Comparing Dynamitic Difficulty Adjustment and Improvement in Action Game

Yi Zu

Comparing Dynamitic Difficulty Adjustment and Improvement in Action Game

Yi Zu

MSc by Research

2016

UNIVERSITY OF BEDFORDSHIRE

COMPARING DYNAMITIC DIFFICULTY ADJUSTMENT AND IMPROVEMENT IN ACTION GAME

by

Yi Zu

A thesis submitted to the University of Bedfordshire in partial fulfilment of the requirements for the degree of Master Research

February 2016

COMPARING DYNAMITIC DIFFICULTY ADJUSTMENT
AND IMPROVEMENT IN ACTION GAME

Y. ZU

ABSTRACT

Designing a game difficulty is one of the key things as a game designer. Player will be feeling boring when the game designer makes the game too easy or too hard. In the past decades, most of single player games can allow players to choose the game difficulty either easy, normal or hard which define the overall game difficulty. In action game, these options are lack of flexibility and they are unsuitable to the player skill to meet the game difficulty. By using Dynamic Difficulty Adjustment (DDA), it can change the game difficulty in real time and it can match different player skills.

In this paper, the final goal is the comparison of the three DDA systems in action game and apply an improved DDA. In order to apply a new improved DDA, this thesis will evaluate three chosen DDA systems with chosen action decision based AI for action game. A new DDA measurement formula is applied to the comparing section.

Author's declaration

I declare that this thesis is my own unaided work. It is being submitted for the degree of Master Research at the University of Bedfordshire.

It has not been submitted before for any degree or examination in any other University.

Name of candidate: Signature:

Yi Zu

Date: 26 February 2016

Dedication

I wish to dedicate to my supervisor Hong Qing Yu who helping me and support me. Without his support, I would give up to finish this project and stay at home. Thanks also to my friend Josh and Lionly, who were help me.

List of Contents

List of Figures

1 Introduction

1.1 Motivation

According to the latest report of the Entertainment Software Association Essential Fact, the total consumer spending on games industry has reached 22 billion dollars in United States in 2014 and the best-selling video game super genres is action game [yz1]by taking 28% in total (ESA, 2015). In general, action game can include traditional action game, role-playing game, shooting game, adventure game, racing game and fighting game (Oxford, 2015). At the meantime, the selling of action game has over 80% market share [yz2]in total. Therefore, action game is an important part of selling product from t[yz3]he last year.

The quality of Artificial Intelligence[yz4] (AI) systems used in games is one of the key factor to make game being more attractive. A high quality AI and the high game enjoyment can lead the differentiator of a game (Forbus & Laird, 2002) . To make the game more fun, game difficulty cannot be too easy or too hard. Malone point it out "Curiosity is the motivation to learn, independent of any goal-seeking or fantasy-fulfilment. Computer games can evoke a learner's curiosity by providing environments that have an 'optimal level of informational complexity.' In other words, the environments should be neither too complicated nor too simple with respect to the learner's existing knowledge. They should be novel and surprising, but not completely incomprehensible." (Malone, 1980)

A game with a balanced difficulty is more fun to play. Portnow shows a powerful skill or strategy can make the games become simple and predictable (Portnow, 2008). Chen, et al. have presented an evolutionary architecture to solve the balance problem in the online Action Role-playing Game (ARPG) by using improved Probabilistic Incremental Program Evolution(PIPE) with Cooperative Coevolutionary Algorithm(CCEA) (Chen, et al., 2012). Leigh, R., Schonfeld, J. and Louis, S. have combined an evolvable agent, a visualization and coevolution that can effectively balance the game of CaST (Capture, Steal, Territory) (Leigh , et al., 2008). Details on the definition of DDA and action[yz5] game are discussed on latter sections.

1.2 Questions and Objectives

Known the research motivations argued in the last section. This aims of the project are to compare[yz6] three different Dynamic Difficulty Adjustment (DDA) syst[yz7]ems. This research project consists of two major parts:

The first part is a research comparing on three related action Artificial Intelligence (AI) system in action games by analysing the advantage and disadvantage of these techniques. The research questions being answered in this thesis are as follows:

- Doing research on three AIs, find out how each action game AI works in the action game. These AIs are focusing on the character actions.

- Analysing the advantage and the disadvantage of each action AI. Chose one of the action AI for following research.
- Research three DDA system[yz8]s, find out how each DDA system effects the game difficulty by using the chosen action AI from previous[yz9] section.

The second part of this project consist the evaluation for future use. It will base on the first part of research. A demo game will be created to show how each AI effects[yz10] the game difficulty during playing. Within the demo, player can choose which DDA they want to play with at the start menu and start to play. As a result,

- Using an artefact to evaluate the advantages[yz11] and disadvantages [yz12]of each DDA system
- Create an improved DDA system that based on the previous result and analysing.
- Designing a game by using the existed[yz13] DDA and improved. It will be more focus on the AI function rather than game interface, display and mechanics.

1.3 Methodology

1.3.1 Online background research

Online research method is the main research method to research the problem domain and the related background. The research tool in this thesis used the EBSCO Discovery Service which [yz14]is provided by the University of Bedfordshire Learning Resources Team. EBSCO Discovery Service provided a powerful tool that researcher can access all of institution's information resources through a single search. The resource has combined both library resource and the online web based sources.

Serval games are played during the research. To fully understanding the game AI system and the difficulty adjustment system in action game. Playing serval action games which they selling on the market is necessary. During the game play, researcher is focusing on the game AI and how the difficulty adjustment can improve the game enjoyment. Also, it can provide the experience on action game to help the research.

1.3.2 Experimental game design

Experimental game design is used in computer[yz15] science to evaluate new solutions for a specific problem. Experimental game design is divided into two steps. At first, the researcher is required to find some measurements that can help whether the questions should be asked about the system before the evaluation. Then a well-designed experiment artefact will start with a list of the questions that the experiment is expected to answer.

1.3.3 Build

A “build" research methodology consists of building an artefact to[yz16] demonstrate that it is possible. To be considered research, the construction of the artefact must be new or it must include new features that have not been demonstrated before in other artefacts.

1.3.4 Methodology Summary

The methodologies are used in this paper has been introduced[yz17] on previous section. They are online background research, documentary analysis, experimental game design and build. Online background research is researching the related topic background information with the provided research tools. Experimental game design is designing a game that can ask the questions. The final build is build an artefact which include the new feature.

1.4 Summary of Thesis Structure

The structure of this thesis is as follows:

Chapter 2 defines the term “action game”. Action game is the main area of the research. It will explain what it is, where it comes from.

Chapter 3 gives the background of action AI which included three action AI[yz18]s could be use in the action game in general. Discuss their advantage[yz19]s and disadvantages.

Chapter 4 introduces background about the game difficulty adjustment and how to measure the game difficulty. It also present three game difficulty adjustment system[yz20]s during the research.

Chapter 5[yz21] show the artefact design such as the GUI, AI commination.

Chapter 6 will present how those three difficulty adjustment[yz22] systems are implemented in the artefact and evaluated the data from the artefact.

Chapter 7 is the conclusion of this thesis.

2 Background

In order to understand what kind of game this thesis use, this chapter describes the background of the action game in a general way. At first, it defines the definition of action game from the research. Then it defines the key element of action game such as level, skill, live and graphic. These key elements can help the research whether the artefact is an action game or not.

2.1 The definition of Action game

The first electronic game was created in the year 1958 named as ‘Tennis for two’ (BNL, 2014). . It can be observed that videos games are gaining popularity day by day and it is expected that the greater number of people are familiar with at least one[yz23] video game. Nowadays, videos games are categorized into various types such as real-time games, role playing games, strategy games, racing games, action games, fighting games, adventure games and so on (Hurst , 2015). Action games is one of the main videos game type. In an action game, player usually control the major character of the game and the character can battling enemies, collecting item and avoiding attack such as Legion of Zelda series (Zelda.com, 2014), Monster hunter 错误!未找到引用源。 series (Monster Hunter, 2015) , Devil May Cry series (Capcom, 2015), Ys series (Falcom, 2014) and Assassin’s Creed series (Ubisoft, 2015). In this thesis, the two terms namely ‘video game’ and ‘computer game’ refers to the same types of games.

Oxford N. is one of expert in Nintendo DS. She point it out “Video games in the "action" genre typically put emphasis on challenging the player's reflexes, hand-eye coordination, and reaction time.” (Oxford, 2015)

In general, whenever there is an action as the main play style in the game, it belongs to action game (Oxford, 2015). The type of game includes different subtype such as fighting game, shooting game and platform game which are generally included the most important part of action games, although some real-time strategy games are also included to be action games. There are many action games has similar design. Action game designer usually design a multiple attack type. Character attack type can be shooting, sword attack and magic attack such as fire ball and thunder attack. The player usually controls the main character of the game and growths from a low level to a high level. At the same time, the enemy will be increased at a steady rate. The terrain will become more dangerous and more difficult to defeat the enemy. In most of the action game, it usually appears "the boss fight" after a certain level. The enemy called “boss” are usually big and player need more time and power to defeat it. Some action game also appears a smaller enemy on the halfway through.

2.2 Defining action game elements

There are some comment elements which can represent the game is action game. This section introduces the major elements in action game. It does not mean every action game have to include all of elements but most of them they have.

2.2.1 Character States

Character states is the basic element in action game such as hit point, lives, mana point and character. In action game, each character usually has health, lives and ability states. Health bar will display on the monitor to represent the current health point (HP). If character was attacked, the health bar will be decrease a certain value. Once the health point reach to zero, the character dies and a “life” is lost. Player have to start again from the previous safe point and play again. Once the number of life reach to zero, player cannot start it again and its game over. In action role-playing game (ARPG), character usually has with a low HP value around 10 or 20 at the beginning of the game and the HP value will be increased when the character level is increased.

Figure 1 shows the game Assassin’s Creed IV Black Flag (Ubisoft, 2015). Player only get 4 hit points at the start and it will be increased thought game progress.

Figure 1. Assassin’s Creed IV Screenshot

Ability state includes the character’s attach value, defence value and others states. In a traditional action game, character states will not be a specific value like massively multiplayer online role-playing game[yz24](MMORPG). The action game designer will design some item or level to double increase attack value or decrease half of defence value.

2.2.2 Character ability

In most action games, the protagonist usually has a range of abilities and skills such as defences, attack, shooting and punching. Many action games designer make a powerful attack that terminates the enemy with a high attack damage, but this attack skill is limited. Players may find a power-up item within the game world that grants temporary or permanent increased their abilities. For example, player may get an item that increase attack speed or

more powerful attacks, or a shield from attacks within a limited time. Some action games even allow players to spend upgrade points on the power ups of their choice. In Devil May Cry 4 (Capcom, 2015), player have a set of combo with a specific weapon. By inputting a set of command like “jump, Y, Y” the controlled character will do the specific combo attack.

2.2.3 Level

Level in action game is including the power of the characters, difficulty level and game stage. As the power of a character, player character usually starts from level one. By getting more and more experience from kill monster, the level of the character will be increased to level two or more. With the level increasing, the character states will be increased as well such as max hit point, attack point. In ZWEI 2 (Figure 2), player can level up the character to level 99 and the character state are increased each level once it gain

Figure 2. ZWEI2 Character States

As a game stage, each level will include all kinds of challenges, the player must solve a puzzle in order to walkthrough the game. In the Legion of Zelda: Ocarina of time (Zelda dungeon, 2014), the game designer is focusing on the stage on each area. Game stage could be linear or nonlinear, sometimes including keyboard shortcuts. In some stage, players may need to find an exit hidden enemy. These stage can also contain the secret and hidden or hard-to-reach objects or contains a value. Reward can be an additional item or non-standard exit, allowing the player to access a hidden level, or jump in previous levels. Action games provides a teleporter, sometimes cause the player's avatar reappear elsewhere at the same stage. Stage normally use locked door which can [yz25]open with a particular key found in other parts of the level. As seen in Figure 3, the player got a key to open the door in order to get to the next room.

Figure 3. Legion of Zelda: Ocarina of Time

For the game difficulty level, the game designer will let the player choose the difficulty level before the game start such as easy, normal or hard. And most of the game will not change the difficulty level during the game. Comparing the normal mode and easy more, hard mode in action game usually increase the enemy hit point and attack point to make the game more difficult. Game designer sometimes uses time limit give player pressure feeling in order to increase the game level. Once the timer expires, the player is usually game over. If there is remaining time, it usually increases the player's score. Game difficulty level introduces more detail in the section 4.

2.2.4 Point of View

Nowadays, there are either 2D or 3D from a variety of viewpoints in action game. 2D action games typically use a side view or a top-down view. In a side view action game, the screen is frequently scrolled as the player discovers the stage, although there are many games scroll through the stage automatically to push the player advancing. For example, Mega Man 10 (Moriarty, 2010) is a classic Japanese side-scrolling action game. Figure 5 is the Mega Man 10 screenshot. In the game, player need control a character to run from left to right in order to reach to finish point.

Figure 4. Mega Man 10

In 3D action games, the perspective is usually a first-person perspective or a third-person perspective. However, some 3D games offer a context-sensitive perspective that is controlled by an artificial intelligence camera. Most of what the player needs to know is contained within a single screen, although action games frequently make use of a heads-up display that

display important information such as health or ammunition. Action games sometimes make use of maps which can be accessed during lulls in action, or a mini-map that is always visible.

2.2.5 Scoring and victory

By setting simple goals is important in action game as well. A common goal is that player need to defeat a boss. This is often presented in the form of a structured story, with a happy ending after winning the game. Many action games, game designer usually keep tracking the player's score or points. Points are awarded for completing certain challenges, or defeating certain enemies. In Devil May Cry series, scored is depends[yz26] on the player attack combo. The more combo player can make; the more point the player will get. In the other way, scoring system is the game designer to evaluate the player skill level.

2.3 Summary

In this chapter, it introduces the definition of action game. Action game is a type of game that focus on challenging the player's reflexes, hand-eye coordination, and reaction time. In the second part, it introduces the key elements in action game which included character states, character skill, level, interface and scoring. Character states represent the power of the character. Character skill is a key[yz27] thing to make the game become action game. Interface is decided which viewport is going to play. Scoring evaluates player skill level. No all the action game includes all these elements. Some of the game may focus on one of the key points [yz28]to become their feature[yz29]s to sell instead of keeping all the feature[yz30]s.

3 Artificial Intelligence in Game

This chapter discusses what kinds of artificial intelligence were found during the research. At the start, it introduces the definition of artificial intelligence in general. In the second section, it describes several types of the AI which are used in videos games. In the third part of this chapter, it will introduce three action game AIs [yz31]in detail and discuss their advantages and disadvantages. This[yz32] research will also find out which action AI is the best AI to be use in the later section.

3.1 The definition of artificial intelligence

There are several definitions about artificial intelligence. Historically, Russels, Norvig and Davis (Russell & Norvig, 2013) has introduced eight definitions of artificial intelligence which organized into four approaches.

- Thinking Humanly:

" And the epitome of entire drama is artificial intelligence, the exciting new effort to make computers think… AI wants only the genuine article: machines with minds, in the full and literal sense" (Haugeland, 1985)

- Thinking Rationally:

"Artificial intelligence is the study of mental faculties through the use of computation models" (Charniak & McDermott, 1985)

"The study of the computations that make it possible to perceive, reason, and act." (Winston, 1992)

- Acting Humanly:

"The art of creating machines that perform functions that require intelligence when performed by people." (Kurzweil, 1990)

"The study of how to make computers do things at which, at the moment, people are better." (Rich, E. and Knight, K., 1991)

- Action Rationally:

"Computational intelligence is the study of the design of intelligent agents." (Poole, et al., 1998)

"Artificial Intelligence (AI), broadly (and somewhat circularly) defined, is concerned with intelligent behaviour in artefacts." (Nilsson, 1998)

3.2 Type of Artificial intelligence using in the video game

This section will briefly introduce some popular applied AI types that can be used in the videos game based on Steve R (Rabin, 2002). He has listed nearly all of game AI's which are based on Russell S. and Norvig P. (Russell & Norvig, 2013) .

3.2.1 Expert system

In a generic form, an expert system is a rule-based system designed to make sense of a very small, specific set of inputs (Copley, 2010). In a broad sense, most AI's are expert systems at some level. As seen in Figure 5 it included an interface, knowledge database and an inference engine. In knowledge database, the data is stored in the form of simple rule. These rules can be an if/then statement, or a series of branches. All of these rules are hand-coded to inference engine to calculate all the user queries from user interface and produce output back to interface. This is usually used in strategy game like chess (Copley, 2010).

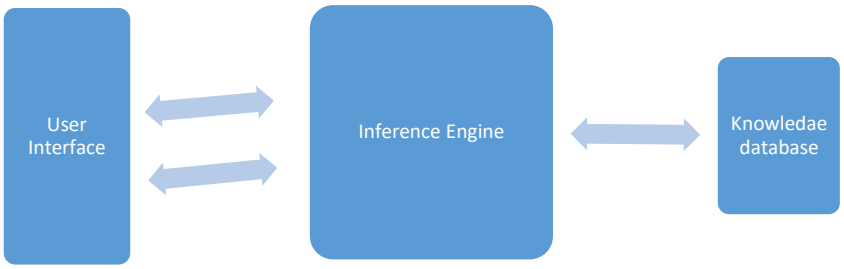

Figure 5 Exper system

3.2.2 Case-based reasoning in game

In 2008, Szczepanski used Case-Based Reasoning (CBR) for improved micromanagement in real-time strategy games (Szczepański & Aamodt, 2008). The CBR model they used in the system architecture is based on Aamodt A. and Plaza E. (Aamodt & Plaza, 1994). Case-Based Reasoning is a simple method to solve the problem using a currently case. In Aadmodt and Plaza model, the idea is to retrieve the most similar case from the case database at the start. After all, it reuses the case to attempt to solve the current problem and revise the proposed solution if necessary. Until confirming the solution, it will retain the solution as a part of a new case. This analogy is shown in[yz33] Figure 6 case based reasoning.

