water volume across a terrain from the left toward its middle zone while fire was spreading over the forest area in the similar manner using heat transfer model. Figure 7-6 shows an example of agent response time test in an increasing map size. Half of the terrain had been populated with forest parameters while 10 percent of them containing fire incident.

This subsection concluded the scalability testing for our framework with satisfying results. The framework provided sufficient stable framerate mostly at 60 fps for gameplay while the computation of map generation and disaster simulation process of every cell was also within the acceptable range (less than few seconds). Although the agent responding time was dramatically requiring more computation in larger map, it could be designed for only few highest scopes of stakeholders having this global awareness for strategy decision-making. Then reduced the necessity in majority of minor scope agents trading off for reduction in a computation time as mentioned in our evaluation.

7.2.2 Wildfire Crisis Scenario Example

In this subsection, we demonstrated the simulation of ongoing crisis in our framework. The example provided experiment with wildfire scenario that was happening on map size 50 x 50. The reason to select the 2500 total cells as a scenario test case was that it accommodated an average performance scale referenced from our previous test. For a landscape, dense forest was covering the northeast area while, in opposite direction, the south-west had a similar dense proportion of urban area. Both areas contained a small section having fire disasters. In regard of agents, we assumed appointment of firefighter squad with abundance resources for fire extinguishing action in the middle area as on the edge of both forest and city as roughly illustrated in Figure 7-7.

All agents were capable of global knowledge of disaster situation and could perform single movement with single extinguishing operation in each simulation turn. For the firefighter mission priority, we gave weight on the resolving of fire situation in city zone over the forest area. For any cell on fire situation each turn, it is resulting in 5% damage on the existing population and floral. Therefore, the evaluation in this given scenario was a measurement on total damage causing on the overall population and floral until the end of turn 25 regarding to the intervention from increasing number of firefighter agents. Figure 7-8 displays an increasing of the total devastating damage in our crisis scenario with non-existed agent reaching 75,000

damages maximum in turn 25. With intervention of every adding available agents, the damage was drastically reducing by roughly 28%, 48%, and 70% for 10 agents, 20 agents, and 30 agents respectively touching at approximate of 23,000 damages. However, there was only minor noticeable improvement for 40 agents, which contributes approximately 5% more reduction in total damage. The damage mitigation was perceptible starting from turn 10 that was due to the time-required for traveling into the disaster areas and thus more work force granted faster process on situation control. In contrast, the over-committed firefighter deployment from our experiment in 40 agents shown only slightly situation improvement because they were trying to perform operation on a crisis with limited accessibility to disaster cells resulting in many of the remaining staying idle wastefully. Figure 7-9 displays one of our test scene for wildfire scenario setup.

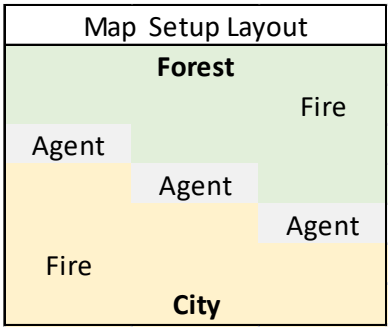


Figure 7-7 Map layout for wildfire example scenario with size 50x50

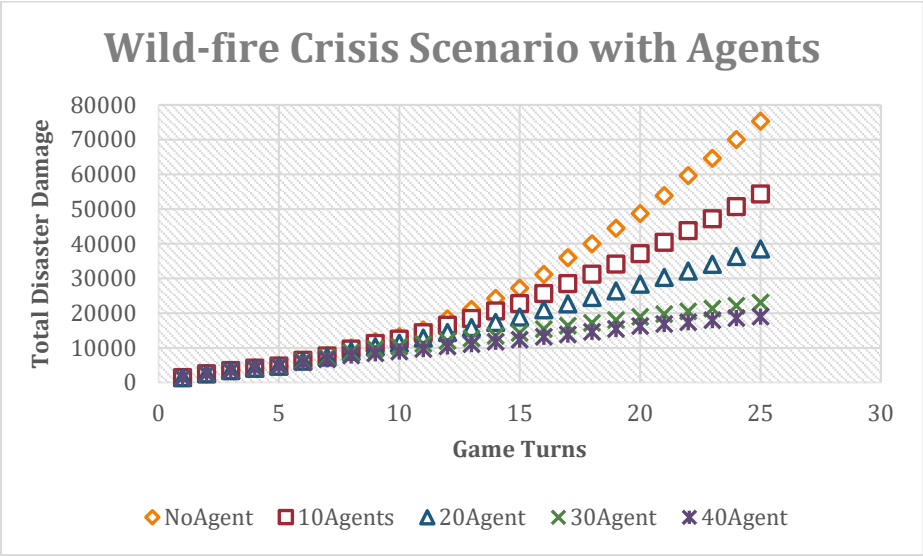


Figure 7-8 Wildfire scenario simulation with increasing firefighter agents during first 25 turns

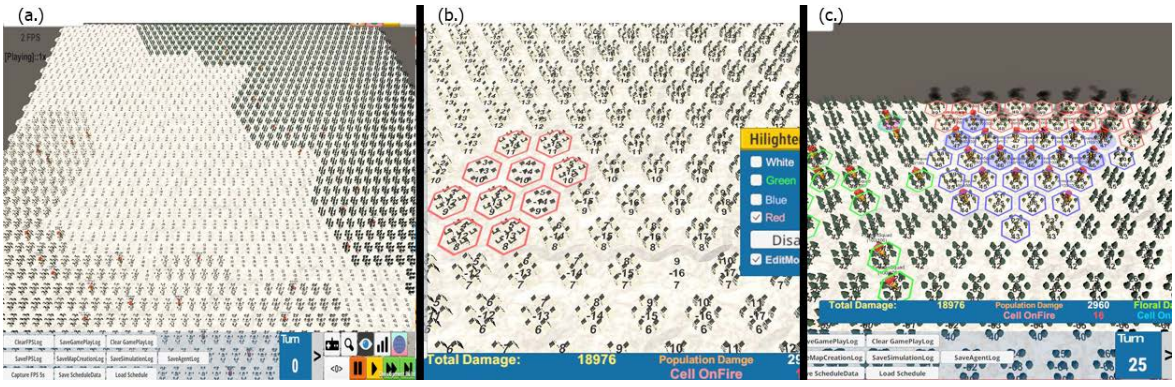


Figure 7-9 Example test scene size 50x50 cells for wildfire scenario simulation (a.) the map layout (b.) starting fire area for urban zone (c.) group of firefighters clear the forest fire area

In comparison to the existing literature, (Bailly & Adam, 2017) demonstrates an evaluation of the communication between crisis agents and population behaviours during bushfire situation in Australia. The study makes a decision to use indicators of number of deaths, number of injuries, total damage to the building, and total cost for agent communication for measuring a success or failure of the different strategies with a visual representation. While the experiment for different impact of each strategy is not in the scope of the thesis, the framework allows a test case scenario similar to the trend from state of the art that emphasis on related visual perception of situation and evaluation indicators that uses a common parameter for model of crisis simulation.

All in all, the example scenario had shown that our framework was suitable and can be used for testing a model of disaster situation including its resolution for resource deployment, strategy of damage control, and visualizing the result effectively.

7.2.3 Comparison of CGSA-SIM with the state of the art applications

We provide the comparison of CGSA-SIM framework with the other systems from their design and features since the performance measurement can become dependent to the running machine, thus, difficult to obtain run-time reference results from every system. The followings are the related systems or simulation model that reflects the aspect of computational crisis simulation whether for training, risk assessment, public awareness. SPRITE is a serious game built on agent-based model to simulate a risk management in costal flood subject on the Oleron Island (France) (Taillandier & Adam, 2018). FloodSim is a policy simulation serious game aiming to raise awareness of government policy regarding flood crisis (PlatGen, 2018). WRF-FIRE (S-Fire) is a fire simulation model implemented on FARSITE simulation engine for modelling fire behaviour with fuel concept, weather condition, and spatial topology (Dobrinkova, 2018). urbanExodus is a terminal client for a large-scale evacuation modelling system with the agent-based model with large population (Veeraswamy et al., 2018). The given example application is described using the scenario of Swinley forest fire covering the southeast region of England in 110 hectares. An urban emergency response plan (UERP) is proposed as a generic framework for assessing an emergency plan with provided agent-based modelling, spatial environment representation, and scenario authorization (Bandyopadhyay & Singh, 2018). UERP provides application test case from the framework as the assessment of infrastructure upgrade of overhead water tanks for fire emergency situation. Table 5 and Table 6 shows characteristics of each systems in compare with others including the proposed framework.