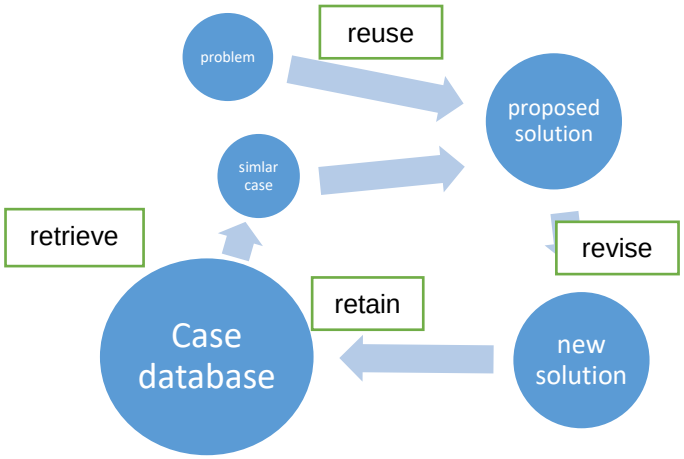

Figure 6 case based reasoning

3.2.3 Finite state machine

Finite state machine represents behaviours using decision tress that consist of a limited number of states (Cryan, 2004). States is only a condition. For example, the state of a window can be open or closed. It is commonly used in the game AI programming. Programmer and game designer would design a set of states that the character will act such as run, stand, jump and attack. In Figure 7, each character holds one state at the same time. The character will keep doing the same thing until some condition or event has been change. During the state, the state machine will keep tracking and recording the character's current state.

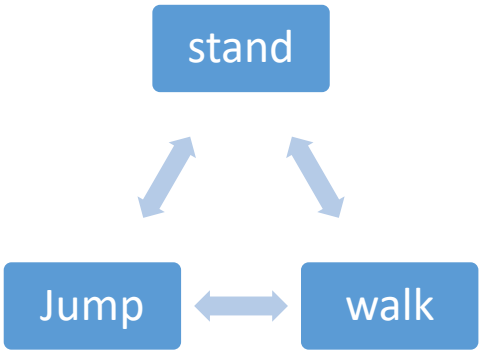

Figure 7 finite state machine

3.2.4 Decision trees

A decision trees is a true or false based algorithm (Rouse, 2012). There is only one decision from the starting point which is the root of the tree. One or more decision branch follows the root which is the leaf. Each choice of decision is made based on the character sensor. Character sensor is usually built in the game. A decision trees is a simple and fast decision mechanisms. As seen in, it typically checks the value or action whether it is true or false. If it is true, then the character does action A. If it is false, then the character does action B.

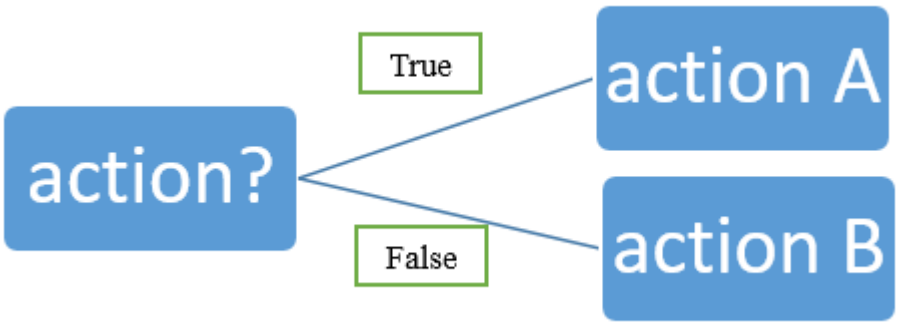

Figure 8 Decision trees

3.2.5 Search algorithm

Search algorithm is a universal technique of problem solving in AI. It focuses on finding a sequence of actions or states within a graph. It is usually used in single-player puzzle games. A search algorithm includes single agent pathfinding problems, brute-force search strategies, informed search strategies and local search algorithms. Brute-force search strategies include breadth-first search, depth-first search, bidirectional search, uniform cost search and iterative deepening depth-first search. Informed search strategies included heuristic evolution functions, pure heuristic search, a* search and greed best first search. For the local search algorithms, it includes hill-climbing search, local beam search, simulated annealing and traveling salesman problem (Verma, et al., 2010)

3.2.6 Flocking algorithm

The modern flocking algorithm was introduced by Reynolds in 1987 (Reynolds, 1987). He assumes a group that is simply the effect of the relations between the actions of the individual birds or flying object. In the video game, flocking focus on the non-player character movement in the virtual world. It is a sub gen of Artificial live.

3.2.7 Genetic algorithms

The first Genetic Algorithm (GA) were described by John Holland in 1975 (John, 1975). He aimed to understand natural adaptation as a mechanism model and imported them into computer systems. In 1995, Mitchel described the demand of using ideas from evolution to solve computational questions and he also gave a detailed example of how a GA was used on automatically discovering improved strategies for playing the Prisoner's Dilemma. (Mitchell, 1995). In 2011, Martin M. has used a genetic algorithm to create adaptive enemy AI (Martin , 2011).

3.2.8 Neural networks

An artificial neuron network (ANN) is a computational model based on the structure and functions of biological neural networks (Anon., 2015). Information that flows through the network affects the structure of the ANN because a neural network changes - or learns, in a sense - based on the input and output (Anon., 2015) . ANNs are considered nonlinear statistical data modelling tools where the complex relationships between inputs and outputs are modelled or patterns are found (Anon., 2015).

3.3 Artificial intelligence in action game

In the last section, this thesis introduces several[yz34] type of AI's that can be used in video games[yz35] such as finite-state machines, decision trees, fuzzy logic etc. In this section[yz36], the researcher will focus on the action decision making AI during the research.

3.3.1 Deduction countermeasures by using K-nearest neighbour algorithm

In the first subsection, the aim is to introduce Yamaoto research about how K-nearest neighbour algorithm deducts the countermeasures in fighting game. (Yamamoto, et al., 2014)

3.3.1.1 What is K Nearest Neighbour algorithm

K-nearest neighbour (kNN) algorithm has started to be used as a non-parametric technique in1970's. (Sayad, 2010). By giving the integer "K" such as three, unknown value "A" need to calculate the distance to other values on the sample map and take three values with the shortest distances. If two of them have the same value, then K become the same value as those two values. (Sayad, 2010)

In 2014, Yamamoto K. et al., proposed an artificial intelligence algorithm that uses the k-nearest neighbour algorithm to predict its opponent's attack action and a game simulator to deduce a countermeasure action for controlling an in-game character in a fighting game (Yamamoto, et al., 2014) . In his approach, there are four algorithms need to be used in order to predict an opponent's attack action.

3.3.1.2 Algorithm 1 Collect data

The first algorithm is the function of collecting date from the game. He created a simple function that based on if/end to save character location and state. For example, "collectData ([30,12],[60,12], gg)" means that our character location is [30,12], enemy location is [60 12] and both of the characters are standing on the ground.

3.3.1.3 Algorithm 2 decide Action

The function of the second algorithm is to decide the action after collecting the data. Before the AI attack, the AI needs to predict whether the enemy is going to attack or not. Therefore, the AI needs to get the current relative coordinates and the AI can classify the enemy's attack

action using the kNN algorithm. The AI's decides[yz37] its own action based on the result from the simulator.

3.3.1.4 Algorithm 3 simulate

The simulator is combined within the AI and conducts simulation with all combinations of the opponent's predicted attack actions and each of the actions which can be performed by our AI. For each combination, the simulator simulates the game up to one second from the current time. The AI then chooses the action with the highest evaluation value as its next action. The evaluation value for each action of the AI is determined by the amount of damages of the opponent minus that of the AI.

3.3.2 Markov Decision process

SARSA algorithm is being used in resent research to learn how to fight by Graepel et al. (Graepel, et al., 2004) . The action decision process they used is Markov Decision Process (MDPs). A Markov Decision Process model contains four processing elements which are a set of possible world states, a set of possible action, a real valued reward function and a description of each action effects in each state.

The policy in Markov Decision process can provides some rule in different state, the future and the past to the AI agent such as 'T= (T1,T2,T3…Tn+1)'. T is the state and 'n' is the rule. For example, the AI agent need an optimal plan or a sequence of action to defeat player in action game. For the MDPs, the AI are looking for an optimal policy$\pi' : S \rightarrow A$. It means the policy π gives an action 'A' for each states. In each states, there are a few action are available. For example, blocking is only available when the character on the ground. By given the chance of state transition and some suitable reward, MDPs is a state based decision algorithm. However, if game designer makes each action in action game as a state and the user input was considered as the MDPs action, it may get different result for a skill based MDPs. The reward function can make AI agent to get as much reward as he can get. For example, there are two action, 'shot' and 'jump'. If shot reward is higher than jump reward, AI agent will choose 'shot' action. A discount rate will discount reward by a specific value after passing one or more stages[yz38].

3.3.3 Real-time Imitation Based Learning for commercial fighting game

At the last section, it will introduce a technique to improve an imitation based AI for fighting games (Lueangrueangroj & Kotrajaras, 2009).

3.3.3.1 What is Imitation Based Learning AI

Imitation learning is a general mechanism for rapidly acquiring new skills or behaviours in humans and robots (Dautenhahn & Nehaniv, 2002). Lueangrueangroj and Kotrajaras's approach divide[yz39]d the imitation based AI into imitation process and learning process. (Lueangrueangroj & Kotrajaras, 2009).

In the imitation process, The AI determines the player's actions whether exited in the case or not. If the case existed, the AI will check if the player's action already exists in the list of actions. Otherwise, the AI will create a case and record it into the database. If the player's action already exists in the list of action, AI will increase the frequency of the action, if not, the AI will add the action with the initial frequency.

Learning process is based on the damage weight balance. If the AI was damaged, it will increase the weight of the action that player uses and it decrease the weight of that action the AI uses as well. On the other hand, if the player was damaged, the AI will increase the weight of the action that the AI uses and it decrease the weight of that action the player uses.

3.3.3.2 Data requirement for AI

In the approach from Lueangrueangroj and Kotrajaras, the AI requires six types of data that are collected from the game. The first one is the distance between the AI and player. The second one is the distance between the AI and a projectile weapon such as fire ball or energy ball. The third one is the state of the AI such as jump, run or attack. The fourth one is the list of actions that the player can performed. The fifth one is the frequency of the action, and the last one is the weight for each action.

3.3.3.3 Case processing

When playing against a human opponent, the AI would check if the opponent's state matched the situation details in one of our cases. The AI would check the followings:

- The distance between the player and the AI in the x-axis and the y-axis.
- The distance between the player and a projectile weapon on the x-axis and the y-axis.
- The player's state matched with the AI's state in the case.

If all values matched, it meant the player behaved in the same way as the AI when that case was created. The AI would then be able to choose an action from the player's action list to perform in order to imitate the player. If there was no match, the AI would perform a random action in order to create more chances for a matching case. Any random action that damaged the human-control character would not be used to update the weight because we only wanted to adjust actions obtained from imitation.

A player's action has its frequency and weight. The frequency reflects the imitation aspect and the weight reflects the learning aspect. To choose an action from the list of actions from a case, these two values have to be considered. Actions would be chosen based on probability. This process of choosing an action could be divided into two parts.

First, the AI uses the frequency value of each action to calculate a percentage value for each action. This percentage value was the probability for each action to be chosen based on imitation alone. The frequency based probability was represented by P_f. Second, the AI

considered the weight value of each action and used it to calculate a percentage value for each action. This percentage value was the probability for each action based on the learned success rate of the action. This probability was represented by P_w.

According to Lueangrueangroj and Kotrajaras's approach, the AI will adjusted a probability value for each action. P_f is a base value. The AI used P_w value to adjust P_f value. The adjusted probability result for an action was represented by P_a. The equation they used is showing below:

$$P_a = P_f + (C \times (P_f - P_w))$$

C is a constant value and set to 0.1 in their experiment. They do not explain the reason for setting value C as 0.1 so constant value C is not a fixed value. It can be set to other value such as 0.2 or 0.3. [yz40]

3.3.4 Comparison

This section will compare the above three AIs and generally discuss the advantage and disadvantage, chose one of these AI to be used in the artefact and explaining the reason. Because the aim of this thesis is to focus on the fighting game difficulty adjustment, the considering element should relate to the topic. The researcher suggests some elements that can help to compare three different AI's to understand their advantage and disadvantage.

3.3.4.1 General Overview

At the beginning of the overview, all of the AI's can be used for 'player vs AI' mode. They are not going to use in 'player vs player' which is not the research topic of the thesis. The next question is whether they support 'multiple players VS multiple AIs' mode or not. The KNN algorithm and the real time limitation learning are used for commercial fighting games. The KNN algorithm is based on the player action that is saved before changing the fighting style, so 'multiple players VS AI' mode is not suitable. With two or more AI agents in the fight scene, the KNN algorithm may perform a different result that compares to one AI agent. In the other hand, the real time limitation based learning is based on the existing case in the case base. The case base can contain multiple players or multiple AI agents. Comparing to the KNN algorithm, the real time limitation based learning can be used in other fighting games. The real time limitation based learning and reinforcement learning contains the skill frequency adjustment. This can be adjusted by the case value and the previous state during the game. Both of them could be also used on both single player mode and the multiple player mode which are suitable for this research. The advantage of the Reinforcement learning is that it does not need any detail or case base from the new player before the first play. All of these three AI's need to present a skill or action rule in order to install the algorithm into the game.

3.3.4.2 Table

The table below shows the comparison for each AI. In general, all of the three AI's can be used on the artefact. However, reinforcement learning is based on skill and state to make the action decision which KNN algorithm and the real time limitation based learning does not support. The reinforcement learning without improvement is not based on skill however Graepel et al. used SARSA algorithm to improve reinforcement learning with skill based system (Graepel, et al., 2004) . They used three states to decide which skill the AI can use. By changing different states, AI can use different skill based on the states. Therefore reinforcement learning can be either skill base or state base. [yz41]

On the other hand, reinforcement learning can be use on every new player who never played this game before. It is an important feature that it can reduce the effect element for the later DDA comparison testing, because the tester or player will become a new player for the AI. At the end, the researcher choses the reinforcement Learning for the later section.

Element	K Nearest Neighbour algorithm	Markov Decision Algorithm	Real Time Limitation Based Learning
Player vs AI(s)	√	√	√
Multi Players VS AI	x	√	√
Multi Players VS Multi AIs	X	√	√
Preset action order/rule	√	√	√
Skill based	√	√[yz42]	X
State based[yz43]	X	√	X
Used in Fighting game platform before	√	√	√
Need case before the first play[yz44]	√	X	√

Table 1 action AI comparison

3.4 Summary

This chart has briefly introduced AI in the game. At the start, the definition of AI is defined with different researchers. The second section introduced some popular AI's that can be used in action games. In order to install the DDA system on a later section, this thesis also introduces three action AI's which are KNN algorithm, Markov Decision Algorithm and real time limitation based learning. In summary, this thesis discussed about the comparisons between the above 3 AI's to make a decision as to which AI would be used for the game. The result is Markov Decision algorithm which will be install on the game artefact on later section.

4 The level of difficulty in game

This section is going to talk about game difficulty. The first subsection will briefly introduce what game difficulty is and how people scale the difficulty level in the game. An improved game difficulty measurement for action games will be introduced based on existing game difficulty measurement approach. There are three different DDA approaches [yz45]which will be briefly introduced in the second part.

4.1 Game Difficulty Scaling

In order to understand dynamic difficulty adjustments in video games, the researcher is necessary to know what difficulty level is and how other researchers' measure and scale the difficulty level at the modern time.

4.1.1 Difficulty Level

As chapter three mentioned, Video games designer usually allow players to affect the game balance by giving a choice of "difficulty levels". These affect how challenging the game is to play. In addition to altering the game's rules, difficulty levels can be used to alter what content is presented to the player (Croshaw, 2010). This usually takes the form of adding or removing challenging locations or events, but some games also change their narrative to reward players who play them on higher difficulty levels or end early as punishment for playing on easy. Difficulty selection is not always presented bluntly, particularly in competitive games where all players are affected equally and the standard "easy/hard" terminology no longer applies.

4.1.2 Scaling Difficulty Level

Scaling game difficulty level is a part of the game design (Boutros, 2008). Juul has explain why difficulty scaling is so important in game design:

"A game is a rule-based formal system with a variable and quantifiable outcome, where different outcomes are assigned different values, the player exerts effort in order to influence the outcome, the player feels attached to the outcome, and the consequences of the activity are optional and negotiable." (Juul, 2003)

This elaboration and definition provide the researcher with a valid idea of how exactly the game engine behaves, and takes into account the most integral parts of the past definitions. However, in order to support this paper, the researcher is required to explain in detail why 'difficulty' is an important aspect of gameplay. The reason why is that the player needs to put effort into gameplay, and overcoming the difficulty is the main purpose of the game. To highlight the important components of gameplay, and primarily, the relationship between prioritizing difficulty and making an interesting game, it is important to create a definition which leaves aside the dynamics and structure of the game and pays attention on video games from the perspective of the players.

4.2 Dynamic game difficulty balancing

In this section, it will introduce what dynamic difficulty is in game.

4.2.1 What is Dynamic game difficulty

Dynamic game difficulty balancing, also known as dynamic difficulty adjustment (DDA) or dynamic game balancing (DGB), is the process of automatically changing parameters, scenarios, and behaviours in a video game in real-time, based on the player's ability, in order to avoid them from becoming bored (if the game is too easy) or frustrated (if it is too hard). The goal of dynamic difficulty balancing is to keep the user interested from the beginning to the end and to provide a suitable level of challenge for the user.

Traditionally, game difficulty increases steadily along the course of the game (either in a smooth linear fashion, or through steps represented by the levels). The parameters of this increase (rate, frequency, starting levels) can only be modulated at the beginning of the experience by selecting a difficulty level. Still, this can lead to a frustrating experience for both experienced and inexperienced gamers, as they attempt to follow a preselected learning or difficulty curve. Dynamic difficulty balancing attempts to remedy this issue by creating a tailor-made experience for each gamer. As the users' skills improve through time (as they make progress via learning), the level of the challenges should also continually increase. However, implementing such elements poses many challenges to game developers; as a result, this method of gameplay is not widespread.

4.3 Literature review on DDA approaches

Different DDA approaches are found in the literature. In order to create an easier challenge or a harder challenge level, Hunicke R. and Chapman V.'s approach controls the game environment settings by using helmet system (Hunicke & Chapman, 2004). For example, if the game gets too hard, the player will recover items to restore some HP. This is called the environment based approach. They establish some metrics for assessing the statistical data within the game world. Their approach was tested on shooting games. The researcher may consider their approach into the research.

In 2014, Nakagawa et al propose a method for online adjustment of the AI's strength by using the k-Nearest Neighbour (KNN) algorithm (Nakagawa, et al., 2014). This approach is basic on the AI's strength adjustment. By using KNN algorithm, AI's strength can be suitable with player skill level. The approach is skill based or action decision making based. It may be suitable for this research as it is for an action game.

In 2010, Linddao, et al created a appropriate challenge level game opponent by the use of dynamic difficulty adjustment. (Linddao, et al., 2010) They used the Dead-End game as their test-bed. They proposed "time-constrained-CI" DDA and "knowledge-based-time-constrained -CI" DDA by using Monte-Carlo Tree Search. As a result, the "time-constrained-CI" DDA is

not applicable for multiplayer online game and the “knowledge-based-time-constrained -CI” DDA is more stable.