Although, each system provides rich features to create a hypothetical crisis simulation with agent-based modelling for both educational and assessment purposes, none of them integrates a pipeline for AI planning existed in CGSA-SIM framework. We distinguish an option to integrate AI planning with common agent component design modularly using web service architecture. In addition, we bridge the gap in allowing crisis designer and a general user share their workspace on the same running client. The development of real-time map editor enables both parties to take turn in propose response and introduce out-of-expected event effectively.

Table 5 Comparison between crisis simulation system and serious game for crisis training (a)

Comparison between crisis simulation system and serious game for crisis training				
	Gameplay / # of concurrent user	Landscape	Objectives	Target Players
CSGS-SIM	single/multiple user sharing same client to create scenario/resolve crisis	Any Imaginary piece of territory designed by the built-in map editor	Generating crisis scenario and study crisis scenario and resolution using ABS	Crisis Scenario Designers and crisis personal, and general public
SPRITE	single user	Oleron island (France)	Education on territorial risks management	Engineering students
FloodSim	single user	United Kingdom	Educational on flood situation to public	General population
WRF-Fire (Sfire)	single user	Any real landscape data in "landscape file" format	Computer simulation of wildfire propagation in Bulgaria	Researcher, crisis management personal
urbanEXODUS	single user	hypothetical data in geo spatial format such as shape files generated on web Interface	Simulate different evacuation procedures for crisis. The example application is given for a forest fire incident	Researcher, crisis management personal
urban emergency response plan (UERP)	single user	Any Imaginary piece of territory in GIS format	Assessment and identify deficiencies in an emergency plan	Related department in crisis management

Table 6 Comparison between crisis simulation system and serious game for crisis training (b)

Comparison between crisis simulation system and serious game for crisis training							
	Engine / Platform	Interface	Real-time gameplay/ result observation	Crisis models	custom scenario	custom agents	Agent's AI planning
CSGS-SIM	Unity3D Game Engine	3D terrains in hexagonal shape generated by PCG	yes	Cellular automata (CA) with implemented example models of wildfire and flood	yes	yes	yes

SPRITE	GAMA - an open source agent platform	2D game board	yes	Flood crisis CA	no	yes	no
FloodSim	Web Flash	2D spatial game board	yes	Flood crisis with the intervention of controlling on flood policy	no	no	no
WRF-Fire (Sfire)	FARSITE	2D spatial GIS geography	yes with supercomputers with 120 cores	Fire incident using fuel behaviour model	yes	no	no
urbanEXODUS	EXODUS Evacuation Simulation Engine	2D spatial GIS geography	no, offline computation in simulation engine for the generated result to be published by the user	Wildfire with Prometheus wildfire simulation (Based on CA technique)	yes	yes	no
urban emergency response plan (UERP)	GAMA	2D spatial GIS geography	yes	Open for development of any crisis model. The assessment demonstrates custom fire emergency response system (FER)	yes	yes	no

7.3 Run-time Results of a CGSA-SIM framework on Unity3D

In this subsection, the run-time application from the latest prototype of our framework had been presented. Figure 7-10 shows an overall screen visualization of the released application. The gameplay main menu allowed a control of turn step progression while each submenu could trigger a configuration panel for subsystems such as simulation controller and 3D assets visualization filter. The user controlled a scene navigation and orientation with keyword inputs and mouse. Both crisis scenario designer and practice personal shared the gameplay screen but different toolsets. The designer would access a map editor subsystem to define a scope of terrain, its parameters, crisis schedule, and a deployment of manpower with resources. The generated exported map file then could be loaded into the similar client for training personal to progress toward gameplay simulation while gave control over agent interaction. Otherwise, simply observation the development of crisis situation via automated agent decision-making was an option to test the hypothesis of crisis model.

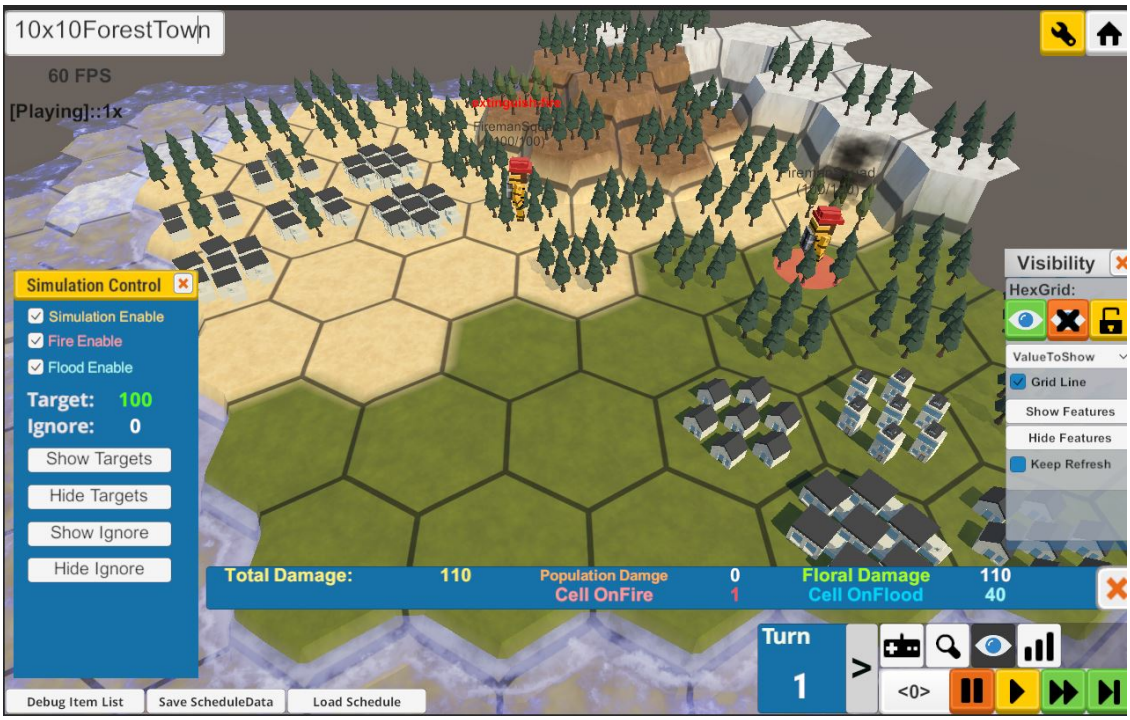


Figure 7-10 Framework application in Unity3D

Figure 7-11 represents a toolkit for map editing that can be used to modify a terrain at any game turn. Currently, the prototype accommodated eight types of parameters in each cell of terrain. The designer used mouse and parameter tool panel to define residing value of its type in the cells. The visualization process of the prototype then provided instant adjustment for terrain topology and placement of its 3D assets dynamically.

Figure 7-11 Map editor toolkit

Although the sculpting terrain gave a "*what you see is what you get*" aspect for terrain editor, the issue of time-consuming process for generating a suitable study case for crisis was not addressed. In our prototype, it provided a subsystem using procedural generation algorithm for terrain visualization for instantiating a draft map with descriptive modifiers for each residing parameters. Figure 7-12 presents an example of such subsystem for assignment of map parameter based on distribution factors during a map initialization process. The set of starting probability for each parameter could be defined beforehand then kept as given system models. This approach gave a head start for scenario designer to work immediately on their idea rather from scratch while still maintain a variation of generated product.