In 2011, Chin et al presented two adaptive algorithms based on reinforcement learning and evolutionary computation to scaling the difficulty of the game. (Chin, et al., 2011) Their difficulty scaling is done in real time 2D driving game. They have been given seven driving behaviour components that will automatically select by adaptive algorithm.

4.3.1.1 Using in recent video games

Archon is a 1984 action strategy game which was developed by Free Fall Associated on PC (Anon., 2010). The enemy in Archon can slowly adjusts over time to help the player’s victory. Dan Bunten Berry designed Multiple Use Labor Element (M.U.L.E.)[yz46] (Ruthner, 2015) to dynamically adjust gameplay between players in 1983 (Plunkett, 2012). SiN Episodes (IGN, 2006) released in 2006 dubbed a "Personal Challenge System". SiN Episodes will adapt itself to the player's skill level. The Challenge system also varies the skill, statistics and toughness of enemies. According to the player's performance, the enemy will ensure a suitable level of challenge. On the other hand, the enemy are able to help the player playing through the whole game.

The 2008 video game *Left 4 Dead* uses a new artificial intelligence technology dubbed "The AI Directo[yz47]r" (Booth, 2009). The AI Director is used to generate a different experience for the players each time the game is played. It monitors individual players' performance and how players work together as a group. The AI Director also decide the number of zombies that attack the player and the location of boss which infected encounters based on receiving game state. During the game play, the AI Director also controls some video and audio elements of the game to get the players' attention to a certain area.

4.4 Dynamic Difficulty Adjustment in video game

This section will introduce three DDA systems during the research. The researcher will focus on the detail of each algorithm and how they adjust the difficulty with the given DDA approach. The first DDA approach is the Hamlet system which was introduced by Robin Hunicke and Vernell Chapman in 2004 (Hunicke & Chapman, 2004). The second approach is the online adjustment with k-Nearest neighbour algorithm which is written by Nakagawa et al. The third approach is basic on the Factor of Difficulty Control (FDC) and Factor of Users’ Adaptation (FUA). These three DDA approach will apply to the game artefact on later section. The implementation of each DDA will be discussed on later section.

4.4.1.1 Hamlet system proactive

According to Hunicke R. paper, they described a DDA system that basic on the supply and demand. Hamlet is a DDA system which was built in the *Half Life* game engine (Hunicke & Chapman, 2004). In the article, author did not propose a specific algorithm or detailed

adjustment system in hamlet system. However, they briefly introduced the whole idea about how the hamlet system works.

In the Hamlet system, adjustment is included in action adjustment, cost adjustment and policies. Hamlet supports two types of adjustments actions. The first one is reactive actions which will adjust the displaying elements such as player HP bar, weapons strength and attack values. In the other hand, proactive actions will adjust some hiding elements such as enemy spawning order or some inactive object.

Each of the Intervention actions are need to be determined a cost value. The individual adjustment actions will be effected by the given observation elements including:

- The player's current location
- How far along the player is in the game?
- The frequency of the player died at this location or level.
- The frequency of the player has repeated the current encounter
- The number of times that the system has intervened in the past

Hamlet can combine the adjustment actions and the coast calculations to build a control model that can control the supply and demand of items. They also provided two examples. The comfort policy which will keep the HP of the player about 50% health and the enemy will be shot less often and less accurately. The discomfort policy will try to keep player fighting with 10 to 20 % HP left. The enemy's accuracy, ammo and health will be relatively increased.

4.4.1.2 Dynamic Challenging Adapter

In 2012, Shin-Hung Chang and Nai-Yan Yang created a new dynamic Challenging level adapter for real-time strategy games. They analysing the basic key elements such as attack power, health point, population and cost experimental platform is basic on Star Craft II (SC II). In the SC II, it has built-in three level of AI (easy, normal and hard). A hard AI can always win a normal AI and a normal AI can always beat an easy level AI. They define a challenging rate (CR) formula which included those four key parameters which can represent the power of the player or AI. It also underlies a dynamic challenge adjustment (DCA) mechanism. The DCA mechanism can decide whether the AI should improve the power such as producing warriors or not. As a results, it successfully reduces the difference between player CR and AI CR. In the meantime, the AI power is adapted to the player skills.

4.4.1.3 Factor of Difficulty Control and Factor of Users' Adaptation

In 2007, Um, et al proposed an algorithm on a falling blocks puzzle game with the factor of difficulty control (FDC) and the factor of user's adaptation(FUA) that they introduce (Um, et al., 2007). They think the FDC and FUA are the key point for auto balancing of difficulty. In the other hand, he analysed each game genre which can apply the dynamic difficulty controlling game system (DDCAS). For the action game, they think the order of the actions

are the pattern of controlling game difficulty (PCD) basis factor of FDC. The Variation of score or enemies are the controlling difficulty levels basic factor of PCD. (Um, et al., 2007)

4.5 Dynamic game elements

From the last two sections[yz48], three different DDA have three different game difficulty controlling systems. They included some similar elements that might be changed via dynamic difficulty controlling include:

- Health of enemies
- Frequency of player action
- Frequency of enemy action
- Power of player
- Power of enemies
- Duration of gameplay experience
- Difficulty level measurement and new challenging rate model

In order to compare three DDA systems, measuring the difficulty level into a mathematical way is essential. There are four measurement elements that are important to measure the level. In 2012, Mealha et al present a preliminary model proposal for analysing video game level scenarios (Mealha, et al., 2012). The models suggest that video game level analysis consists of four elements: videos game; player, metrics and gaming experience. At first, the genre of the game needs to be confirmed before analysis. The second of the elements 'Players' represent a player behaviour in the game such as motivation and skills. The term 'Metrics' describes the gameplay and behaviours metrics such as time, energy, health, player action and location coordinates. The last element is 'Gaming Experience'. Gaming experience controls a selection of attribute[yz49]s that may be consider with the model.

This thesis is focusing on the action game therefore the first element 'video game' is action game. For the second element, the game artefact has to be well designed become an action game in order to make sure that the player has more motivation and more gaming experience.

This thesis defines a measurement formula bas[yz50]ed on the key variable 'hit point' [yz51]to represent the difficulty level as the following equations:

$$\text{Difficulty Level} = \sum_{t=1}^{n} (H_{enemy}^{t-1} - H_{enemy}^{t}) - (H_{player}^{t-1} - H_{player}^{t})$$

Equation 1 Difficulty Level

This formula can define the difficulty level in an action game. The key idea in formula is based on the difference between enemy HP and player HP. "H_{enemy}^{t}" is the enemy HP value at the

't' second. Value "H_{enemy}^{t-1}" represents the enemy HP at t-1 second. "$H_{enemy}^{t-1} - H_{enemy}^{t}$" represent the totally damage that AI received in one second. In the other hand, "$H_{player}^{t-1} - H_{player}^{t}$" represent the total damage that player received in the specific second. If player receive damage, difficulty level will be decrease. If AI receive damage, difficulty level will be increase.

4.6 Summary

In this chapter, difficulty scaling has been discussing at the start point. Difficulty level will affect how player challenge the game. Section 4.3 has review three studies have been done about difficulty adjustment. Section 4.4 present what DDA is and what the key elements of DDA. Three different DDA has been given for the comparison which they are Hamlet system, dynamic challenging level adapter and FUA based system. At the later section, a new difficulty measurement has been introduced which base on the key element "HP" in difficulty measurement.

5 Game Design

5.1 The demo artefact

In this section, the researcher presents a detailed description of the demo action game and its two main characters, robot 1 and 2.

5.1.1 Platform

The game artefact is created by unreal engine 4 editor. Although Unreal engine 4 editor can package the game into either windows platform or android, whereas windows platform is chosen. The artefact can be run on either 64 bit or 32 bit windows 8 and 7. Windows XP may work but it is not recommended.

5.1.2 Character behaviours design

As an action game, simulating human action is an importance requirement that is complex and ambitious. In this thesis, researcher have designed a few action that are usually use in fighting game.

5.1.2.1 Movement

Character movement is an action that changes the character's location. The character in the game can move left, right, front, back, dash and jump into the air. Player can use WASD keys to control the character. By moving mouse up or down, it can change the look up view or look down. By moving mouse left or right can turn the view camera left or right. Dash is an action that character can move faster than run. Player can also press 'R' to run.

5.1.2.2 Attack

Attack is an action that the character can deal damage to another object. Basically, an attack is divided into combo sword attack and long range gun shot.

5.1.2.2.1 Combo attack

Combo attack (or attack chain) is a chain action that character can use in a certain order. In this demo, main character can swing sword twice by clicking mouse left bottom twice. When the character swings the sword, character cannot do any other movement such as forward and jump at the same time. Player can perform an attack by using mouse left key

5.1.2.2.2 Magic attack

The magic attack is a long range attack dealt by the player. The playable character can cast a fire ball. Once the fie ball appear, fire ball will move very fast from the front of character and disappear after a few second. If the fire ball hits the shield, it can either be non-lethal which mean zero damage or it can be lethal which rated to 15 damages. The player can perform this form of attack by pressing the 'R' key.

5.1.2.3 Block

Block is an action that character can use to block enemy attacks. In this demo, character use her shield to block gun attack and sword attack. AI enemy[yz52] always face to player when AI enemy[yz53] is using his shield to block attack from player.

5.1.3 Table of set up state

This is the table of game data setup. Each sword attack will deal 20 HP and magic attack will reduce enemy HP by 15. Both AI[yz54] enemy and player movement speed is 150 and 300. Maximum HP is 1000

Sword swing 1 damage	20
Sword swing 2 damage	20
Magic damage	15
Movement speed walk/run	150 /300
Jump height (cm)	300
Maximum HP	1000

Figure 9 character state

5.1.4 Level Environment

In order to make the game simpler, the environment of each level is designed into a flat ground without any obstructing objects. However, there are four wall on each side that prevents both the AI enemy[yz55] and player from exceeding the boundaries or falling over the edge.

5.2 Game structure

5.2.1 General game navigation

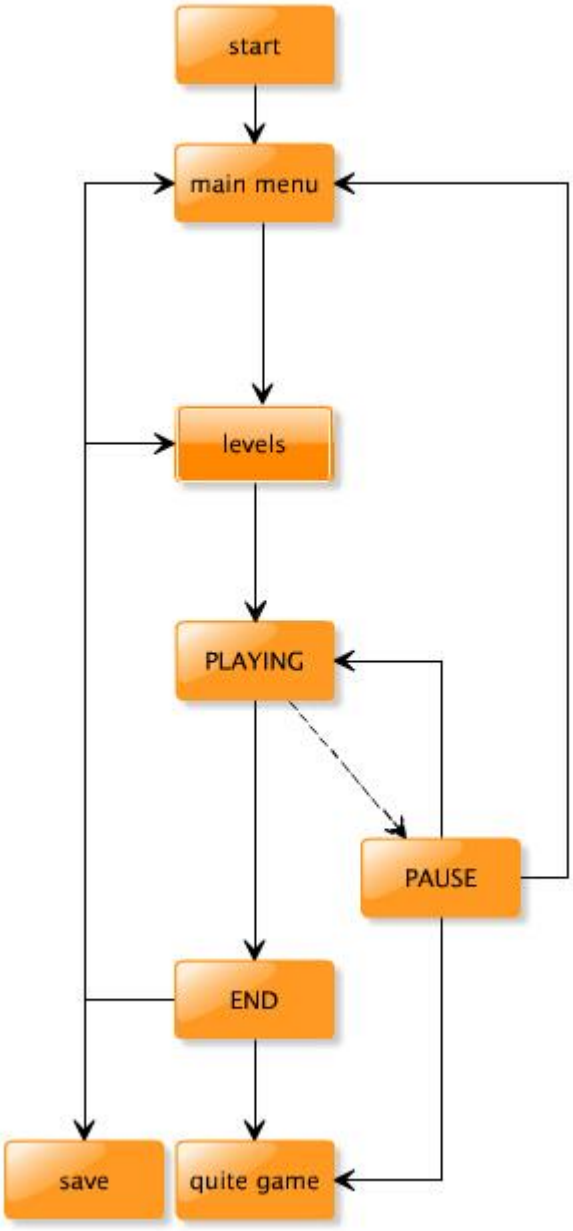

Figure 10 navigation map

Figure 10 navigation mapshows the navigation of the artefact. Player will start at the start main menu. There are four different levels which apply four different DDA system. Players can choose one which they want to play with. After the player choses the level, the game

starts to play. While playing the game, the player can pause whenever they want, the game will show a pause menu. In the pause menu, the player can go back to the main menu or quite the game. Player can also resume the game as well. When the game is finished, the game will show the ending screen. The ending screen will display the score of the player and save the players progress. During the saving process, the system will ask player to input the name before saving into the system. If the player does not want to save the game or is not happy with the result, the player can go back to main menu or restart the current level he chose. If the player does not want to play anymore, he can just quite the game by clicking the quite button on the screen.

5.2.2 Unreal engine 4 structure

In order to install all the DDA system into the game, it is important to know about the Unreal Engine 4 structure. This subsection is going to introduce how each blue print communicates within the unreal engine 4.

5.2.2.1 The Blue Print

Before introducing Unreal Engine structure, it is essential to know about what blue print is. The Blue print in the Unreal Engine is a visual scripting system. Blue print is based on the concept of using a node based interface to create gameplay element from within Unreal Editor. This system is extremely flexible and powerful as it provides the ability for designers to use virtually the full range of concepts and tools generally only available to programmers. [yz56]

5.2.2.2 Blue Print communication and build-in AI system

Figure 10 shows the communication with each blue print within the Unreal Engine 4.

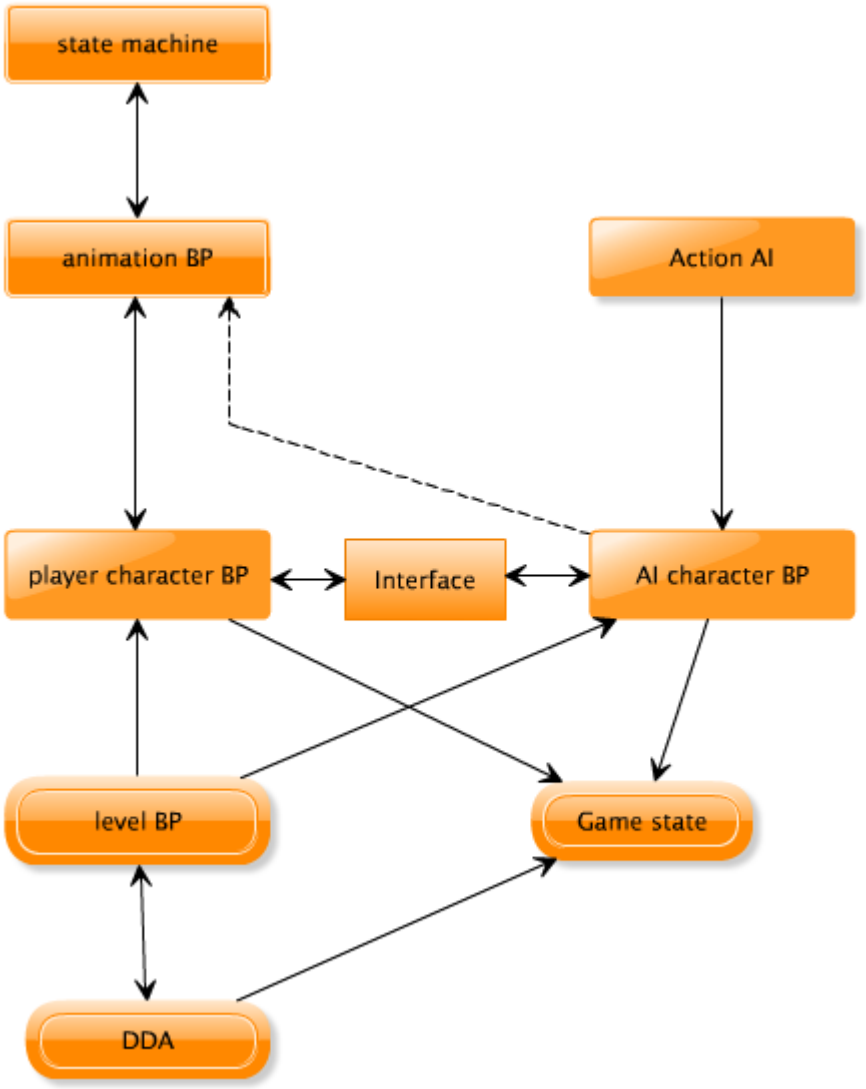

Figure 11 Blue Print Communication

State machine is one of the build-in system in Unreal Engine 4. State machine is used for the controlled character. By setting the condition and animation file, state machine can set up the state of the control character. In this game artefact, state machine can create all states[yz57] that the character will happen in the game such as jump, run, stand and walk. They are just animation setup. The player [yz58]character blue print is the code of the action like jump, attack or receiving damage. Interface is used to display the HP bar for both player and the AI. Action AI will be installed within AI character blue print. When AI character is running in the level, the game engine will run the action AI as well. Game state is used to display game state such as AI state and Player state. In this artefact, game state will be used to save game state data and output the related data into a text file. After the game state generate text file, researcher can use this data to analyse what happened during the gameplay.

Level blue print are usually setup a level coding. Level blue print can easily get information and event from both player character blue print and AI blue print such as attach value, damage event or any other event. Therefore, the research will install each DDA algorithm into the level blue print for each level in this artefact. The more detail about how each DDA algorithm apply into the game will be described in the next section.

5.3 Game blueprint in artefact.

5.3.1 Learning from Markov Decision Processes Blueprint

This subsection will show how the researcher apply an improved Markov Decision Processes (MDP) apply into action game. For the Markov decision processes, “Markov Decision Processes” is based on action outcomes which depend on the current state. The key function is that the algorithm checks three initialise stage before submit an action. The first initialise stage is available action. It is necessary to check whether the action is able to use but it is depending on the game engine. If the game engine can check the available action, it is not necessary to create a new function to the valid actions. The second initialise is the AI stage and the last initialise is submitting an action with given an optimal policy.

However, the problem of MDP in action game is the accuracy of the action. In some case, available action may not change AI state. In this artefact, game designer will change the stage to become character action and action become accuracy. Therefore, accuracy outcomes depend only on the current action. The base logic is if AI use action A and hit the target, action A will get reword and increase probability of action A. The new formal below will apply into the game artefact. It is the probability that accuracy in action ‘a’ at time‘t’ will lead to action a' at time t+1

$$\rho(a_{t+1} = a' | a_t = a, Accuracy_a = Accuracy)$$

Equation 2 MDP

5.3.1.1 Action and reward design

The diagram shows below is the action transition diagram. When AI standing, there are 50% that AI will get closer to player or AI will run away from player. If AI want to get close to player, AI will walk 1 sec and calculate the distance[yz59] between AI and player. If the distance [yz60]is smaller than before one sec, it means AI is successfully move closer to player and S1 will get reward. The reward function will set to be 1% and discounting factor will set to be 0.5%. The probability of S1 (P1) will be increase to 51%. If AI failed at S1, P1 will decreased by discounting factor. P1 will be 49.5%.