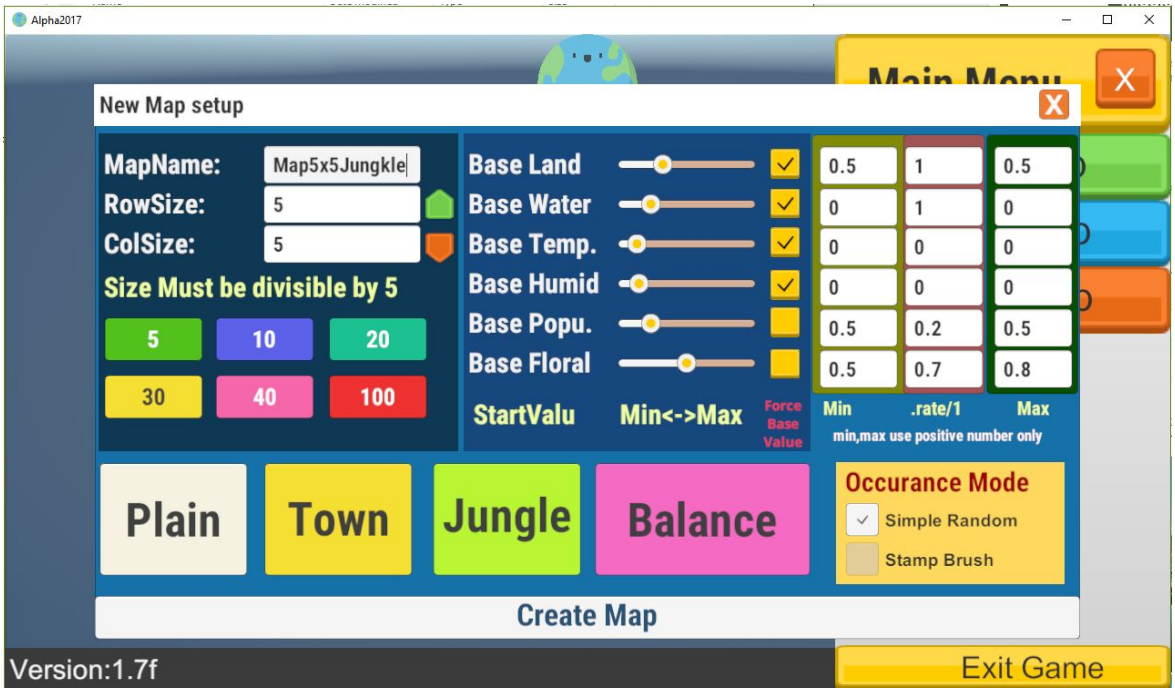


Figure 7-12 Probability distribution toolkit for Map initialization process

Figure 7-13 Fire and Flood disasters

Figure 7-13 visualizes a crisis state on cells for fire and flood disaster. The left image displayed a terrain with fire incident determined by a fire and smoke particle effect. An agent under direct control of practice personal was highlighted with the red colour. The right image shown an effect of flooding situation from exceeding

water mass than landmass parameter. We chose the fire and flood incidents as the result example since the visualization for these disasters were straightforward to their crisis nature.

Figure 7-14 shows how agent can be visualized on the map terrain with a corresponding 3D model to their role. The correct 3D assets representing agent had been assigned in association with predefined keywords during a compilation stage of prototype source code. The data to represent this keyword with 3D assets had been embedded in the exported map file, so using a “firefighter” as an agent role would obtain a result as fireman 3D model being displayed.

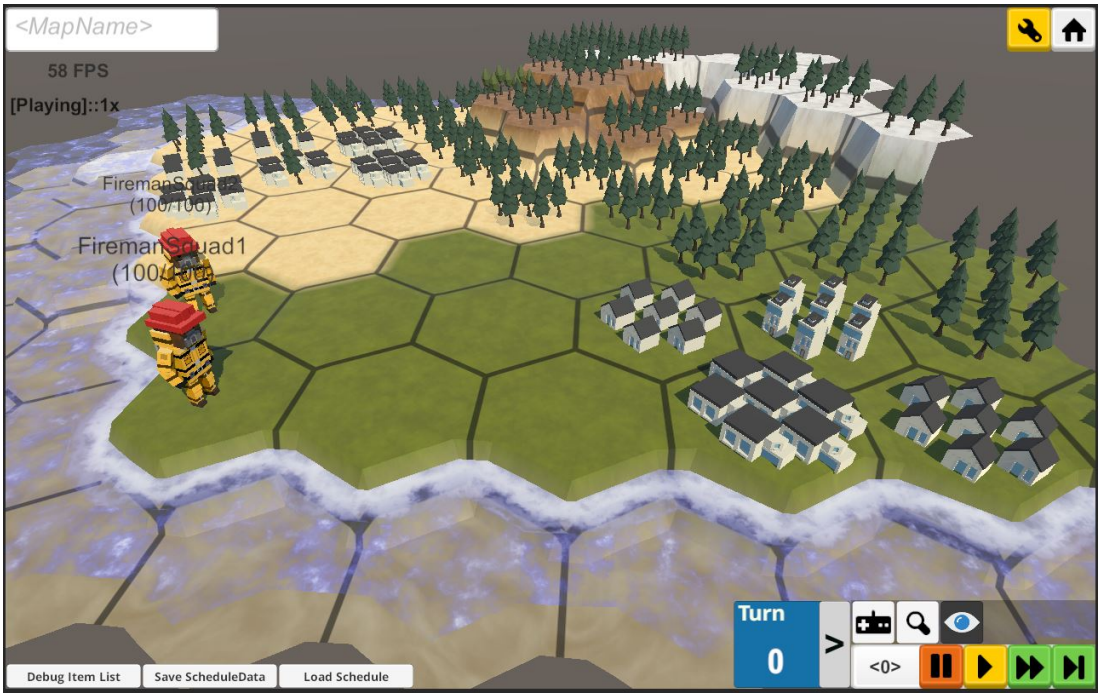


Figure 7-14 The agent as a firefighter on the map

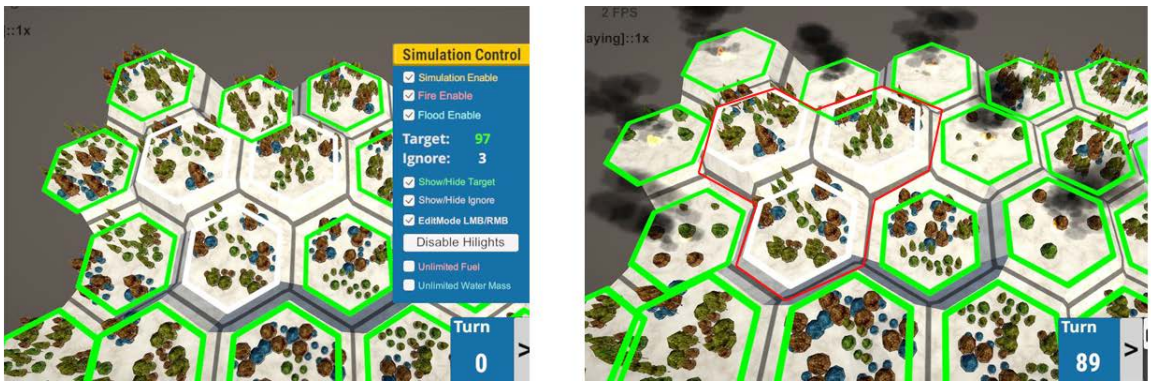


Figure 7-15 The example of crisis scenario designer marked an ignored list displayed in white border (left) and simulation manager simulate fire state transition CA except cells in ignore list (right)

Figure 7-15 demonstrates how a crisis designer can mark an ignore list for CA behaviour to influence a crisis model simulation direction. The simulation control subsystem provided a control panel for modification of crisis model. Importantly, it gave an option to include or exclude a specific area to be calculated in simulation process. This feature eased the generation of crisis disaster with a constrain direction such as a flooding without adding more parameters in flood crisis model itself.

Figure 7-16 Group of firefighter agent on the fire situation. The numbers are marked with a count of influenced agent units from the neighbour cells

Figure 7-16 shows how a group of agents trying to solve a fire incident and its implication on how fast each cell can be resolved. The closest on-fire cell surrounding by four fireman units was supposed to be handled first as it could be resolved more effectively than others. Moreover, the framework allowed a development of different strategy regarding to the incident resolution not just one shown in the example.

7.4 Summary

In this chapter, the run-time performance results of CGSA-SIM framework have been presented following with a test on performance scalability and a wildfire crisis scenario case study.

The sets of screen results show that we provide a modern game-based crisis simulation framework as a by-product of the thesis. It packs with a 3D visualization and real-time map editor with an option to procedurally generate a terrain from cell parameters distribution. The framework demonstrate how agent and crisis incident such as fire in the area can be displayed on the terrain grid with a game-like interpretation. Then the methodology to evaluate CGSA-SIM has been described by using a quantitative performance scalability test on visualization framerate, game initialization, simulation time, and agent decision-making time. The results on our visualization framerate and map initialization time produce an acceptable real-time quality of at least 30 fps in a middle-size map and an average of 0.15 seconds for the map initialization. Next, the evaluation on cellular automata simulation model shows that the process requires more time in evaluating a cell new state after a disaster model has been applied to the cell parameters. By limiting an area of simulation in the grid using our map editor tool, the computation time can be reduced. Lastly, the agent run-time has been tested using a simple reflex type IF-ELSE logics including a sensing environment, selecting a goal, and selecting an

action. The result shows that the agent performance has been influenced by the map size with the assumption that the agents have access to global map knowledge. Larger map size increases the goal evaluation stack in agent automated logic. The solution in this case is similar to a simulation procedure whereas only a few agents have an ability for global perception on the map and the rest contain only a local or limited sensing perception. All in all, the scalability test reports that our CGSA-SIM framework can operate in real-time for crisis simulation with agents.

Moreover, the test case scenario for a wildfire situation in the forest and urban area has been conducted. The running framework provides an observable collateral damage from fire incident upon civilians and environment while also showing how deployment of firefighter agents can influence the outcome of the ongoing emergency. By increasing the number of available firefighters in the simulation, the damages are visibly decreasing until a lowest unchangeable point since the traveling speed and accessibility to an impact area constant. Therefore, the test study shows that CGSA-SIM has demonstrated an ability to capture an emergency incident progression while allowing a study of resource and manpower management effectively.

In the last chapter, we conclude our thesis and highlight the contributions, novelties and future work.

8 CONCLUSION

Crisis management becomes an active area in current research area since crisis often involves a large coverage terrain and can potentially inflict a damage to environment, society, and the economy. By designing a crisis scenario as a hypothetical situation imitating an occurring emergency, study of its outcome and development of suitable personal training become possible.

By practice of using computer simulation in study of crisis scenario, the simulation framework provides a cost-effective solution compared to a complicated setup of real-world training session and this approach also allows impossible or dangerous scenario to be modelled safely. However, development of a crisis simulation software has a challenge since it requires to incorporate a variety of different components. Knowledge from an artificial intelligence is often required to create a human model as agent-based modelling (ABS) for imitating how civilian and crisis personal might behave during a disaster allowing a final outcome to be evaluated. Moreover, a creation of crisis scenario is relied on an area expert to deliver a content of narrative setup. This process is time-consuming and poses an issue to more demanding content in modern computer systems. Therefore, by incorporating a procedural content generation approach, the placeholder or even ready-to-use system content can be autonomously delivered by a suitable computer algorithm. Next, the issue is how to provide a generic framework to simulate a disaster model and its ever-changing impact on the area of environment, our study identified that cellular automata (CA) technique has a potential to become an efficient solution to model a crisis. The disasters often involve a large area of environment and its behaviour can be abstractly described as relationships between parameters and states of current area and its neighbours. Furthermore, on top of the crisis simulation, the agent model can have a AI planning technique being integrated to solve for a logical sequence of operation for each stakeholder. The resolution plan and disaster outcome generated from combined these components in a computer simulation system can be useful in crisis preparedness as a reference to the real-world emergency decision support system.

Fortunately, game technology has improved for a past decade and, thus, become an effective development tool to prototype a design of disaster simulation system. The pipeline of game engines in the modern market covers an essential component for computer applications in an entertainment such as graphics rendering, networking, cross-platform operational and deployment, input/output, and scripting language. By developing a crisis simulation system on-top of game technology as a serious game, the requirement to overcome a complexity of component design have been greatly reduced, thus fasten the prototyping to focus an objective of study crisis and its analytical study.

By study the literatures on crisis modelling and simulation systems, the results show that the common approach to model a disaster situation is to use cellular automata as a main or supplement logic to control how this crisis behave toward an environment. The automated content generation has been applied in both terrain construction and visualization. More importantly, a narrative sequence of event scenario can be

tailored using a computational approach by using a seed-based randomization, AI planning , and optimization generic algorithm. In addition, the agent-based modelling has been observed being applied as a main tool to simulate crisis in a large-scale setup of terrain coverage.

Although, there are several existing simulation systems aiming to provide an insight on crisis management and disaster outcome observation. The development of such a system on the game technology still lacks of a generic framework providing an integration of essential components: to model a different crisis event seamlessly; to adjust and generate a scenario content procedurally or in cooperation with human; to design and implement an agent model capable of intelligent in decision-making according to its role with AI planning capability; and, last but not least, to provide a real-time visual observation feedback as an immersive metaphor in study and practice of resource management during emergency.

To introduce a software development aspect in designing such a generic framework for crisis simulation and agent modelling on a game technology, this thesis discuss the background requirements and literatures on crisis simulation. Then the architecture for a game-based crisis simulation framework has been proposed based on the identified specifications focusing on three main aspects that are an environment representation; crisis modelling template and scenario simulation; and agents modelling with internal decision-making behaviour toolkits. As a result, we present the software design schematic in UML for the proposed architecture developed on top of Unity3D game engine consisting of its eight main components, which are map manager, simulation manager, crisis event scheduler, agents, planning manager, game manager, user interface, and data manager. The sub processes of each component have been explained in details. The CGSA-SIM (Crisis Game for Scenario design and Agent modelling Simulation) framework focus on an objective to be a generic solution to prototyping a design of crisis model and its scenario. Then allows observation its outcome via built-in simulation logic based on cellular automata to obtain a potential analytical study using a game technology.

The implementation of map representation focuses on using a procedural content generation (PCG) technique to ease a visualization structure of hexagonal grid as our main terrain representation. This method provides a benefit of using control parameters designed as an area abstract characteristic to be calculated in a visual adjustment of hexagonal landscape reflecting a dynamic environment in the simulation. It allows a setup of hypothetical environment procedurally using parameter preferences to produce a draft landscape for domain experts to use an integrated map editor for completing a final detail manually. With a map data structure designed in hexagonal 2D grid with cell having this parameters, the cellular automata logic has been implemented to facilitate a modelling of crisis phenomenon. The disaster situation inflicts a change in state of cell parameters which is also observable in the visual representation of terrain. Our study demonstrate a feasibility of using CA into model a wildfire incident as an example running on Processing2D. It shows the process of applying CA algorithm for more sophisticated wildfire situation and flooding crisis models. Moreover, the implementation of crisis event scheduler provides a concrete pipeline to design a sequence of narrative crisis situation event that is reusable in different map configuration. These narrative events can be exported for other system using JSON format parser from the data manager. Lastly, the agent representation for different stakeholders has been addressed. We provide a template to script a decision-making behavior of agents with an additional capability of AI planning. The planning pipeline in the CGSA-SIM is based on communicating with an external planner using web service to provide a flexibility in domain design,

configuration, and observable generated plan. This approach indirectly separates a computation requirement from the client-side with exponential time complexity for planning.