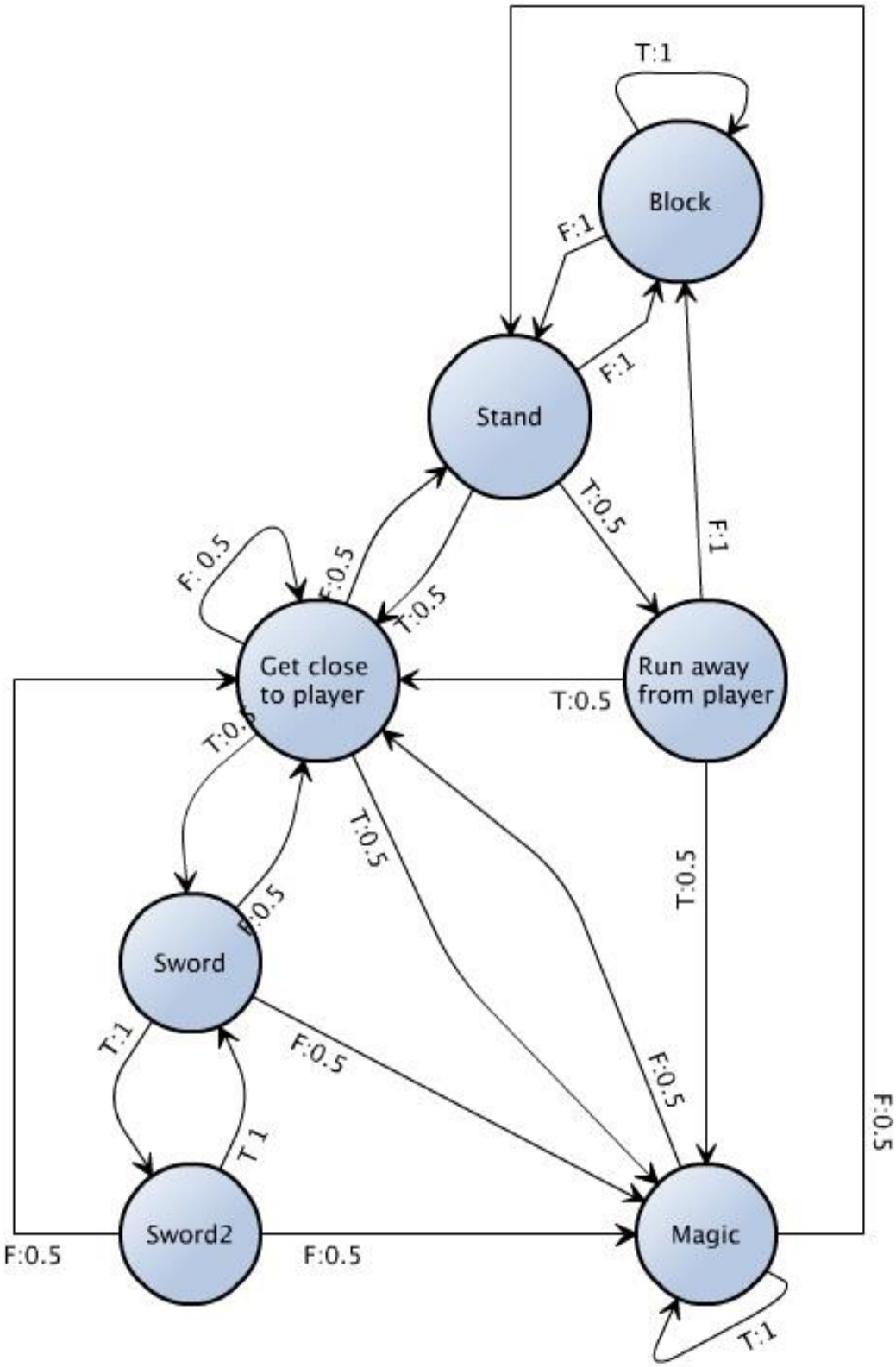

Figure 12 Improved Markov Decision Processes action overview

After AI successful get closer to player, AI will try to attack player using sword or magic. If sword hit the player successful, AI will use combo attack sword immediately. If sword 2 hit the player successfully as well, AI will use sword 1 again. If AI use sword 1 and miss the target, AI will use magic attack. If AI use sword 2 and miss the target, AI will use either magic or stand. After using magic, AI will use magic again or just stand. Table 1 shows more detail about each relation between action and accuracy.

SOURCE ACTION	**Source Action Success or Fail (T/F)**	**POSIABLIITY (%)**	**Next ACTION**
Stand	(F) Fail to stand 1 sec	100	Block
Stand	(T)Stand 1 sec	50	Get closer
Stand	(T)Stand 1 sec	50	Run away
Stand	(F)Block no damage within 2 seconds	100	Stand
Block	(T)Block some damage within 2 second	100	Block
Get closer	(T)Distance'<Distance	50	Sword
Get closer	(T)Distance'<Distance	50	Magic
Get closer	(F)Distance'>Distance	50	Get closer
Get closer	(F)Distance'>Distance	50	Stand
Run away	(T)Distance'>Distance	50	Magic
Run away	(T)Distance'>Distance	50	Get closer
Run away	(F)Distance'<Distance	100	Block
Sword	(F)attack miss	50	Get closer
Sword	(F)Attack miss	50	magic
Sword	(T)Attack hit	100	Sword 2
Sword 2	(F)Attack miss	50	Get closer
Sword 2	(F)Attack miss	50	Magic
Sword 2	(T)Attack hit	100	Sword
Magic	(F)Magic Miss	50	Stand
Magic	(F)Magic Miss	50	Get closer
Magic	(T)Magic hit	100	Magic

Table 1 action transition table

5.3.1.2 AI Character Action

This subsection will show all the available action blue print that will be perform during the battle and generally explain how each action blue print work.

5.3.1.2.1 *Flinching (damage reaction)*

Figure 13 Damage Reaction Blue Print. When character received damage from his opponent, the game engine will call event any damage. Once the event any damage called, it will decrease AI current HP by damage value and add player attack successful times. If AI HP below or equal to zero, AI character will be destroy or play flinch animation.

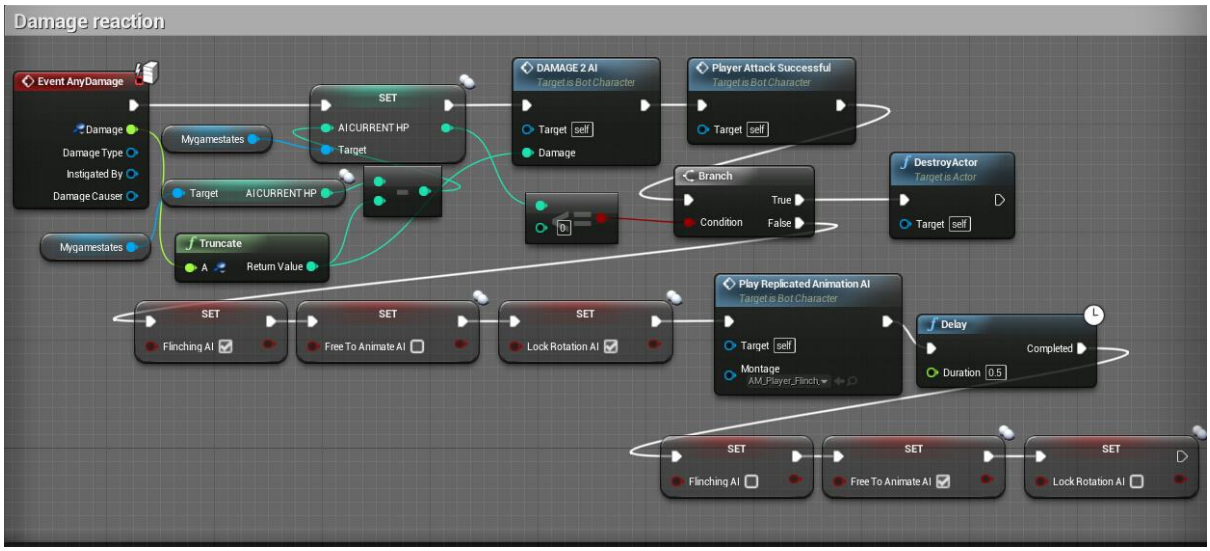

Figure 13 Damage Reaction Blue Print

5.3.1.2.2 *Base Movement*

Figure 14 walking blue print 1and Figure 15 walking blue print 2is the movement event. If AI speed is faster than 150, variable 'Walking AI' will set to 'true' and 'running AI' will set to 'false'.

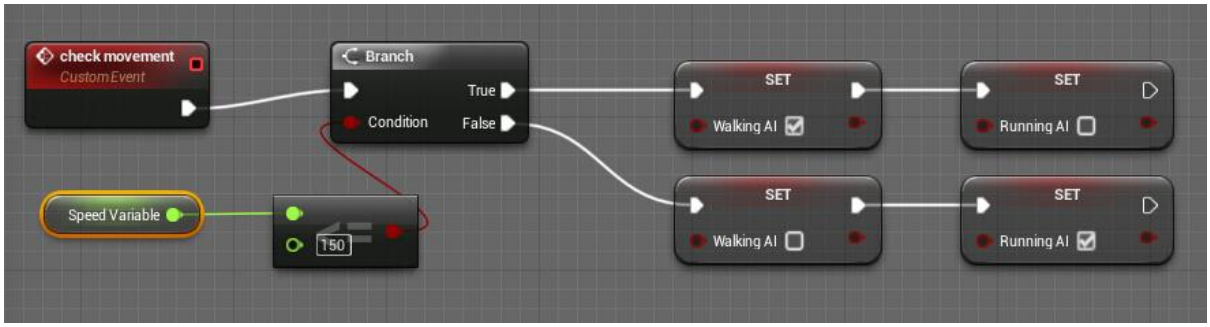

Figure 14 walking blue print 1

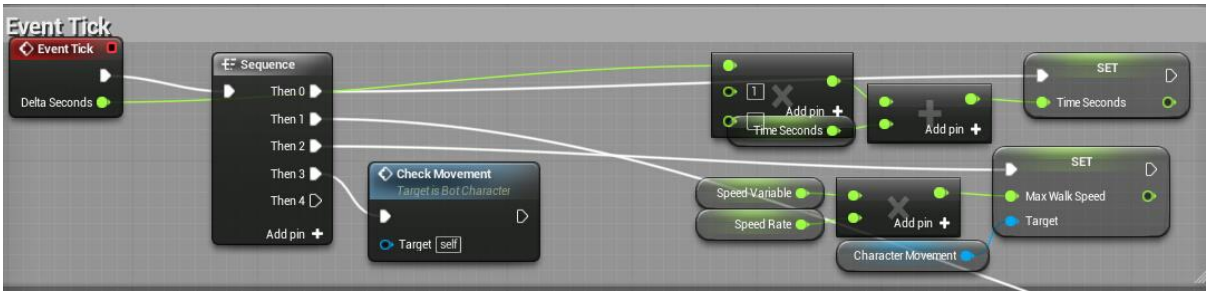

Figure 15 walking blue print 2

5.3.1.2.3 ***Combo Attack***

Figure 16 Bot attack1 event part 1, Figure 17 Bot attack1 event part 2, Figure 18 Bot attack2 event part 1 and Figure 19 Bot attack2 event part 2is the combo attack event of AI. The variable 'During S1 AI', 'During E1 AI', 'During S2 AI', 'During E2 AI' represent the attack stage 'swing 1 start', 'swing 1end' 'swing 2 start' and 'swing 2 end'. When AI start to attack, 'bot_attack1' will be called and set 'During S1 AI' to true. If AI deal damage to player, 'bot_attack2' will be called. If AI miss the attack, AI will play swing end animation. Value 5 is the probability of next action. If the random number is below the value 5, AI will move to player location. If the random number is higher than value 5, AI will use magic fire ball. The whole action flow is decided by the action transition from last section. Each attack will add sword attack times for DDA calculation. After AI finished sword attack action and decided which next action is, variable 'sword1' will set to false. Otherwise, the tick event will call this action again. Variable 'Free to animate AI' is used for check whether AI allow to move. When this variable is set to false, AI cannot move during the attack or using magic.

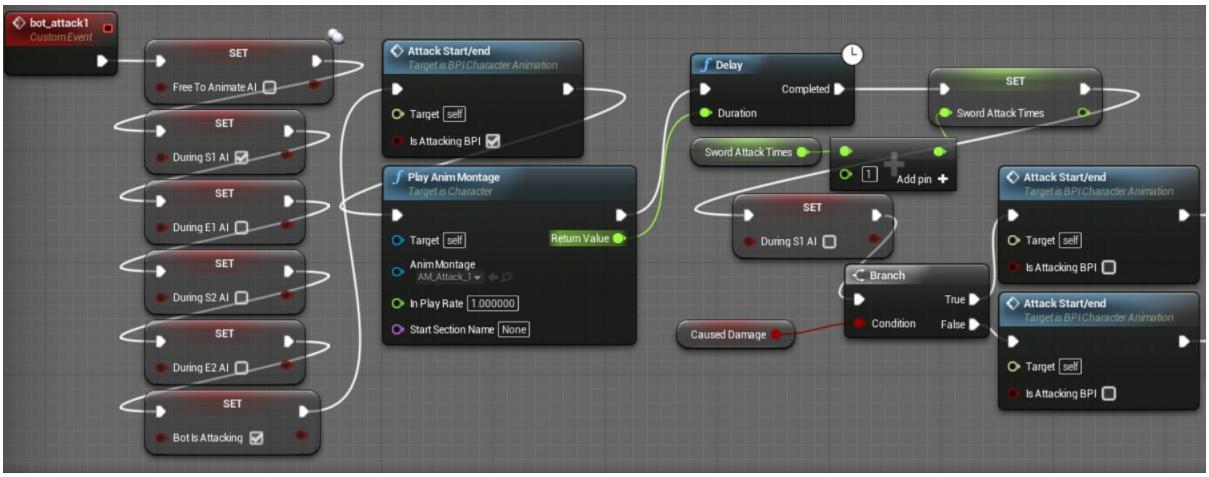

Figure 16 Bot attack1 event part 1

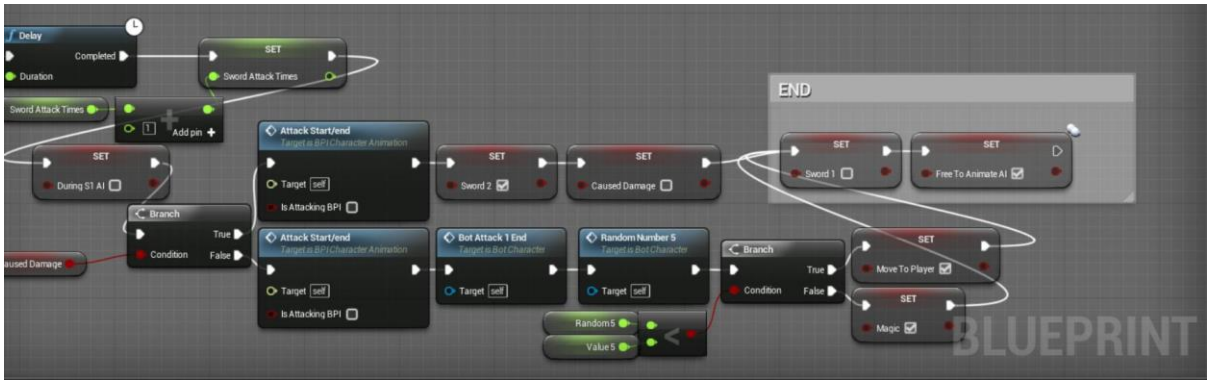

Figure 17 Bot attack1 event part 2

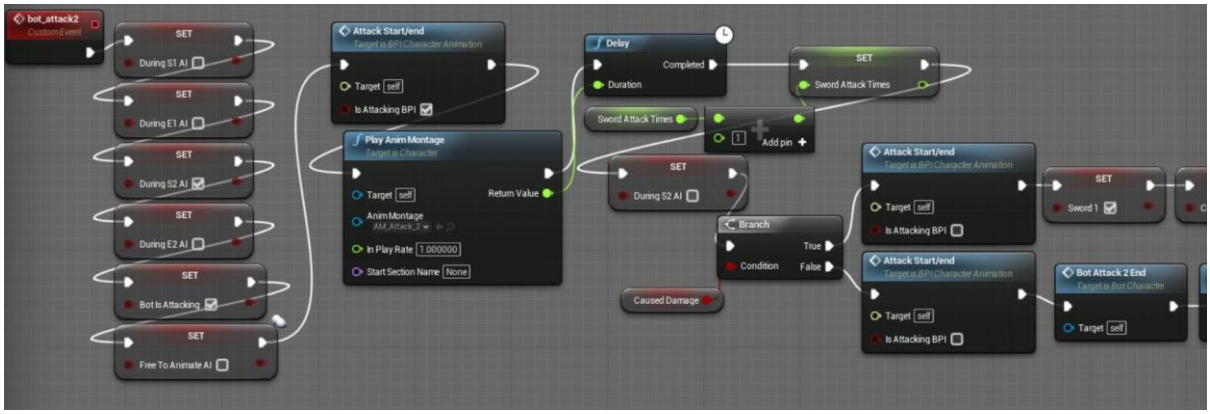

Figure 18 Bot attack2 event part 1

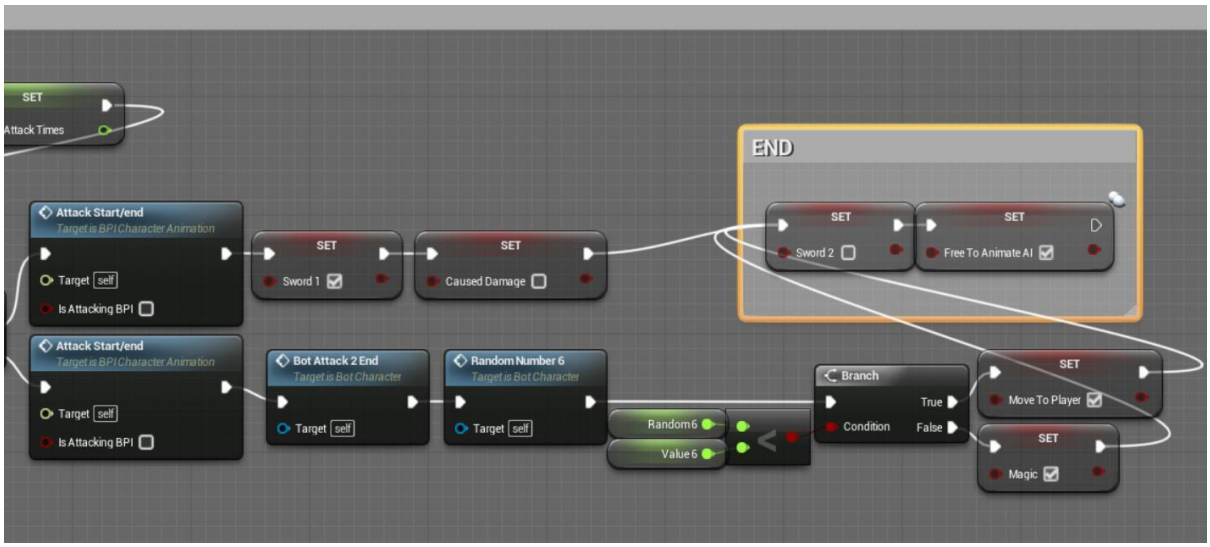

Figure 19 Bot attack2 event part 2

5.3.1.2.4 ***Using magic fire ball***

Figure 20 AI magic action Blue Print part 1, Figure 21 AI magic action Blue Print part 2 and Figure 22 AI magic action Blue Print part 3is the AI magic attack blue print. When AI try to use magic, AI will rotate his character and face to player character. When ‘magic attack’ event is called, the collision of character itself need to ignore fire ball collision. Otherwise, the fire ball will deal damage to AI itself. Other nodes are similar with sword attack. After AI finish using magic, the AI capsule collision will set to default value. If player receive damage from fire ball, AI will use magic again. If player did not receive damage from the fire ball, AI will go to ‘stand’ action.