Finally, the evaluation of CGSA-SIM has been presented quantitatively and qualitatively. The performance scalability have been tested using a visualization framerate, map initialization time, a crisis state evaluation, and agent decision-making time. The results infer that CGSA-SIM can achieve an acceptable real-time framerate for a visualization of the environment and its simulation up to a game-based application standard of at least 30fps. The time complexity for crisis state evaluation and agent decision-making are strongly depended on the map size in particular to a total number of cell these procedures require to process. We have discussed that the state evaluation complexity can be alleviated by limiting an area of simulation to a local group of cells. An agent decision-making solution should applied a similar approach by only a few agents are equipped of large areas' knowledge then limit simple agents with only a smaller-range perception.

Furthermore, the test case scenario has been conducted to verify the usability of the proposed framework. The wildfire situation on the medium-size map of area consisting a half portion being an urban and forest provides an insight on how damage of uncontrolled disaster can be evaluated comparing to an intervention of firefighter agent's deployment. The increasing number of agents proves a reduction of total damage inflicted by the incident until a constant unchangeable threshold. The CGSA-SIM can capture a nature of wildfire crisis phenomenon reasonably thus it verifies that the framework is capable of being a tool to study a crisis situation effectively.

All in all, the provided implementation result on representation of the core components and testing in visualization stage from the framework shows that CGSA-SIM is a flexible tool to support modelling, visualization, and testing of general crisis scenario situation in real-time. We have contributed a method of using PCG to designing a 3D environment for the crisis management system with an influence from control parameters of cellular automata model. In consequence, the scenario designers have an ability to use an interactive map editor to control both environment generation and scenario setup seamlessly using our design approach. With the provided event scheduler for a crisis management system, it eases the design of reusable narrative crisis scenario. By providing an agent framework in the game-based system, a new multi-agent model for crisis management has been developed efficiently using our approach of hybrid combination of conditional agent and planning algorithms using a web service with both FF and FF-Metric planners. The simulation of different crisis models can be stacked on top of each other in the same game time-step to combine a related situation with each another in our provided CA system. The majority of 3D assets in the framework are flexible to be replaced with a necessary level of detail suitable for a final application based on a user preference. As a result, the development of our CGSA-SIM lies on the incorporating of these features to become a novel generic framework potentially able to model and simulate an outcome of crisis management. In addition, the portability from our implementation on Unity3D game engine allows final game-based application to be built directly for different platforms including web-based and mobile devices. These aspects provide a benefit of accessibility to crisis scenario manager, designer, crisis personals, and interested individuals.

From our contributions, the current crisis decision-support system and training domain can have a benefit from assisting simulation framework that exploits modern game engine for visualization and development of crisis model. Since design and test crisis scenario are time-consuming, general crisis-simulation framework is very valuable for rapid prototyping of new open crisis theory. It also mitigates the

platform dependent issue and allows ease of extension for new visualization and interface technologies which often being integrated into game engine. The framework provides an adequate scalability performance, and it can be effectively applied for any crisis scenario simulation based on cellular automata technique. Therefore, our CGSA-SIM can become one of a tool that allows crisis personals to better interact, understand, develop model, and predict possible outcome of crisis incidents.

However, the work is preliminary and several aspect of CGSA-SIM component still lack a potential feature. First, the generation of a terrain did bases on the hypothetical preference and human-expert while the real-world data can be integrated to guide a PCG algorithm to realize more accurate mapping of geographical landscape and cell parameters. Next, the simulation logic bases on cellular automata has not include a physics model and still faces scalability issue on the large map without restriction. Furthermore, the scenario cases are conducted on an assumption of only a simplified agent-behaviour model, and whilst the AI planning system and its architecture has been developed, actual integration can still become an open challenge of how effective combined agent behaviour decision-making model can perform. More analytical test on developing a complex multi-agent structure using CGSA-SIM can give a potential benefit to future improvement of framework design.

Future work

This section provides a direction for a prospective extension on the current work in the thesis. Many features integration, tests, and experiments have been considered out of the thesis scope. The future work concerns several ways to improve the usability and performance of the system.

First, the map representation can be integrated with geographic information system to generate an accurate detail of environment parameters on each cell in the grid. The extracted building placement and a road network can be mapped into a different data layers to facilitate an evacuation response. Another area of improvement is for visualization of the system with a state of the art in virtual reality and augmented reality technologies. It would create an immersive aspect to both trainee and scenario designer. Fortunately, both technologies are being supported in Unity3D game engine.

Next, the simulation of crisis scenario can have a benefit from a distributed computation architecture such as distributed computing platform. While the simulation model in our work is limited to the turn time-step with simplified rules, combining several crisis models at the same time will increase the computational complexity greatly. With an extension to synchronize a world data into networked servers for a hazard simulation, the client can reduce its CPU/GPU workload resulting in a better user-experience framerate.

Likewise, the framework can have an extension for a multiplayer architecture to link a design, simulation, and visualization process with different stakeholders at the same time. By sharing an option to interact with a virtual ongoing simulation, every group of participants can have a clear understanding of their role, constraint, and outcome of the action.

Lastly, the multi-agent architecture in our framework can be improved by providing more template on implementing a standard decision-making behaviour such as finite-state machine, decision tree such as ID3 algorithm, fuzzy logic, auction-net protocol, and rule-based reasoning using external scripting such as Lua and Python. The planning algorithm could be internally implemented as an optional built-in Unity3D-based script

since we provided a class structure for PDDL predicate representation and parsing. In addition to the planner, constraints system and replanning would be developed to cope with real-world uncertainties and adapting resources and agent's actions to very dynamic crisis scenario. Eventually, the web service accessibility can feature a portal connecting to CGSA-SIM application in obtaining a simulation report, editing and designing agent template, and lastly extracted a useful information to be integrated with real decision support system for crisis management.

REFERENCE

Aeronautiques, C., Howe, A., Knoblock, C., McDermott, I. D., Ram, A., Veloso, M., . . . Christianson, D. (1998). PDDL| The Planning Domain Definition Language.

Allen, M., Prusinkiewicz, P., & DeJong, T. (2005). Using L-systems for modeling source–sink interactions, architecture and physiology of growing trees: The L-PEACH model. *New phytologist, 166*(3), 869-880.

Amal Dar Aziz, J. C., Raylene Yung. (2004). Basics of Automata Theory. Retrieved from https://cs.stanford.edu/people/eroberts/courses/soco/projects/2004-05/automata-theory/basics.html

Bacchus, F., & Kabanza, F. (2000). Using temporal logics to express search control knowledge for planning. *Artificial intelligence, 116*(1-2), 123-191.

Bailly, C., & Adam, C. (2017). An interactive simulation for testing communication strategies in bushfires. *ISCRAM, Albi*.

Bandyopadhyay, M., & Singh, V. (2018). Agent-based geosimulation for assessment of urban emergency response plans. *Arabian Journal of Geosciences, 11*(8), 165. doi:10.1007/s12517-018-3523-5

Barkun, M. (1977). Disaster in history. *Mass Emergencies, 2*(3), 219-231.

Barrett, C. L., Eubank, S. G., & Smith, J. P. (2005). If smallpox strikes Portland. *Scientific American, 292*(3), 54-61.

Barros, L. M., & Musse, S. R. (2005). *Introducing narrative principles into planning-based interactive storytelling.* Paper presented at the Proceedings of the 2005 ACM SIGCHI International Conference on Advances in computer entertainment technology.

Barros, L. M., & Musse, S. R. (2007). Planning algorithms for interactive storytelling. *Computers in Entertainment (CIE), 5*(1), 4.

Bays, C. (2010). Introduction to Cellular Automata and Conway's Game of Life. In: Springer.

Bekins, D., & Aliaga, D. G. (2005). *Build-by-number: Rearranging the real world to visualize novel architectural spaces.* Paper presented at the Visualization, 2005. VIS 05. IEEE.

Bellifemine, F. L., Caire, G., & Greenwood, D. (2007). *Developing multi-agent systems with JADE* (Vol. 7): John Wiley & Sons.

Benjamin, S. C., Johnson, N. F., & Hui, P. (1996). Cellular automata models of traffic flow along a highway containing a junction. *Journal of Physics A: Mathematical and General, 29*(12), 3119.

Blum, A. L., & Furst, M. L. (1997). Fast planning through planning graph analysis. *Artificial intelligence, 90*(1-2), 281-300.

Boucher, A., Canal, R., Chu, T.-Q., Drogoul, A., Gaudou, B., Moraru, V., . . . Sempé, F. (2009). *The AROUND project: Adapting robotic disaster response to developing countries.* Paper presented at the Safety, Security & Rescue Robotics (SSRR), 2009 IEEE International Workshop on.

Chad, A. (2018). Evolving Cellular Automata Rules for Maze Generation.

Challenger, R., Clegg, C., Robinson, M., & Leigh, M. (2009). Understanding crowd behaviours: simulation tools. *UK Cabinet Office*.

Christopoulou, E., & Xinogalos, S. (2017). Overview and Comparative Analysis of Game Engines for Desktop and Mobile Devices. *INTERNATIONAL JOURNAL OF SERIOUS GAMES, 4*(4), 21-36.

Ciesla, R. (2017). Mostly codeless game development. *Berkeley, CA: Apress*.

Cowan, B., & Kapralos, B. (2014). *A survey of frameworks and game engines for serious game development.* Paper presented at the Advanced Learning Technologies (ICALT), 2014 IEEE 14th International Conference on.

Crisci, G. M., Rongo, R., Di Gregorio, S., & Spataro, W. (2004). The simulation model SCIARA: the 1991 and 2001 lava flows at Mount Etna. *Journal of Volcanology and Geothermal Research, 132*(2), 253-267.

Dijkstra, E. W. (1959). A note on two problems in connexion with graphs. *Numerische mathematik, 1*(1), 269-271.

Dobrinkova, N. (2018). Wildfire Optimizations in Modeling and Calibrations for Bulgarian Test Cases. In S. Fidanova (Ed.), *Recent Advances in Computational Optimization: Results of the Workshop on Computational Optimization WCO 2016* (pp. 25-40). Cham: Springer International Publishing.

Dorigo, M., Maniezzo, V., & Colorni, A. (1996). Ant system: optimization by a colony of cooperating agents. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics), 26*(1), 29-41.

Duarte, R., Goudoulakis, E., El Rhalibi, A., & Merabti, M. (2013). *A conversational avatar framework for digital interactive storytelling.* Paper presented at the Presented at the the 14th Annual Post Graduate Symposium on the Convergence of Telecommunications, Networking and Broadcasting (PGNet2013), Liverpool.

Duncan, S. C. (2011). Minecraft, beyond construction and survival. *Well Played: a journal on video games, value and meaning, 1*(1), 1-22.

Eilenberg, S. (1974). *Automata, languages, and machines*: Academic press.

El Rhalibi, A., Goudoulakis, E., & Merabti, M. (2012). *DIS planning algorithms evaluation.* Paper presented at the Consumer Communications and Networking Conference (CCNC), 2012 IEEE.

Fikes, R. E., & Nilsson, N. J. (1971). STRIPS: A new approach to the application of theorem proving to problem solving. *Artificial intelligence, 2*(3-4), 189-208.

Gad-el-Hak, M. (2009). *The art and science of large-scale disasters* (Vol. 57).

Galán, J. M., Izquierdo, Luis R., Izquierdo, Segismundo S., Santos, José Ignacio, del Olmo, Ricardo, López-Paredes, Adolfo and Edmonds, Bruce. (2009). Errors and Artefacts in Agent-Based Modelling. *Journal of Artificial Societies and Social Simulation, 12*(1), 1.

Geffner, P. H. H., & Haslum, P. (2000). *Admissible heuristics for optimal planning.* Paper presented at the Proceedings of the 5th Internat. Conf. of AI Planning Systems (AIPS 2000).

Gerevini, A., & Serina, I. (2002). *LPG: A Planner Based on Local Search for Planning Graphs with Action Costs.* Paper presented at the AIPS.

Gershenson, C., & Rosenblueth, D. (2009). Modeling self-organizing traffic lights with elementary cellular automata. Submitted. In.

Ghallab, M., Nau, D., & Traverso, P. (2004). *Automated Planning: theory and practice*: Elsevier.

Greuter, S., Parker, J., Stewart, N., & Leach, G. (2003). *Real-time procedural generation ofpseudo infinite'cities.* Paper presented at the Proceedings of the 1st international conference on Computer graphics and interactive techniques in Australasia and South East Asia.

Grois, E., Hsu, W. H., Voloshin, M., & Wilkins, D. C. (1998). *Bayesian network models for generation of crisis management training scenarios.* Paper presented at the AAAI/IAAI.

Hart, P. E., Nilsson, N. J., & Raphael, B. (1968). A Formal Basis for the Heuristic Determination of Minimum Cost Paths. *IEEE Transactions on Systems Science and Cybernetics, 4*(2), 100-107. doi:10.1109/TSSC.1968.300136

Hawe, G. I., Coates, G., Wilson, D. T., & Crouch, R. S. (2012). Agent-based simulation for large-scale emergency response: A survey of usage and implementation. *ACM Computing Surveys (CSUR), 45*(1), 8.

Helmert, M. (2006). The fast downward planning system. *J. Artif. Int. Res., 26*(1), 191-246.

Hendrikx, M., Meijer, S., Van Der Velden, J., & Iosup, A. (2013). Procedural content generation for games: A survey. *ACM Transactions on Multimedia Computing, Communications, and Applications (TOMM), 9*(1), 1.

Hoffmann, J. (2001). FF: The fast-forward planning system. *AI magazine, 22*(3), 57.

Hoffmann, J. (2003). The Metric-FF Planning System: Translating``Ignoring Delete Lists"to Numeric State Variables. *Journal of Artificial Intelligence Research, 20*, 291-341.

Holcomb, S. D., Porter, W. K., Ault, S. V., Mao, G., & Wang, J. (2018). *Overview on DeepMind and Its AlphaGo Zero AI*. Paper presented at the Proceedings of the 2018 International Conference on Big Data and Education, Honolulu, HI, USA.

Hooshyar, D., Yousefi, M., & Lim, H. (2018). A Procedural Content Generation-Based Framework for Educational Games: Toward a Tailored Data-Driven Game for Developing Early English Reading Skills. *Journal of Educational Computing Research, 56*(2), 293-310. doi:10.1177/0735633117706909

Hopcroft, J. E., Motwani, R., & Ullman, J. D. (2001). Introduction to automata theory, languages, and computation. *Acm Sigact News, 32*(1), 60-65.

Hullett, K., & Mateas, M. (2009). *Scenario generation for emergency rescue training games*. Paper presented at the Proceedings of the 4th International Conference on Foundations of Digital Games.

Hussain, F., Gurung, A., & Jones, G. (2014). *Cocos2d-x game development essentials*: Packt Publishing Ltd.

Ilghami, O. (2006). Documentation for JSHOP2. *Department of Computer Science, University of Maryland, Tech. Rep*.

Izquierdo, L. R., & Polhill, J. G. (2006). Is Your Model Susceptible to Floating-Point Errors? *Journal of Artificial Societies and Social Simulation, 9*(4), 4.

Jain, S., & McLean, C. (2003). *Simulation for emergency response: a framework for modeling and simulation for emergency response.* Paper presented at the Proceedings of the 35th conference on Winter simulation: driving innovation.

Johnson, L., Yannakakis, G. N., & Togelius, J. (2010). *Cellular automata for real-time generation of infinite cave levels.* Paper presented at the Proceedings of the 2010 Workshop on Procedural Content Generation in Games.

Karafyllidis, I., & Thanailakis, A. (1997). A model for predicting forest fire spreading using cellular automata. *Ecological Modelling, 99*(1), 87-97.