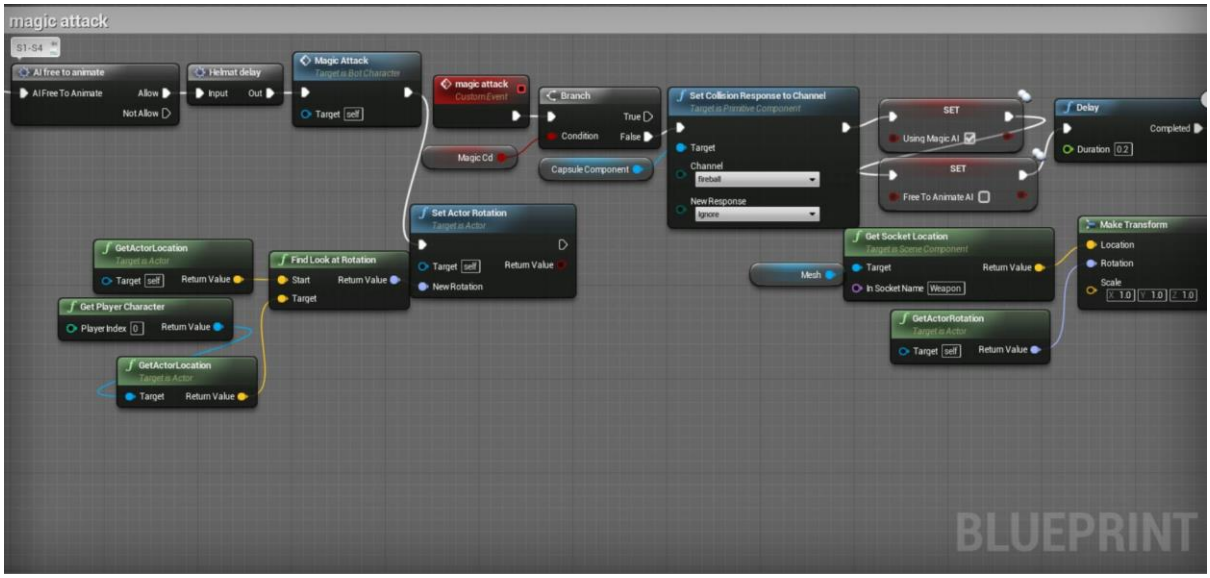

Figure 20 AI magic action Blue Print part 1

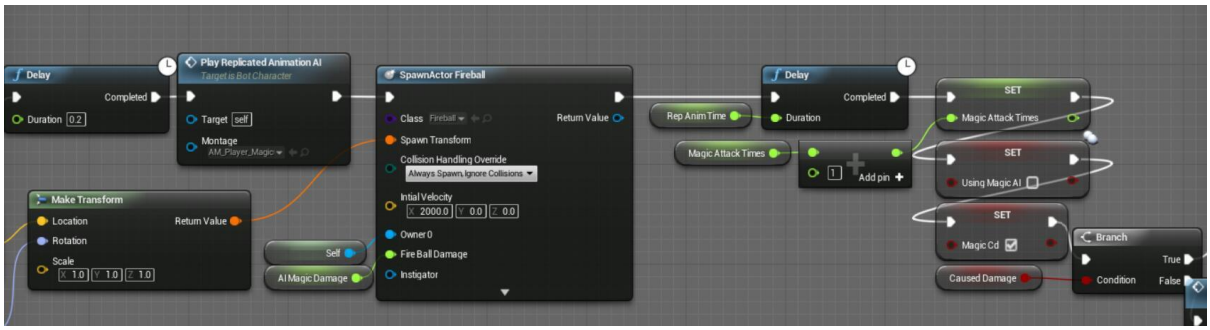

Figure 21 AI magic action Blue Print part 2

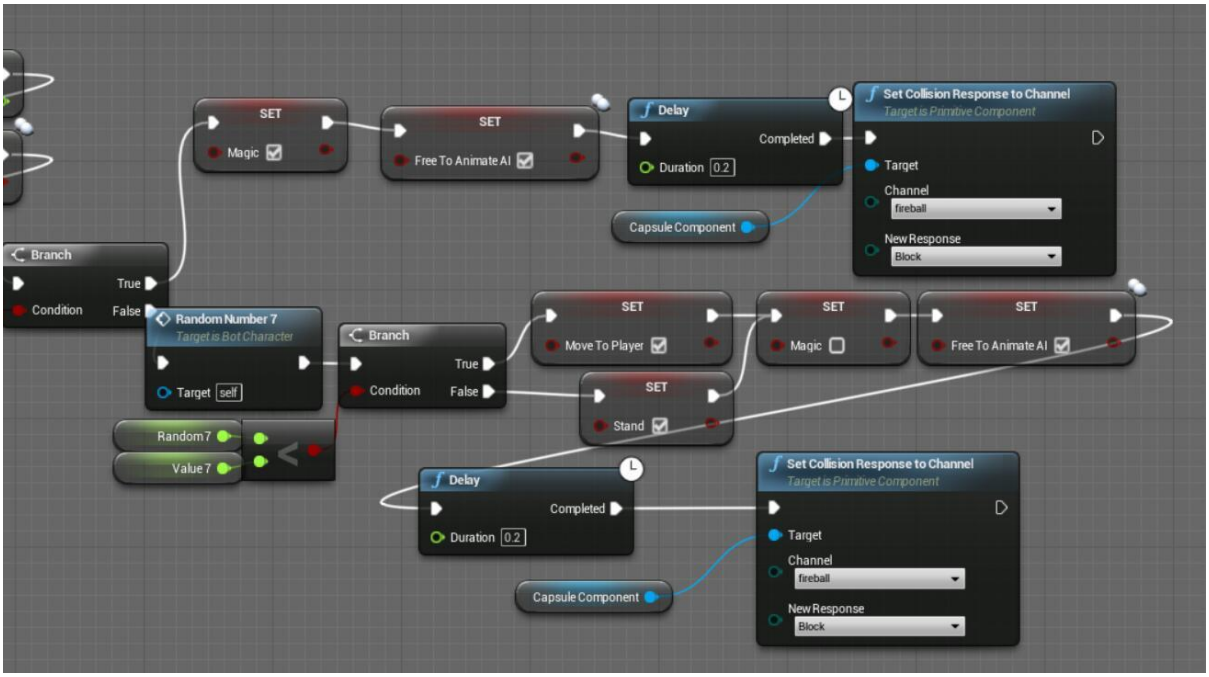

Figure 22 AI magic action Blue Print part 3

5.3.2 Hamlet support system design and blueprint

This section is showing how the Hamlet system support player behaviour within the artefact. Hamlet system is divided into two level. When player's HP below 50% MAX HP, system will start to delay AI action between each action decision. The delay time will start from delay 0.5 sec and will be increase to 2 second until player HP below 20%. When player HP below 20%, Hamlet system is going to respawn some items that will affect the AI state and player state during the game. There five items that help player if player get in dangerous.

- Health cube: add 15 HP to the character who pick up.
- Iron cube: decrease enemy sword attack damage value and magic damage value.
- Yellow cube: slow enemy speed.
- Red cube: Increase sword attack damage by 10.
- Blue cube: Increase magic damage by 20

In the other hand, if AI HP bellows 20%, these five items will appear on the map as well and AI will run to the item location rather than run away from player. Each of cube will randomly respawn near player location within 500cm radius. If cube is spawned on the map, AI action 'run away' will run to the cube location instead of run to random point. AI[yz61] blue print are[yz62] showing below.

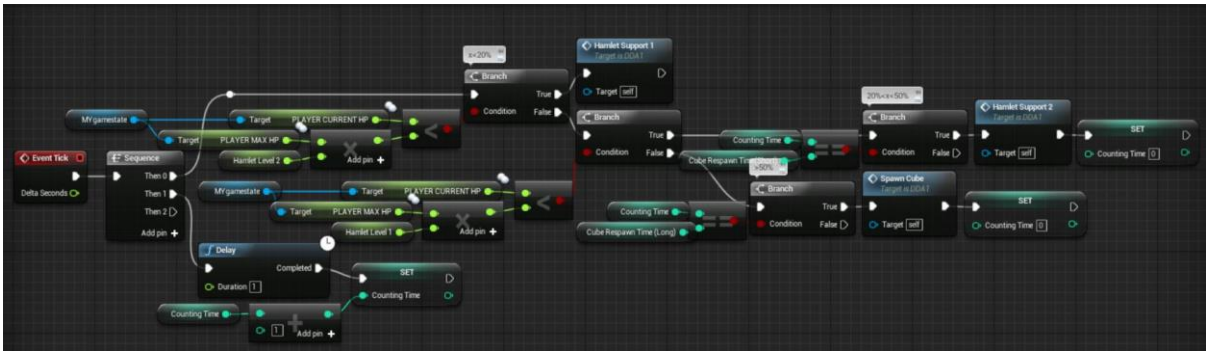

Figure 23 Hamlet system event tick

Figure 23 is the tick event and it will be called every frame. When player HP below 50%, the engine will call Hamlet support one. When player HP below 20%, the engine will call Hamlet support two.

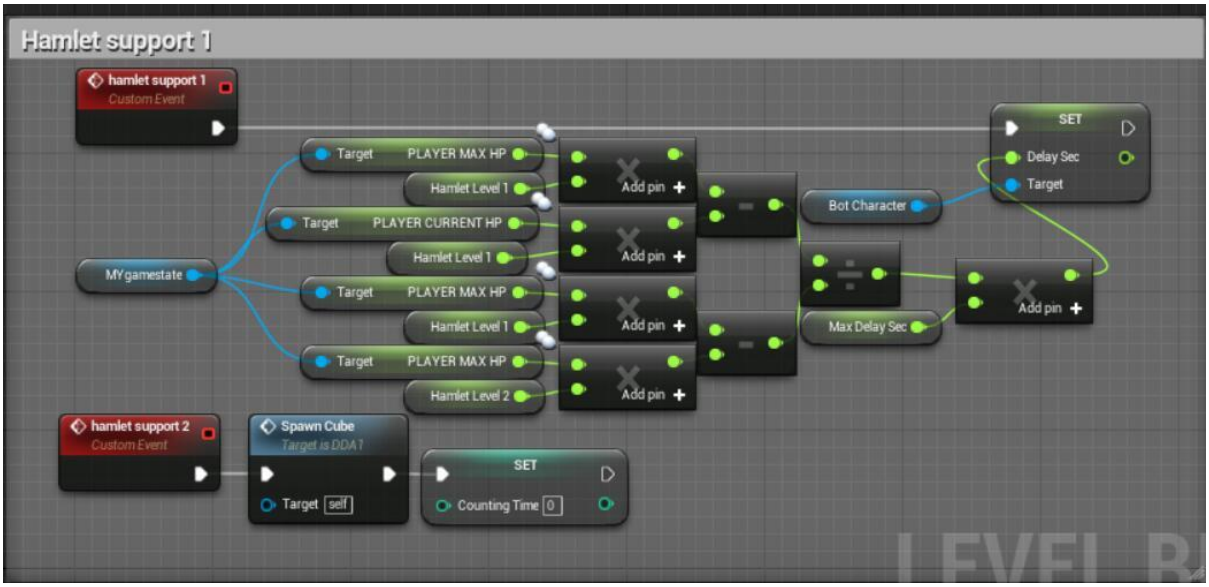

Figure 24 Hamlet system support 1 and 2

Figure 24 Hamlet system support 1 and 2shows how to scale the action delay based on the Player HP. When Player HP reach to Hamlet level 1 (50%), the delay second will be stated to increase and the maximum delay time is set to two second. When the game engine call hamlet supports 2, it will call spawn cube event.

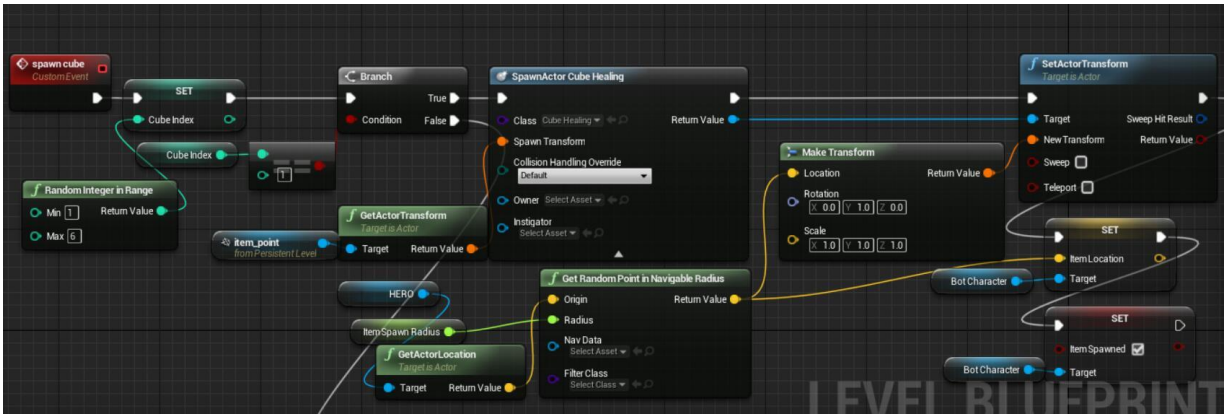

Figure 25 Hamlet system spawn cube

Figure 25 Hamlet system spawn cubehow to spawn cube on the map. First of all, the game engine will generate a random integer from minimum 1 to maximum 6 and will spawn a specifics cube base on the integer 'Cube Index'. The item will spawn near player character.

5.3.3 Dynamic Challenging Level Adapter Blueprint

This section is showing how the Dynamic Challenging level adapter works within the artefact. The base idea of the challenging level adapter is using some algorithm to calculate the Power of AI and Player. In order to decrease the power difference between AI and Player, algorithm need to change some AI state data during the gameplay. The challenging rate formula is given below.

$$\mathrm{CR} = \frac{A}{a} + \frac{H}{h}$$

Equation 3 challenging rate formula

. If CR value is increasing, it means the game difficulty is increasing. In the CR formula, parameter A (Attack) represents the sum of AI attack power. Parameter H (Hit Point) represents the sum of Character hit point. 'a' and 'h' are two constants for normalization in the CR formula They are configured as 5 for 'a' and 10 for 'h'.[yz63] The value "5" and "10" is not a certain value. They could be change to other value but 'a' and 'h' must be above 1. If 'a' or 'h' is set between '0' to '1', the CR value will decrease which does not match to the game difficulty.

Figure 26 Dynamic Challenging Rate Event Tick. If variable 'CR AI' is higher than 'CR player'. The game engine will call 'increase AI power' event. If variable 'CR Player' is higher than 'CR AI', 'reduce AI power' will be called.

Figure 26 Dynamic Challenging Rate Event Tick

Figure 27 Challenging Rate Formula. The calculation data is receiving from the AI character blue print and player character blue print. This formula will called every second not every frame[yz64] because player can deal different damage within one second. If formula is called every frame, the output data will be more detail. [yz65]

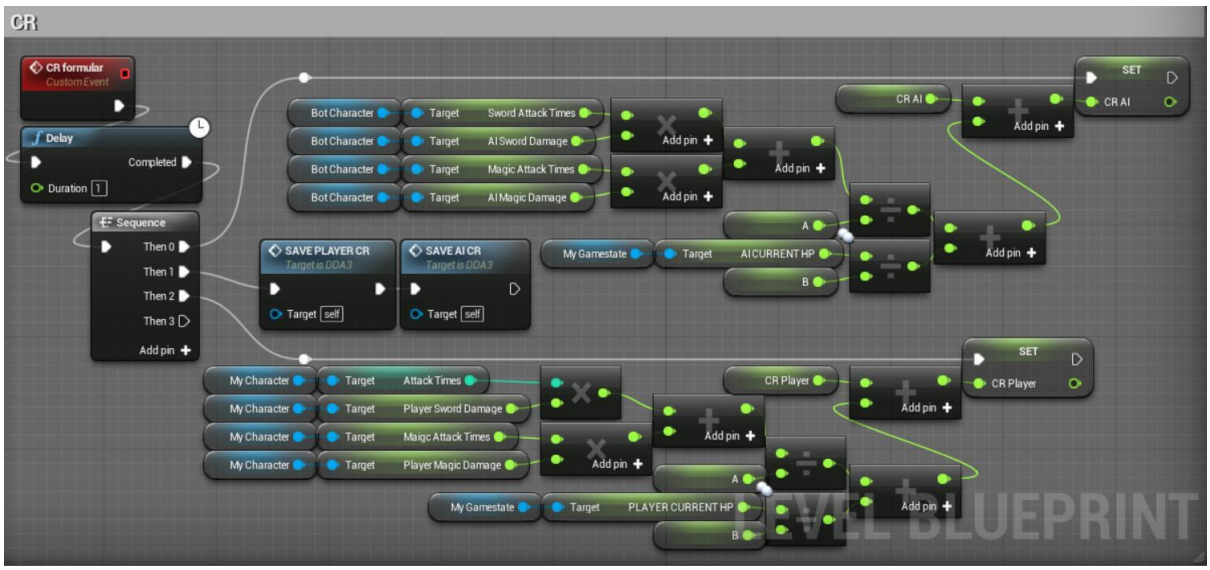

Figure 27 Challenging Rate Formula

Figure 28 Increase AI power eventis the ‘increase AI power’ event. When this event is called, value 1, value 3 and value 7 will be change. Increasing value 1 can give more chance that AI run away from player. By increasing value 3 and decreasing value 7 can give more chance to AI stand. They are the same as ‘decrease AI power’ event show as Figure 29 Reduce AI power Event.

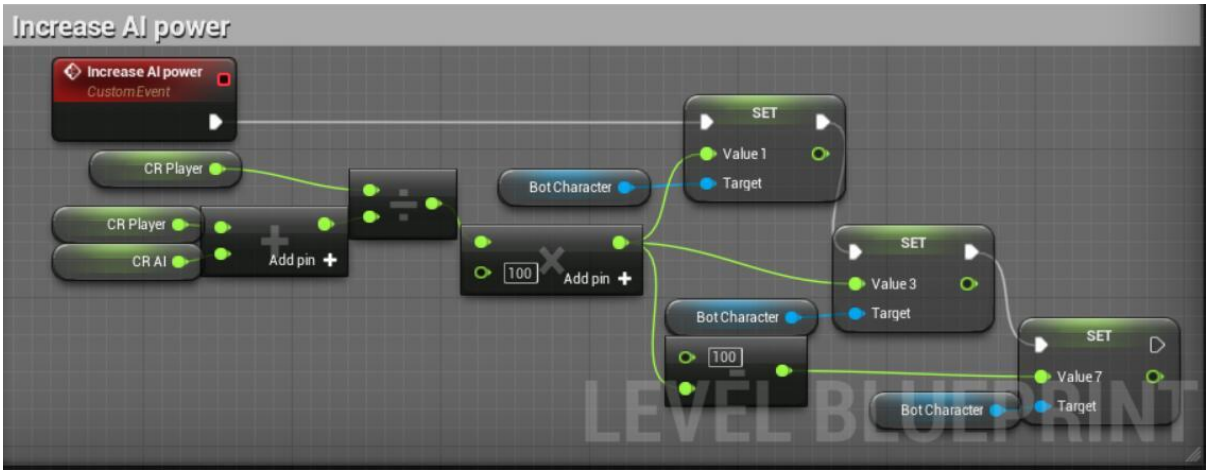

Figure 28 Increase AI power event

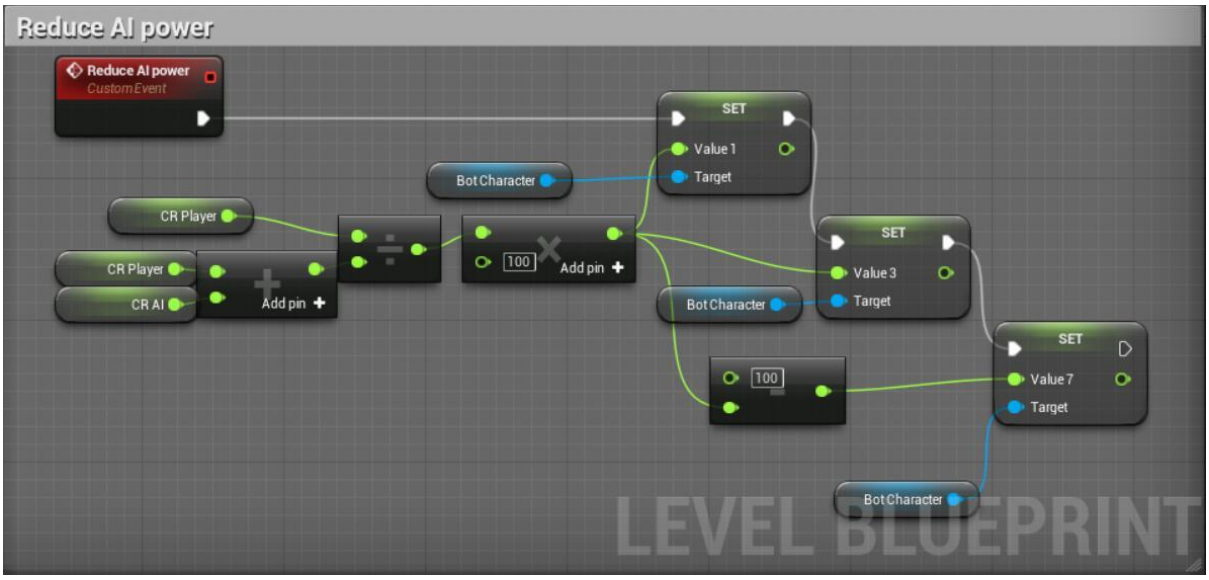

Figure 29 Reduce AI power Event

Figure 30 Output Fileis the event for file output. After the game is finish, the game will ask player to save the recorded CR into a text file. This event will be called when player click the save button at the end of the game.

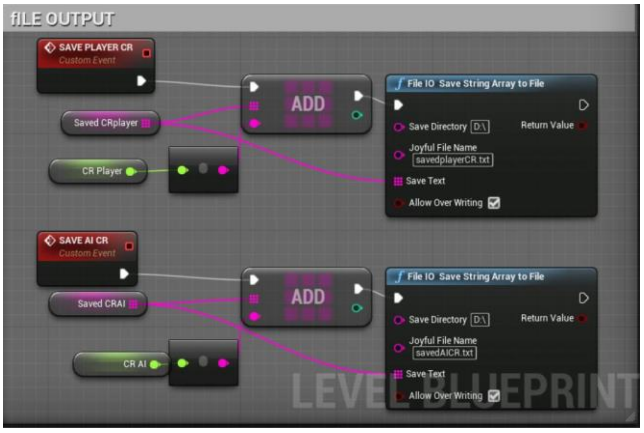

Figure 30 Output File

5.3.4 FDC and FUA Blueprint

According to Sang-Won et al (Sang-Won , et al., 2007), Factor of User's Adaptation (FUA) in action game is defined into the variation of score and enemies and the Pattern of controlling difficulty (PCD) is the order of actions. However, the action decision is decided by MDP. Therefore, FDC cannot directly apply into the game. If FDC applied into the game, the result will not apply to the aim of this thesis. In order to apply FDC with MDP, game designer decided to change pattern in action game. The aim of the action order is let AI use the action which game designer designed. So if FDC can modify the probability of each action, this would make the same effect in order to apply the order of actions. For example, if FDC is given action order [1-2-3-4-5], the new method will increase the action probability by given order [1-2-3-4-5], instead of using the original method. After increase action 1, new FDC will increase the action 2 probability and restore default probability value to action 1 probability.

The table below have signed each action with a unique number to become FDC.

Action id	Action name
1	Idle
2	Move away from player
3	Move closer to player
4	Sword attack
5	Magic attack

Table 2 Action ID

Fitness value is required because fitness value is part of FUA. The fitness value is given below:

$$Fitness = \frac{HP_{player}}{HP_{player} + HP_{AI}}$$

Fitness calculation is showing at Figure 31 FDA event tick.

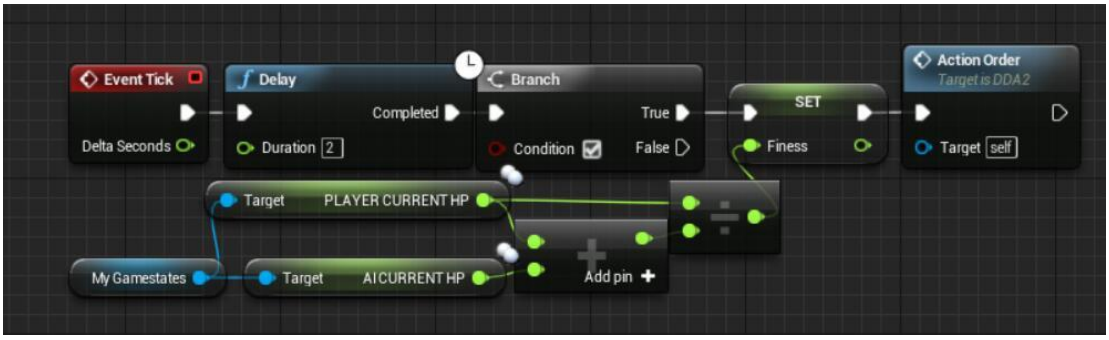

Figure 31 FDA event tick

Figure 32 FDA action orderis the action order event. During the game play, if the random float is smaller than fitness value, ‘attack style’ event will be called. If the random float is higher than fitness value, ‘defence style’ event will be called. Both event will change the probability value of each action and restore to default value after two seconds.

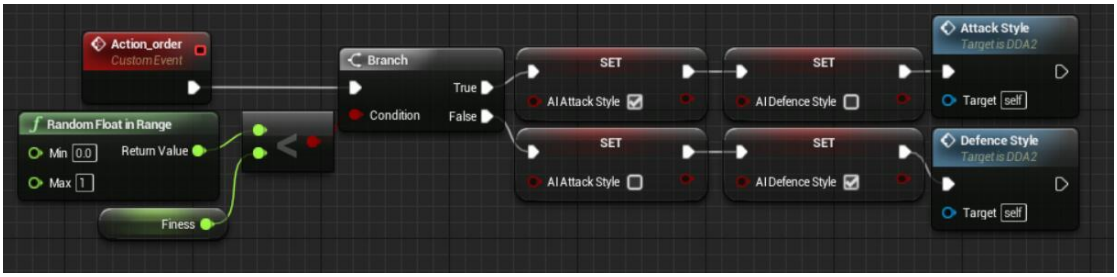

Figure 32 FDA action order

Figure 33 Attack actionis the attack action event blue print. When this event is called, a random integer will be generated to choose which attack action probability will be increased. By increasing value 1 to 100, AI will always move to player when AI is standing. By increasing ‘value3’ to 100, AI will always move to player when AI just moved to player. By increasing ‘value5’ to 100, AI will always use magic attack after AI miss the sword attack. By increasing ‘value7’ to 100, AI will always move to player when AI miss the magic attack.

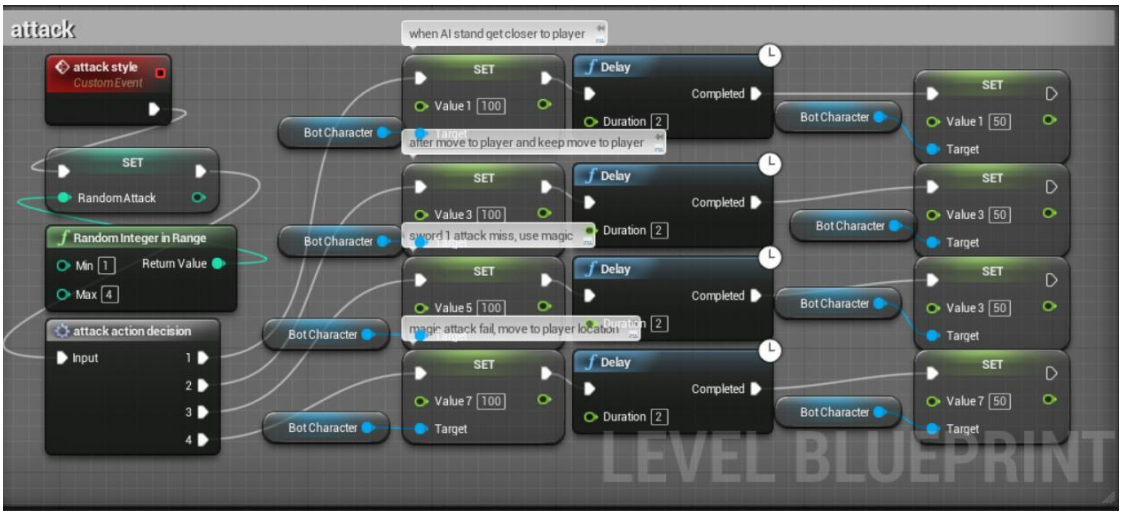

Figure 33 Attack action

Figure 33 Attack action shows the ‘defence style’ event. By setting ‘value1’ to 0, AI will always run away from player when AI is standing. By setting ‘value4’ to 0, AI will always use magic player after AI running away from player. By setting ‘value1’ to 0, AI will always run away from player when AI is standing. By setting ‘value7’ to 0, AI will always move to player location when AI miss the magic attack.

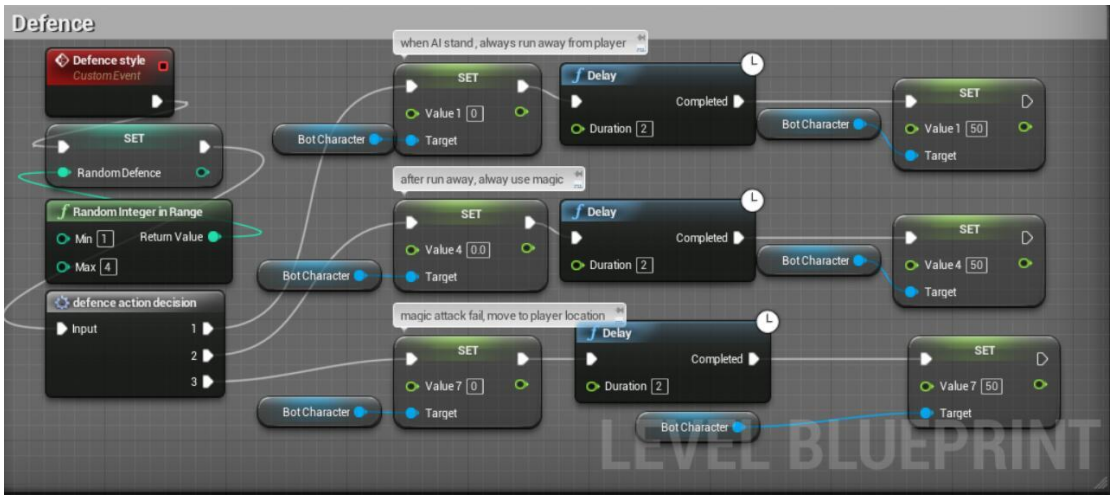

Figure 34 Defence action

5.4 Summary of the chapter

At the start of this chapter, the game structure and detailed game state is given. An action transition diagram is introduced for the game artefact. Each DDA blue print has been introduced in detail. In the next chapter, research will focus on evaluated each DDA.

6 Game evaluation and analysis

This charter is going to introduce the experiments on AI applied to one to one computer action game. Our test-bed is a computer game where the player has to defeat its enemy (i.e. AI) by using the skill it provided from last chapter. By viewing the game from the enemy perspective, researcher is going to attempt to use the same skill to [yz66]fight.

6.1 Hamlet Support System evaluation

The diagram below shows the difficulty level by using Hamlet system. At the start point of the figure, the system did not detect any action or damage because long distance between player and AI at the start spawn point. From the figure, it is evidence that player is stronger than AI at the start but AI deal more damage after a few seconds. Player can feel AI is very strong if the AI did not apply the action delay. The main reason why AI is so strong because of the flinch. When player receive sword damage from AI, player is hard to move his character away from the sword damage. In the other hand, if AI sword attack successful, AI has 100% to use sword combo attack. Player was activated Hamlet system level 1 to decrease the cube spawn time but it did not help until player activated Hamlet system level 2 to delay all the action of AI. After a few second, player was able to beat the AI and finish the game. Player can feel AI is getting weak because of the action delay. Therefore, it is necessary to reduce the maximum delay time.

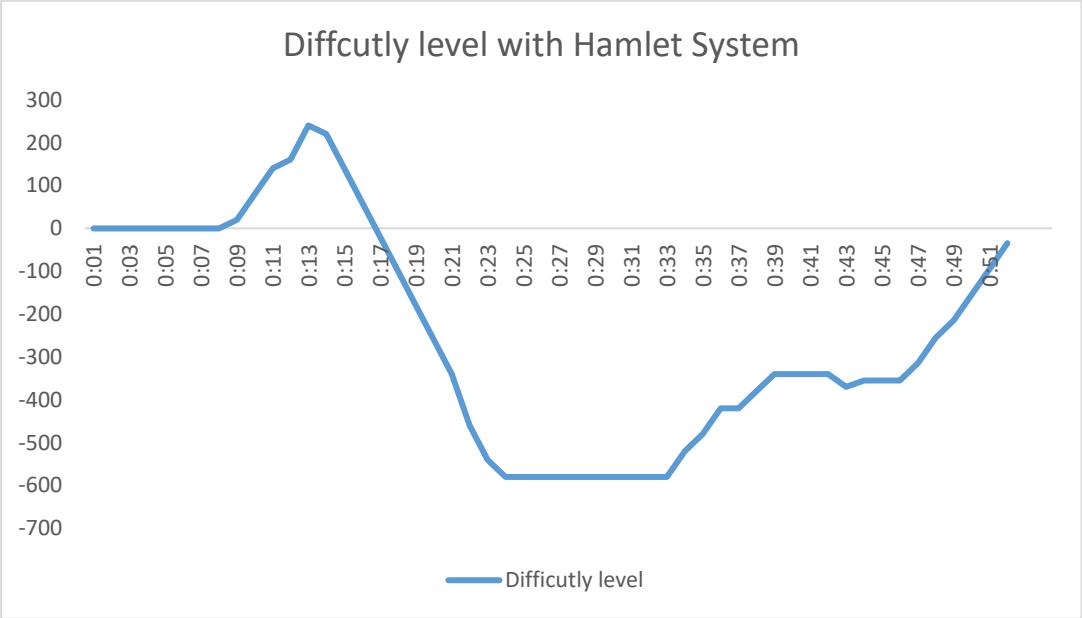

Figure 35 Hamlet system test

The X axis is the game time and Y axis is the difficulty level point. At above figure, the game start from 0 sec and finish after 51 seconds. The highest difficulty level is about 290 at 0:13. The lowest difficulty level is about 550 at 0:23. After 10 sec, difficulty level is start increasing and finish at -50. The calculation is stored at game state blue print. Each frame the game

state will calculate the current difficulty level and stored the value into an array. When the game is finish, the array will stored into a text file.

6.2 FDA and FUD evaluation

Figure 36 Dynamic Challenging Level Adaptershows the difficulty level by using FDA and FUD. From the last evaluation, the AI is very strong without any DDA control. From this figure, it proves that dynamic challenging level adapter has low effect to the difficulty. The highest difficulty level is zero and the lowest difficulty level is -500. From 35 second to the end, difficulty level is increasing slowly but player cannot win the game. Dynamic Challenging level adapter is trying to help player but the functionally of this DDA is insignificant by comparing previous DDA.