Kari, J. (2005). Theory of cellular automata: A survey. *Theoretical Computer Science, 334*(1), 3-33. doi:https://doi.org/10.1016/j.tcs.2004.11.021

Kautz, H., & Selman, B. (1992). *Planning as satisfiability*. Paper presented at the Proceedings of the 10th European conference on Artificial intelligence, Vienna, Austria.

Kermack, W. O., & McKendrick, A. G. (1991). Contributions to the mathematical theory of epidemics—I. *Bulletin of Mathematical Biology, 53*(1), 33-55. doi:10.1007/bf02464423

Khalil, K. M., Abdel-Aziz, M., Nazmy, T. T., & Salem, A.-B. M. (2009). Bridging the gap between crisis response operations and systems. *arXiv preprint arXiv:0908.4290*.

Kim, J.-C., Jung, H., Kim, S., & Chung, K. (2016). Slope Based Intelligent 3D Disaster Simulation Using Physics Engine. *Wireless Personal Communications, 86*(1), 183-199. doi:10.1007/s11277-015-2788-1

Kitano, H., Tadokoro, S., Noda, I., Matsubara, H., Takahashi, T., Shinjou, A., & Shimada, S. (1999). *Robocup rescue: Search and rescue in large-scale disasters as a domain for autonomous agents research.* Paper presented at the Systems, Man, and Cybernetics, 1999. IEEE SMC'99 Conference Proceedings. 1999 IEEE International Conference on.

Koehler, J., Nebel, B., Hoffmann, J., & Dimopoulos, Y. (1997). *Extending planning graphs to an ADL subset.* Paper presented at the European Conference on Planning.

Kose, H., Tatlidede, U., Meriçli, C., Kaplan, K., & Akin, H. L. (2004). *Q-learning based market-driven multi-agent collaboration in robot soccer.* Paper presented at the Proceedings of the Turkish symposium on artificial intelligence and neural networks.

Koto, T., & Takeuchi, I. (2003). *A distributed disaster simulation system that integrates sub-simulators.* Paper presented at the First International Workshop on Synthetic Simulation and Robotics to Mitigate Earthquake Disaster.

Kubera, Y., Mathieu, P., & Picault, B. (2010). *Everything can be agent!* Paper presented at the Proceedings of the 9th International Conference on Autonomous Agents and Multiagent Systems: volume 1 - Volume 1, Toronto, Canada. https://dl.acm.org/citation.cfm?id=1838474

Lechner, T., Watson, B., Ren, P., Wilensky, U., Tisue, S., & Felsen, M. (2004). Procedural modeling of land use in cities.

Li, W. (1987). Power spectra of regular languages and cellular automata. *Complex Systems, 1*(1), 107-130.

Li, X., & Song, Q. (2018). Wargaming-Based Crisis Drills.

Lipman-Blumen, J. (1975). A Crisis Framework Applied to Macrosociological Family Changes: Marriage, Divorce, and Occupational Trends Associated with World War II. *Journal of Marriage and Family, 37*(4), 889-902. doi:10.2307/350840

Mao, J., Wang, S., Ni, J., Xi, C., & Wang, J. (2017). Management System for Dam-Break Hazard Mapping in a Complex Basin Environment. *ISPRS International Journal of Geo-Information, 6*(6), 162.

Martin, G., Schatz, S., Bowers, C., Hughes, C. E., Fowlkes, J., & Nicholson, D. (2009). *Automatic scenario generation through procedural modeling for scenario-based training.* Paper presented at the Proceedings of the Human Factors and Ergonomics Society Annual Meeting.

Maurina, E. F. (2006). *The game programmer's guide to Torque: under the hood of the Torque Game Engine*: CRC Press.

McIntosh, H. V. (1990). Wolfram's class IV automata and a good life. *Physica D: Nonlinear Phenomena, 45*(1-3), 105-121.

Metello, M. G., Casanova, M. A., & de Carvalho, M. T. M. (2008). *Using Serious Game Techniques to Simulate Emergency Situations.* Paper presented at the GeoInfo.

Mniszewski, S. M., Del Valle, S. Y., Stroud, P. D., Riese, J. M., & Sydoriak, S. J. (2008). *EpiSimS simulation of a multi-component strategy for pandemic influenza.* Paper presented at the Proceedings of the 2008 Spring simulation multiconference.

Moore, E. F. (1959). *The shortest path through a maze.* Paper presented at the Proc. Int. Symp. Switching Theory, 1959.

Moore, E. F. (1962). *Machine models of self-reproduction.* Paper presented at the Proceedings of symposia in applied mathematics.

Müller, P., Wonka, P., Haegler, S., Ulmer, A., & Van Gool, L. (2006). *Procedural modeling of buildings.* Paper presented at the ACM Transactions on Graphics (TOG).

Müller, P., Zeng, G., Wonka, P., & Van Gool, L. (2007). Image-based procedural modeling of facades. *ACM Transactions on Graphics (TOG), 26*(3), 85.

Mysore, V., Narzisi, G., & Mishra, B. (2006). *Agent modeling of a sarin attack in manhattan.* Paper presented at the Proceedings of the First International Workshop on Agent Technology for Disaster Management, ATDM.

Nagel, K., & Schreckenberg, M. (1992). A cellular automaton model for freeway traffic. *Journal de physique I, 2*(12), 2221-2229.

Narzisi, G., Mysore, V., & Mishra, B. (2006). Multi-objective evolutionary optimization of agent-based models: An application to emergency response planning. *Computational Intelligence, 2006*, 224-230.

Nau, D., Cao, Y., Lotem, A., & Munoz-Avila, H. (1999). *SHOP: Simple hierarchical ordered planner.* Paper presented at the Proceedings of the 16th international joint conference on Artificial intelligence-Volume 2.

Nilson, B., & Söderberg, M. (2007). Game Engine Architecture. *Mälardalen University*.

North, M. J., & Macal, C. M. (2007). *Managing business complexity: discovering strategic solutions with agent-based modeling and simulation*: Oxford University Press.

Padia, K., Bandara, K. H., & Healey, C. G. (2018). *Yarn: Generating Storyline Visualizations Using HTN Planning.* Paper presented at the Graphics Interface.

Parish, Y. I., & Müller, P. (2001). *Procedural modeling of cities.* Paper presented at the Proceedings of the 28th annual conference on Computer graphics and interactive techniques.

Pattrasitidecha, A. (2014). Comparison and evaluation of 3D mobile game engines.

Perlin, K. (1985). An image synthesizer. *ACM Siggraph Computer Graphics, 19*(3), 287-296.

Perry, R. W. (2007). What Is a Disaster? In *Handbook of Disaster Research* (pp. 1-15). New York, NY: Springer New York.

Petridis, P., Dunwell, I., Panzoli, D., Arnab, S., Protopsaltis, A., Hendrix, M., & de Freitas, S. (2012). Game engines selection framework for high-fidelity serious applications. *International Journal of Interactive Worlds, 2012*, 1.

PlatGen. (2018). FloodSim.

Porteous, J., Cavazza, M., & Charles, F. (2010). Applying planning to interactive storytelling: Narrative control using state constraints. *ACM Transactions on Intelligent Systems and Technology (TIST), 1*(2), 10.

Prusinkiewicz, P. (1986). *Graphical applications of L-systems.* Paper presented at the Proceedings of graphics interface.

Prusinkiewicz, P., & Lindenmayer, A. (1990). Modeling of cellular layers. In *The Algorithmic Beauty of Plants* (pp. 145-174): Springer.

Quade, E. S., & Carter, G. M. (1989). *Analysis for public decisions*: MIT Press.

Reese, R., & Johnson, J. (2015). *jMonkeyEngine 3.0 game development: A practical guide*: P8tech.

Rocha, R. V., Rocha, R. V., & Araújo, R. B. (2010). *Selecting the best open source 3D games engines.* Paper presented at the Proceedings of the brazilian Symposium on Games and Digital Entertainment, Florianópolis, Santa Catarina, brazil.