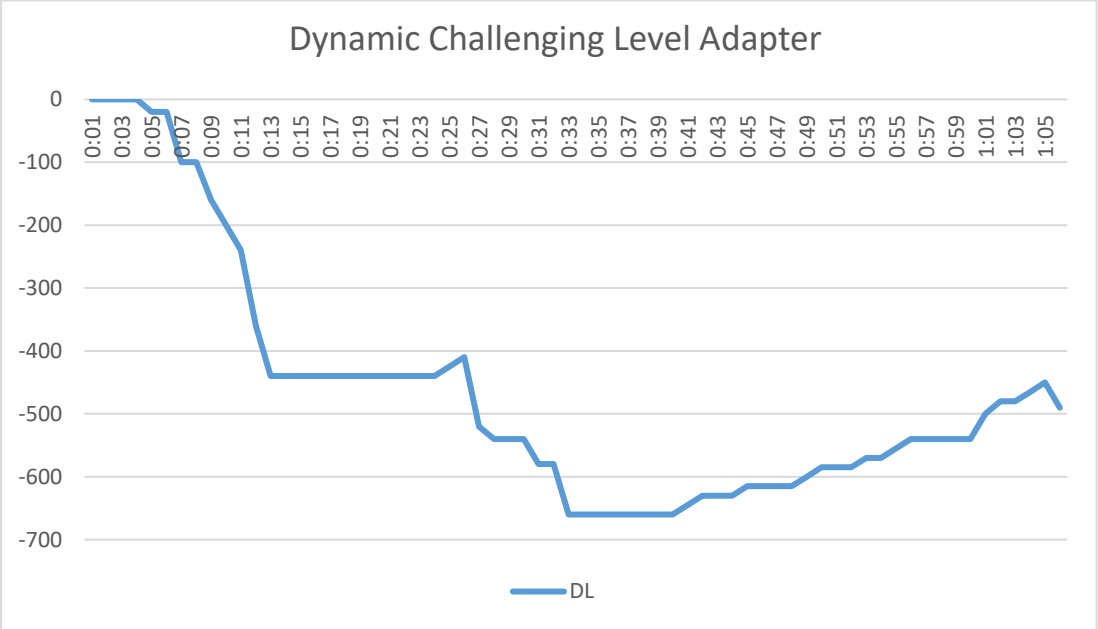

Figure 36 Dynamic Challenging Level Adapter

6.3 Dynamic Challenging Rate evaluation

Figure 37 Challenging RATE without controlshow the CR diagram without DDA control. [yz67]From 21 second to the end of the game, AI is stronger than player and the difference between AI and player is increasing.

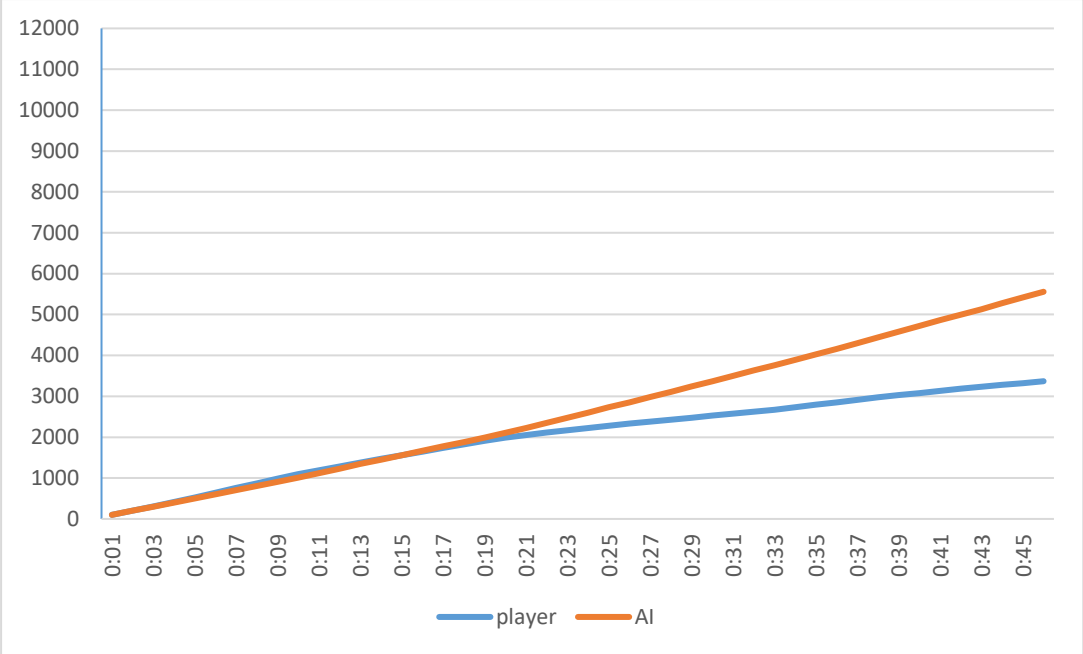

Figure 37 Challenging RATE without control

Figure 38 Challenging Rate with Controlshows the challenging rate with DDA control. It is evidence that the challenging rate at the end of the game is higher than previous figure which is no dynamic control.

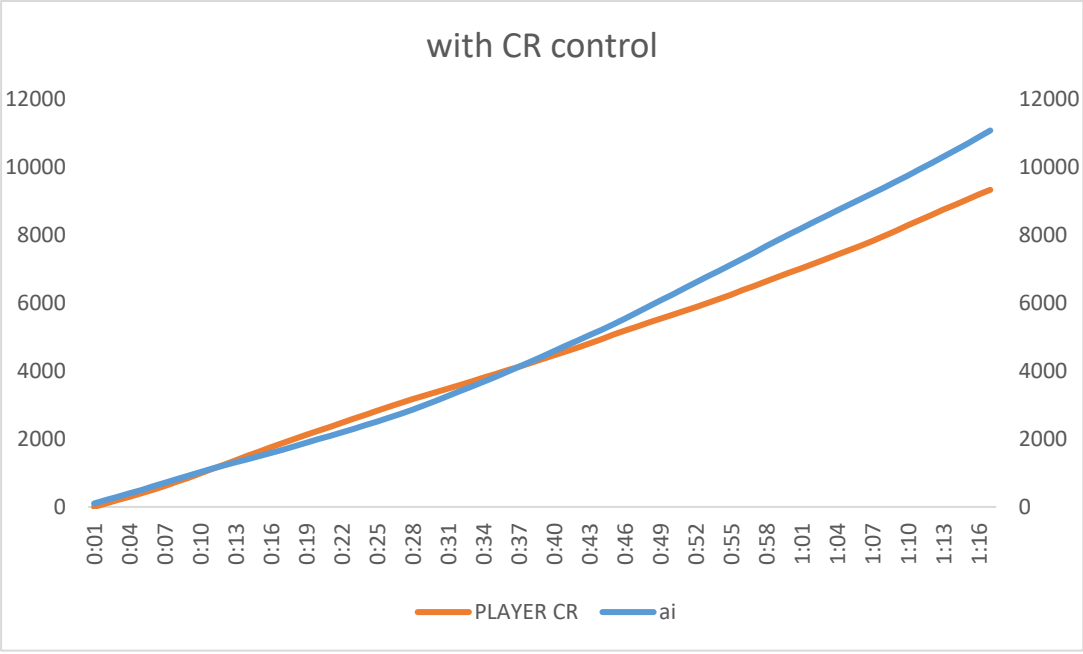

Figure 38 Challenging Rate with Control

Figure 39 Difficulty Level with CR controlprovide confirmatory evidence that the difficulty level with CR control is more effected then other DDA. Difficulty level is around between 280 and -280. The gap between player and AI is much smaller than previous result.

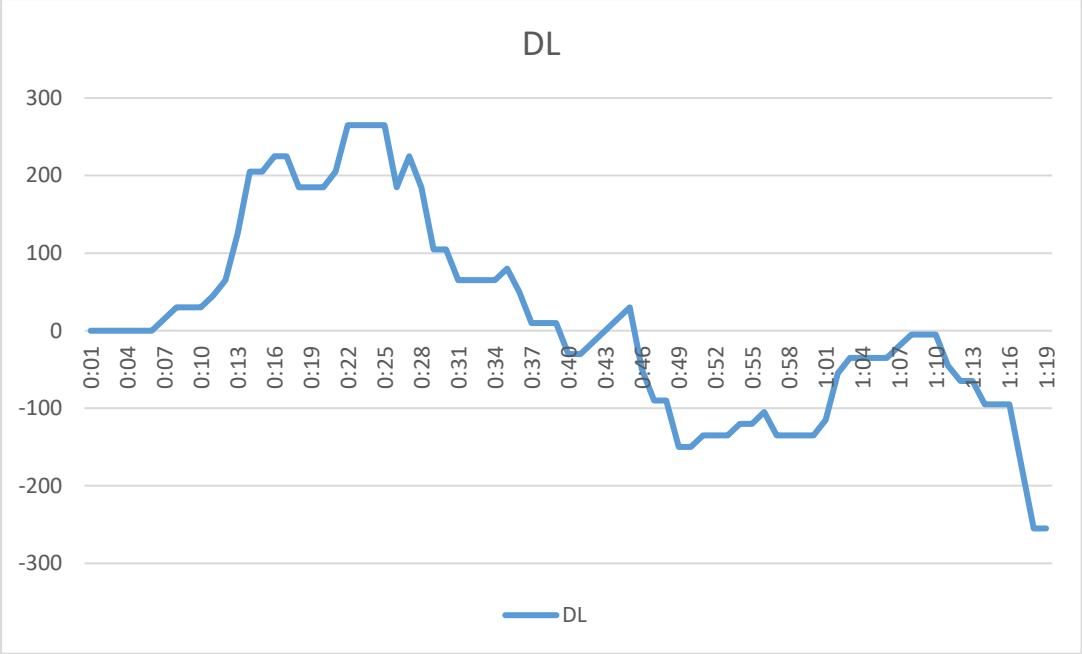

Figure 39 Difficulty Level with CR control

6.4 DDA comparisons

Figure 40 DDA comparison are showing three difficulty levels [yz68]within a diagram. It is evidence that the game time of the CR formula control is longer than other DDA. The shortest game time is Hamlet system. CR game time is about 30% longer than Hamlet system. From the ending of each difficulty level, hamlet system is close to zero which means the system has change the difficulty level equal to the player skill level. CR DDA finished about -210. However, CR DDA has increased from -100 to zero at the end part of the game and begin top drop to -200. By comparing CR to other, CR has low amplitude changes with maximum 200 and minimum -210. In the other hand, hamlet system has higher incensement effect. By using hamlet system, difficulty level can grow from nearly -600 to 0 within 20 seconds. For the FUA, FUA growth slow and steady. It started to rise the difficulty from 33 seconds to the end of the game. FUA is not suitable for a 1 minutes or 2 minutes' game play. FUA may apply into a longer game time if the game time is designed to 10 minutes or even longer. As result, the advantage of CR DDA is the stable adjustment from the beginning of the game and hamlet system is featured with high adjustment policy.

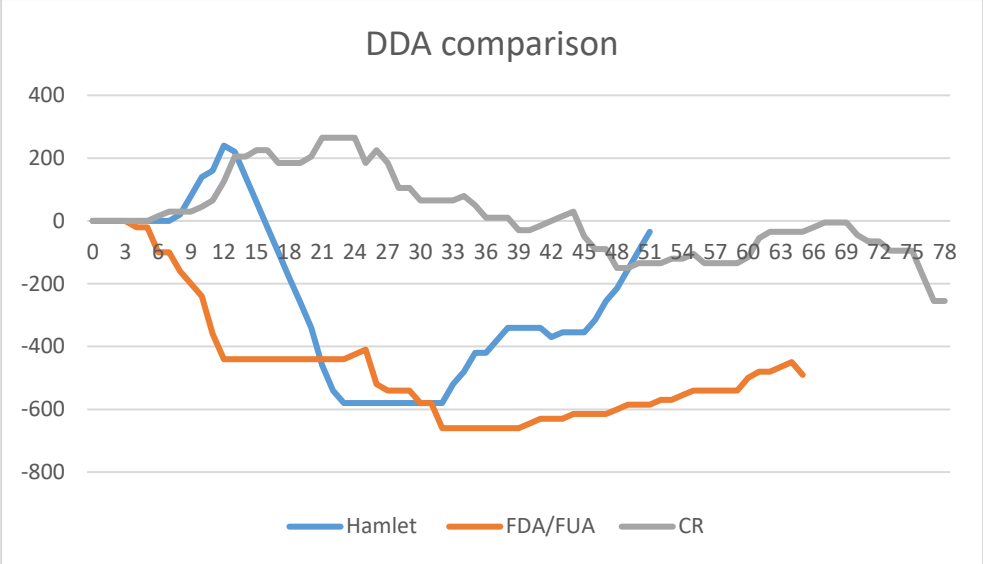

Figure 40 DDA comparison

6.5 Improved DDA

Section 6.5 present an improved DDA solution after analysis previous three DDA systems. The improved DDA will base on combine two DDA into one adjustment system. The aim of this improved DDA should be suitable for one to two minutes' game play, steady difficulty scaling and fast response to the adjustment. The improved DDA will apply finesse formula to smooth adjust the spawn cube time instead of fixed time. The maximum delay action time will reduce from 2 seconds to 1 second.

6.5.1 Improved DDA blue print.

From the Figure 41 Improved DDA event tick and Figure 42 Improved DDA event tick 2, improved DDA event tick structure is similar with hamlet system and CR based adjustment system.

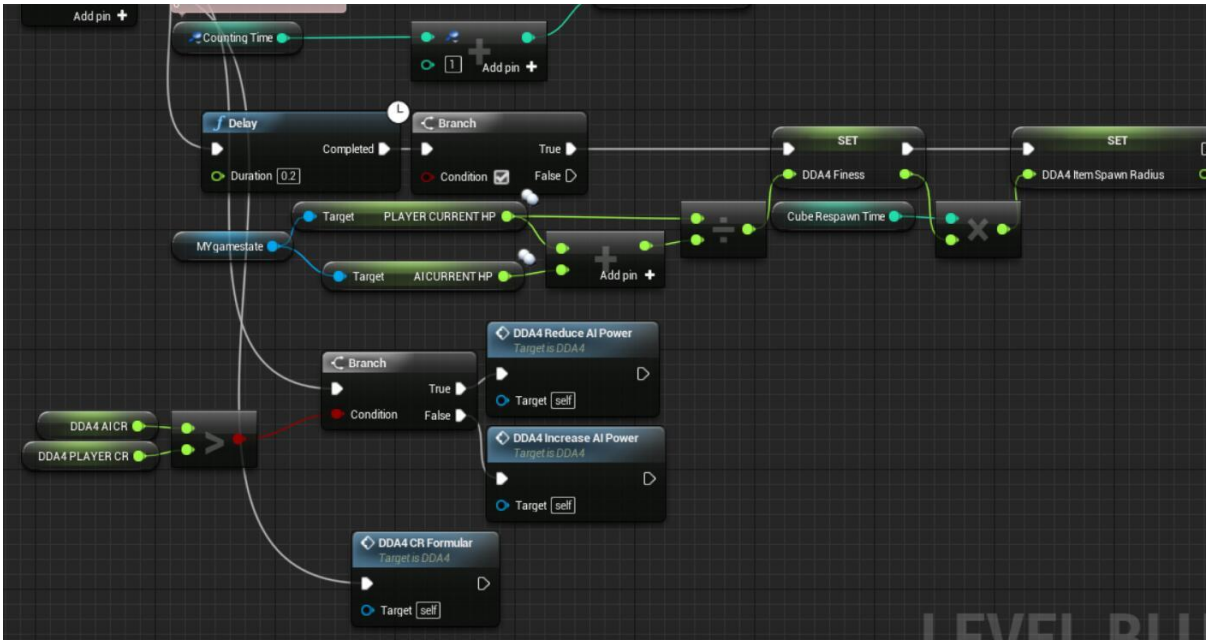

Figure 41 Improved DDA event tick

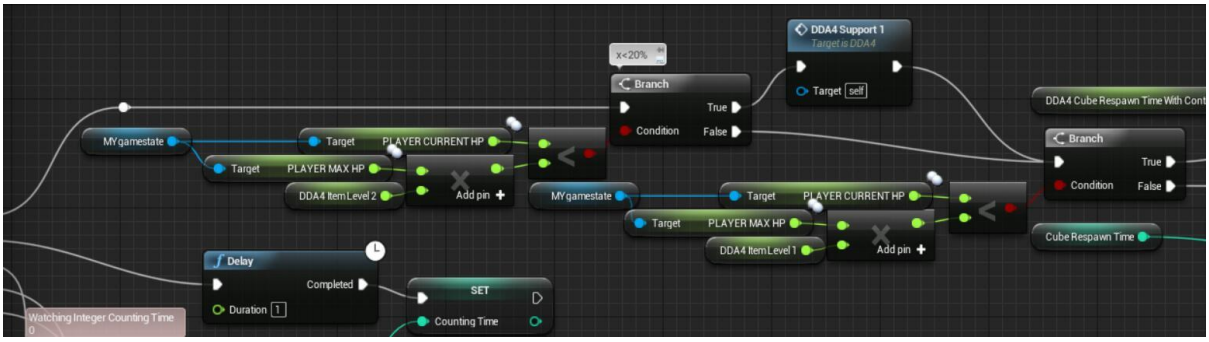

Figure 42 Improved DDA event tick 2

6.5.2 Result

Figure 43 Improved DDA comparisonpresent the improved DDA diagram by comparing with previous three DDA. At the beginning of the game play, difficulty level is grow 100 points[yz69] and start drop after that. The difficulty level is reducing smooth and steady until one minute's time. At about one-minute time, the DDA activate the support level 2 and the difficulty level is increasing sharply. At the second half of the game, the difficulty level has grown over zero point until the game finish. For the game time, the improved DDA has reached nearly two minutes which achieve the requirement of the game time.

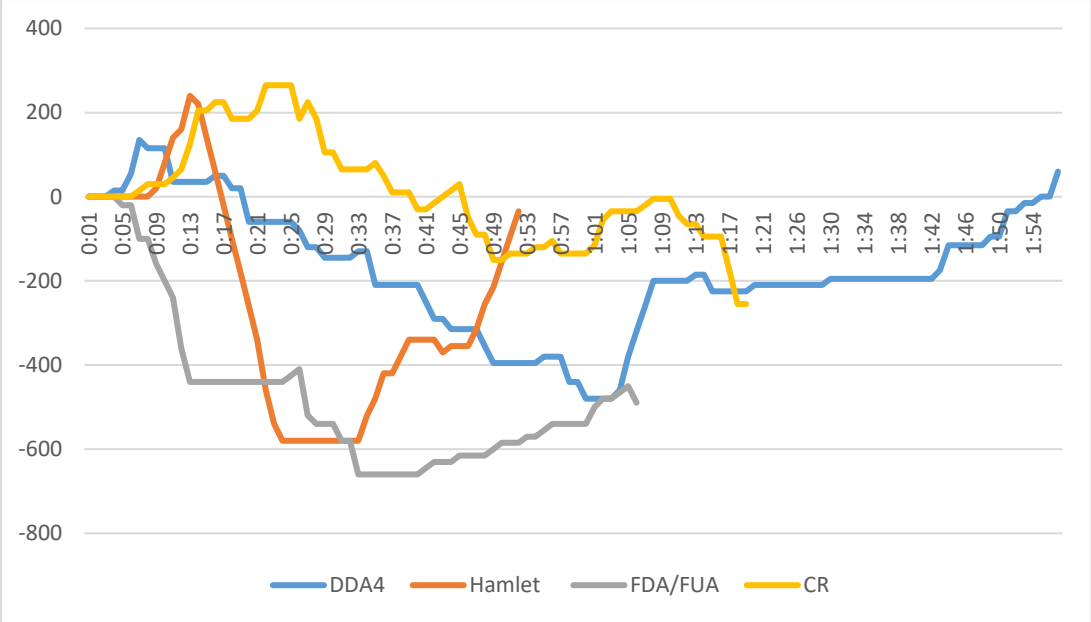

Figure 43 Improved DDA comparison

6.6 Summary

In summary, this chapter has presented[yz70] the comparison between four different DDA by using a game artefact. The advantage of CR adjustment is the low amplitude changes and steady adjustment. The advantage of hamlet system is the high effect growth when player in danger but the short game play is the main problem. FDA is not appreciated with this artefact thought FDA effected with steady growth.

7 Conclusion

In the final conclusion of this thesis, three questions which listed in chapter one have been asked[yz71]. In the second chapter, the definition of action game has been given in order to understand what action game is. Action game included five key elements which are character states, character ability, level, point of the view and the game scoring. Chapter three provides different definition of AI in different area and briefly introduce eight types of AI which included expert system ,case-based reasoning in game, finite state machine, 错误!未定义书签。decision trees, search algorithm, flocking algorithm, genetic algorithms and neural networks. Section 3.2 introduce three different kind of action game action AIs [yz72]which are included Markov decision process, k Nearest Neighbor algorithm, real-time limitation based algorithm. Their advantages and disadvantages has been discussed thought a table. The result is Markov Decision algorithm will apply to the game artefact. Chapter 4 introduced what game difficulty is and reviewed[yz73] a few dynamic difficulty adjustment in video[yz74] game. Three DDA have [yz75]been proposed. Hamlet system adjustment is reactive actions which will adjust the displaying elements. In the other hand, proactive actions will adjust some hiding elements such as enemy spawning order or some inactive object. Challenging rate was used by strategy game but it can apply to action game by changing some of the variable. FUA and FDA is the action order adjustment based but it just change the attack style instead of just use the specific action. Charter five has presented[yz76] a number of game designs[yz77] and blue prints[yz78] to show how to apply previous three DDA to unreal engine. Charter six has discussed[yz79] and evaluated three DDA systems[yz80]. The advantage of CR adjustment is the low amplitude changes and steady adjustment. The advantage of hamlet system is the high effect growth when player in danger but the short game play is the main problem. FDA is not appreciated with this artefact thought FDA effected with steady growth.

However, the game artefact has designed only two kind of attack action. By add more attack actions[yz81], it may make[yz82] the DDA result more accurate. For future work, add more attack action is essential such as attack in the air or three combo attack chain. Magic mana may be considered to apply the limitation for using magic. Stamina state can restrict character movement. For example, character will consume 1 stamina point by running each unit on the ground and regenerate two or three point for every 10 seconds. The AI or DDA need to consider how balance the use of stamina and magic mana.

8 References

Aamodt, A. & Plaza, E., 1994. Case-Based Reasoning: Foundational Issues, Methodological Variations,and system approaches. *AI Communications,* 7(1), pp. 39-59.

Anon., 2010. *giantbomb.* [Online]
Available at: http://www.giantbomb.com/archon-the-light-and-the-dark/3030-3036/
[Accessed 4 Octorber 2015].

Anon., 2015. *Legion of Zelda.* [Online]
Available at: http://www.zelda.com.
[Accessed 9 September 2015].

Anon., 2015. *techopedia.* [Online]
Available at: https://www.techopedia.com/definition/5967/artificial-neural-network-ann
[Accessed 3 October 2015].

Blizzard Entertainment, 2014. *Diablo III.* [Online]
Available at: http://us.battle.net/d3/en/
[Accessed 9 September 2015].

BNL, 2014. *History: The First Video Game?.* [Online]
Available at: http://www.bnl.gov/about/history/firstvideo.php
[Accessed 9 September 2015].

Booth, M., 2009. *valve software.* [Online]
Available at: http://www.valvesoftware.com/publications/2009/ai_systems_of_l4d_mike_booth.pdf
[Accessed 2 October 2015].

Boutros, D., 2008. *Difficulty is Difficult: Designing for Hard Modes in Games.* [Online]
Available at:
http://www.gamasutra.com/view/feature/3787/difficulty_is_difficult_designing_.php?print=1
[Accessed 25 October 2015].

Capcom, 2015. *Devil May Cry 4 Special Edition.* [Online]
Available at: http://www.capcom.co.jp/devil4se/
[Accessed 9 September 2015].

Charniak, E. & McDermott, D., 1985. *Introduction to artificial intelligence.* 1 ed. Boston: Addison-Wesley Longman Publishing Co.

Chen, H., Mori, Y. & Matsuba, I., 2012. Evolutionary Approach to Balance Problem of On-Line Action Role-Playing Game. *Wireless Communications, Networking and Mobile Computing (WiCOM), 2012 8th International Conference on,* 21-23 September.pp. 1-4.

Chin, H. T., Kay, T. C. & Tay, A., 2011. Dynamic Game Difficutly Scaling Uing Adaptive Behavior-based Ai. *IEEE Transactions on Computational Intelligence and AI in Games,* 3(4), pp. 289 - 301.

Copley, S., 2010. *IGCSE ICT.* [Online]
Available at: http://www.igcseict.info/theory/7_2/expert/
[Accessed 7 Octorber 2015].

Croshaw, Y., 2010. *On Difficulty Levels.* [Online]
Available at: http://www.escapistmagazine.com/articles/view/video-games/columns/extra-

punctuation/7820-On-Difficulty-Levels
[Accessed 1 November 2015].

Cryan, M., 2004. *The university of Edinburgh.* [Online]
Available at: http://www.inf.ed.ac.uk/teaching/courses/inf1/cl/notes/Comp1.pdf
[Accessed 4 Octorber 2015].

Dautenhahn, K. & Nehaniv, C. L., 2002. *Imitation in Animals and Artifacts.* 1st ed. Cambridge: MIT Press.

Deloura, M., 2008. *The best of game programming gems.* 1st ed. Boston: MA: Course Technology, Cengage Learning..

Domingos, P. & Hulten, G., 2000. Mining High-Speed Data Streams. *Proceeding of the sixth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining,* pp. 71-80.

Elkin, A., 2012. *Adaptive Game AI and Video Game Enjoyability.* [Online]
Available at: http://cs.union.edu/~rieffelj/classes/497-Finals/Elkin.pdf

ESA, 2015. *ESA'S 2015 essential facts about the computer and video game industry 2015.* [Online]
Available at: http://www.theesa.com/wp-content/uploads/2015/04/ESA-Essential-Facts-2015.pdf
[Accessed 9 September 2015].

Falcom, 2014. *Ys.* [Online]
Available at: http://www.falcom.com/licence/character/ys_e.html
[Accessed 9 September 2015].

Federof, M. A., 2002. *midole east technical university.* [Online]
Available at: http://ocw.metu.edu.tr/mod/resource/view.php?id=1415
[Accessed 5 November 2015].

Forbus, K. D. & Laird, J., 2002. AI and the entertainment industry. *IEEE Intelligent Systems,* Jul-Aug, 17(4), pp. 15-16.

Graepel, T., Herbrich, R. & Gold, J., 2004. LEARNING TO FIGHT. *Proceedings of the International Conference on Computer Games: Artificial Intelligence, Design and Education,* pp. 193-200.

Haugeland, J., 1985. *Artificial intelligence.* 1st ed. Cambridge: MIT Press.

Hunicke, R. & Chapman, V., 2004. [Online]
Available at: http://www.cs.northwestern.edu/
[Accessed 10 Noverber 2015].

Hunicke, R. & Chapman, V., 2004. *AI for Dynamic Difficulty Adjustment in Games.* [Online]
Available at: http://www.cs.northwestern.edu/~hunicke/pubs/Hamlet.pdf
[Accessed 9 September 2015].

Hurst , J., 2015. *Thought catalog.* [Online]
Available at: http://thoughtcatalog.com/jane-hurst/2015/02/12-types-of-computer-games-every-gamer-should-know-about/
[Accessed 14 October 2015].

IGN, 2006. *IGN.* [Online]
Available at: http://uk.ign.com/games/sin-episodes-emergence/pc-758287
[Accessed 9 September 2015].

John, H. H., 1975. *Adaptation in Natural and Artificial Systems.* 1st ed. Cambridge: MIT Press.

Juul, J., 2003. The Game, the Player, the World: Looking for a Heart of Gameness. *In Level Up: Digital Games Research Conference Proceedings,* pp. 30-45.

Kim, K.-J. & Cho, S.-B., 2014. Game AI Competitions: An Open Platform for Computational Intelligence Education. *IEEE Computational Intelligence Magazine,* 8(3), pp. 64-68.

Kurowiak, M., 2015. *Devil May Cry 4 - Game Guide and Walkthrough.* [Online]
Available at: http://guides.gamepressure.com/devilmaycry4/guide.asp?ID=4445
[Accessed 9 September 2015].

Kurzweil, R., 1990. *The age of intelligent machines.* 1st ed. Cambridge: MIT Press.

Leigh , R., Schonfeld, J. & Louis, S. J., 2008. Using coevolution to understand and validate game balance in continuous games. *Proceedings of the 10th annual conference on Genetic and evolutionary computation - GECCO '08,* pp. 1563-1570.

Linddao, S. et al., 2010. Creating appropriated challenge level game opponent by the use of dynamic difficutly adjustment. *Sixth International Conference on Natural Compuation,* pp. 3897-3901.

Lueangrueangroj, S. & Kotrajaras, V., 2009. Real-Time Imitation Based Learning for Commercial Fighting Games. *2nd Annual International Conference on Computer Games,* October.pp. 1-3.

Lu, F. et al., 2013. Fighting game artificial intelligence competition platform. *IEEE 2nd Global Conference on Consumer Electronics,* 1-4 October.pp. 320-323.

Malone, T. W., 1980. What makes things fun to learn? heuristics for designing instructional computer games. *Proceedings of the 3rd ACM SIGSMALL symposium and the first SIGPC symposium on Small systems - SIGSMALL '80,* pp. 162-169.

Martin , M., 2011. *gamasutra.* [Online]
Available at:
http://www.gamasutra.com/blogs/MichaelMartin/20110830/90109/Using_a_Genetic_Algorithm_to_Create_Adaptive_Enemy_AI.php
[Accessed 4 October 2015].

Mealha, O. et al., 2012. A Video Game Level Analysis Model Proposal. *16th Inthernation Conference on Information Visualisation ,* pp. 474-479.

Mitchell, M., 1995. Genetic Algorithms: An Overview. *Complexity,* 1(1), pp. 31-39.

Monster Hunter, 2015. *Monster Hunter.* [Online]
Available at: http://www.monsterhunter.com/
[Accessed 9 September 2015].

Moriarty, C., 2010. *IGN:mega-man-10-review.* [Online]
Available at: http://uk.ign.com/articles/2010/03/31/mega-man-10-review
[Accessed 7 Oct 2015].

Muise, C., Mcllraith, S., Baier, J. A. & Reimer, M., 2009. Exploiting N-Gram Analysis to Predict Operator Sequences. *Proceedings of the Nineteenth International Conference on Automated Planning and Scheduling,* pp. 374-377.

Nakagawa, Y., Yamamoto, K. & Thawonmas, R., 2014. Online Adjustment of the AI's Strength in a Fighting game using the k-nearest neighbour algorithm and a game simulator. *IEEE 3rd Global conference on consumer eletronics,* pp. 494-495.

Nilsson, N., 1998. *Artificial Intelligence.* San Francisco: Calif.: Morgan Kaufmann .

Niranjan, M. & Rummery , G. A., 1994. On-line q-learning using connectionist systems. *CUED/F-INFENG/TR,* p. 166.

Oxford, N., 2015. *about.* [Online]
Available at: http://ds.about.com/od/glossary/g/Action-Game.htm
[Accessed 12 September 2015].

Plunkett, L., 2012. *kotaku.* [Online]
Available at: http://kotaku.com/5885546/a-salute-to-dani-bunten-a-transgender-video-gaming-pioneer
[Accessed 4 Octorber 2015].

Poole, D. & Mackworth, A., 2010. *Artificial Intelligence: Foundations of Computational Agents.* 1 ed. Cambridge: Cambridge University Press.

Poole, D., Mackworth, A. & Goebel, R., 1998. *Computational intelligence :A Logical Approach.* 1 ed. New York: Oxford University Press.

Portnow, J., 2008. *The Fine Art of Balance.* [Online]
Available at: http://gamecareerguide.com/features/478/
[Accessed 5 Oct 2015].

Rabin, S., 2002. *AI Game Programming WISDOM.* 1st ed. Hingham: Charles river media.

Rabin, S., 2008. *AI game programming wisdom 4.* 1 ed. Boston: MA: Course Technology, Cengage Learning.

Reiter, R., 1993. Proving Properties of states in the situation calculus. *Artificial Intelligence,* December, 64(2), pp. 337-351.

Reynolds, C. W., 1987. Flocks, herds and schools: A distributed behavioral model. *ACM SIGGRAPH Computer Graphics,* 21(4), pp. 25-34.

Rich, E. and Knight, K., 1991. *Artificial intelligence.* New York: : McGraw-Hill.

Rintanen, J., 2010. *Aalto University.* [Online]
Available at: http://users.ics.aalto.fi/rintanen/jussi/planning.html
[Accessed 1 October 2015].

Rouse, M., 2012. *whatl.* [Online]
Available at: http://whatis.techtarget.com/definition/decision-tree
[Accessed 1 October 2015].

Russell, S. & Norvig, P., 2013. *Artificial Interlligence A Modern Approach.* 3rd ed. Upper Saddle River: Prentice Hall.

Ruthner, S., 2015. *planetmule.* [Online]
Available at: http://www.planetmule.com/about/

Sang-Won , U., Chung-Ang Univ, . S., Tae-Yong, K. & Jong-Soo, C., 2007. Dynamic Difficulty Controlling Game System. *Consumer Electronics, IEEE Transactions,* 53(2), pp. 812-818.

Sayad, S., 2010. *saedsayad.* [Online]
Available at: http://www.saedsayad.com/k_nearest_neighbors.htm
[Accessed 1 October 2015].

Sha, L. et al., 2011. Create appropriate Challenge Level game oppent by the use of Dynamic Diffculty Adjustment. *Sixth International Conference on Natural Computation ,* pp. 3898-3901.

Singh, H., 2014. Design of Enhanced Arithmetic Logical Unit for Hardware Genetic Processor. *Advances in Computing, Communications and Informatics,* 24-27 September.pp. 549-553.

Sutton , R. S. & Barto, A. G., 1998. *Reinforcement Learning:An Introduction.* 1st ed. Cambridge, Massachusetts : The MIT Press .

Szczepański, T. & Aamodt, A., 2008. *Case-based reasoning for improved micromanagement in Real-time strategy games.,* Trondheim, Norway: Norwegian University of Science and Technology.

Tan, C., Tan, K. and Tay, A., 2011. Dynamic Game Difficulty Scaling Using Adaptive Behavior-Based AI.. *IEEE Trans. Comput. Intell. AI Games,* 3(4), pp. 289-391.

Thompson, J., Berbank-Green, B. and Cusworth, N., 2007. *The computer game design course.* London: Thames & Hudson.

Ubisoft, 2015. *Assasin'c creed home page.* [Online]
Available at: http://assassinscreed.ubi.com/en-gb/home/
[Accessed 2 October 2015].

Ubisoft, 2015. *Assassin's Creed® IV Black Flag.* [Online]
Available at: http://assassinscreed.ubi.com/en-gb/games/assassins-creed-black-flag.aspx

Um, S.-W., Kim, T.-Y. & Choi, J.-S., 2007. Dynamic Difficutly Controlling Game System. *IEEE Transactions on Consumer Electronics,* 53(2), pp. 812-818.

Vallim, R. M. M. & Gama, J., 2010. Data Stream Mining Algorithms for Building Decision Models in a Computer Role-Playing Game Simulation. *Games and Digital Entertainment (SBGAMES), 2010 Brazilian Symposium on,* 8-10 Noverber.pp. 108-116.

Verma, G. K. et al., 2010. *tutorials points.* [Online]
Available at:
http://www.tutorialspoint.com/artificial_intelligence/artificial_intelligence_popular_search_algorithms.htm
[Accessed 9 Oct 2015].

Vorderer, P. & Hartmann, T., 2003. Explaining the enjoyement of playing video games: The role of competition. *Proceedings of the Second International Conference on Entertainment Computing,* January.

Winston, P. H., 1992. *Artificial Intelligence.* 3rd ed. s.l.:Addtion-Wesley.

Wu, Z., Gianvecchio, S., Xie, M. & Wang, H., 2009. Battle of Botcraft: Fighting Bots in Online Games with. *Proceedings of the 16th ACM conference on Computer and communications security,* pp. 256-268.

Yamamoto, K., Mizunol, S., Chu, C. Y. & Thawonmas, R., 2014. Deduction of fighting-game countermeasures using the k-nearest neighbor algorithm and a game simulator. *Computational Intelligence and Games (CIG), 2014 IEEE Conference on,* 26-29 Auguest.pp. 1-5.

Zelda dungeon, 2014. *Ocarina of Time Walkthrough.* [Online]
Available at: http://www.zeldadungeon.net/ocarina-of-time-walkthrough/fire-temple/
[Accessed 7 Oct 2015].

Zelda.com, 2014. *Legend of Zelda: Ocarina of Time 3D.* [Online]
Available at: http://zelda.com/universe/?ref=https://www.google.co.uk/
[Accessed 12 Nov 2014].