Roden, T., & Parberry, I. (2004). From artistry to automation: A structured methodology for procedural content creation. *Entertainment Computing–ICEC 2004*, 301-304.

Rossier, J., Petraglio, E., Stauffer, A., & Tempesti, G. (2004). *Tom Thumb Algorithm and von Neumann universal constructor.* Paper presented at the International Conference on Cellular Automata.

Rozenberg, G., & Salomaa, A. (1980). *The mathematical theory of L systems* (Vol. 90): Academic press.

Russell, S., Norvig, P., & Intelligence, A. (1995). A modern approach. *Artificial Intelligence. Prentice-Hall, Egnlewood Cliffs, 25*, 27.

Russell, S. J., Norvig, P., Canny, J. F., Malik, J. M., & Edwards, D. D. (2003). *Artificial intelligence: a modern approach* (Vol. 2): Prentice hall Upper Saddle River.

Russo, L., Vakalis, D., & Siettos, C. (2013). Simulating the wildfire in Rhodes in 2008 with a cellular automata model. *Chemical Engineering Transactions, 35*, 1399-1405.

Sabino, A., & Rodrigues, A. (2009). *A visual language for spatially aware agent-based modeling in crisis scenarios.* Paper presented at the Proceedings of the 12th AGILE International Conference on Geographic Information Science. Hanover, Germany. SANTOS, JA.

Sampath, D. (2004). *ABRCon, Adaptive oBject Re-CONfiguration: an approach to enhance, repeat playability of games and repeat watchability of movies.* Paper presented at the Proceedings of the 2004 ACM SIGCHI International Conference on Advances in computer entertainment technology.

Saoud, N. B.-B., Mena, T. B., Dugdale, J., Pavard, B., & Ahmed, M. B. (2006). Assessing large scale emergency rescue plans: an agent based approach. *The International Journal of Intelligent Control and Systems, 11*(4), 260-271.

Sato, K., & Takahashi, T. (2010). A study of map data influence on disaster and rescue simulation's results. In *Advances in Practical Multi-Agent Systems* (pp. 389-402): Springer.

Schelling, T. C. (1971). Dynamic models of segregation. *Journal of mathematical sociology, 1*(2), 143-186.

Schoenharl, T., & Madey, G. (2011). Design and implementation of an agent-based simulation for emergency response and crisis management. *Journal of Algorithms & Computational Technology, 5*(4), 601-622.

Shiva. (2018).

Shubik, M. (1972). On the Scope of Gaming. *Management Science, 18*(5-part-2), 20-36. doi:10.1287/mnsc.18.5.20

Siddhartha, H., Sarika, R., & Karlapalem, K. (2009). Score vector: A new evaluation scheme for RoboCup Rescue simuation competition 2009. *Rescue Technical Committee*.

Stiny, G. (1980). Introduction to shape and shape grammars. *Environment and planning B: planning and design, 7*(3), 343-351.

Sun, J., Yu, X., Baciu, G., & Green, M. (2002). *Template-based generation of road networks for virtual city modeling.* Paper presented at the Proceedings of the ACM symposium on Virtual reality software and technology.

Sutton, J., & Tierney, K. (2018). *Disaster Preparedness: Concepts, Guidance, and Research*.

Sycara, K. P. (1998). Multiagent systems. *AI magazine, 19*(2), 79.

Taillandier, F., & Adam, C. (2018). Games Ready to Use: A Serious Game for Teaching Natural Risk Management. *Simulation & Gaming, 49*(4), 441-470. doi:10.1177/1046878118770217

Takeuchi, I. (2004). *A massively multi-agent simulation system for disaster mitigation.* Paper presented at the International Workshop on Massively Multiagent Systems.

Tully, D., El Rhalibi, A., Pan, Z., Carter, C., & Sudirman, S. (2015a). *Automated procedural generation of urban environments using open data for city visualisation.* Paper presented at the International Conference on Image and Graphics.

Tully, D., El Rhalibi, A., Pan, Z., Carter, C., & Sudirman, S. (2015b). *Mesh Extraction from a Regular Grid Structure Using Adjacency Matrix*, Cham.

Tully, D., Rhalibi, A. E., Carter, C., & Sudirman, S. (2016, Aug. 31 2016-Sept. 2 2016). *Generating a Novel Scene-Graph Structure for a Modern GIS Rendering Framework.* Paper presented at the 2016 9th International Conference on Developments in eSystems Engineering (DeSE).

Turing, A. M. (1937). On computable numbers, with an application to the Entscheidungsproblem. *Proceedings of the London mathematical society*, *2*(1), 230-265.

Unity3D. (2018).

UnrealEngine4. (2018).

Uskov, A., & Sekar, B. (2014). *Serious games, gamification and game engines to support framework activities in engineering: Case studies, analysis, classifications and outcomes.* Paper presented at the Electro/Information Technology (EIT), 2014 IEEE International Conference on.

Vasudevamurt, V. B., & Uskov, A. (2015). *Serious game engines: Analysis and applications.* Paper presented at the Electro/Information Technology (EIT), 2015 IEEE International Conference on.

Vasudevamurt, V. B., & Uskov, A. (2015, 21-23 May 2015). *Serious game engines: Analysis and applications.* Paper presented at the 2015 IEEE International Conference on Electro/Information Technology (EIT).

Veeraswamy, A., Galea, E. R., Filippidis, L., Lawrence, P. J., Haasanen, S., Gazzard, R. J., & Smith, T. E. L. (2018). The simulation of urban-scale evacuation scenarios with application to the Swinley forest fire. *Safety Science*, *102*, 178-193. doi:https://doi.org/10.1016/j.ssci.2017.07.015

von Neumann, J., & BURKS, A. (1966). Theory of Self-Reproducing Automata (Urbana, IL: University of Illinois).

Walker, W. E., Giddings, J., & Armstrong, S. (2011). Training and learning for crisis management using a virtual simulation/gaming environment. *Cognition, Technology & Work*, *13*(3), 163-173. doi:10.1007/s10111-011-0176-5

Warren, I., & Bach, H. K. (1992). MIKE 21: a modelling system for estuaries, coastal waters and seas. *Environmental Software*, *7*(4), 229-240.

Weiss, G. (1999). *Multiagent systems: a modern approach to distributed artificial intelligence*: MIT press.

Wolfram, S. (1984a). Cellular automata as models of complexity. *Nature*, *311*(5985), 419.

Wolfram, S. (1984b). Universality and complexity in cellular automata. *Physica D: Nonlinear Phenomena*, *10*(1-2), 1-35.

Wonka, P., Wimmer, M., Sillion, F., & Ribarsky, W. (2003). *Instant architecture* (Vol. 22): ACM.

Wooldridge, M., & Jennings, N. R. (1995). Intelligent agents: Theory and practice. *The knowledge engineering review*, *10*(2), 115-152.

Yokoo, M., Durfee, E. H., Ishida, T., & Kuwabara, K. (1998). The distributed constraint satisfaction problem: Formalization and algorithms. *IEEE Transactions on knowledge and data engineering*, *10*(5), 673-685.

Zaitsev, D. A. (2017). A generalized neighborhood for cellular automata. *Theoretical Computer Science*, *666*, 21-35. doi:https://doi.org/10.1016/j.tcs.2016.11.002

Zhong, W., & Kim, Y. (2011). *Searching for sweet spots of communication during an emergency.* Paper presented at the Proceedings of the 11th Biennial and 20th Anniversary Public Management Research Conference.

Zook, A., Lee-Urban, S., Riedl, M. O., Holden, H. K., Sottilare, R. A., & Brawner, K. W. (2012). *Automated scenario generation: toward tailored and optimized military training in virtual*

environments. Paper presented at the Proceedings of the international conference on the foundations of digital games.

Zyda, M. (2005). From Visual Simulation to Virtual Reality to Games. *Computer, 38*(9), 25-32. doi:10.1109/MC.2005.297