University of St Andrews | FOUNDED 1413

Automated Creation of Puzzle games with Constraint Programming

Han BAO

August 27, 2018

170010050

School of Computer Science
University of St Andrews Dissertation MSc in Advanced Computer Science

Abstract

Because “Match Three” puzzles and games have been popular for a long time and more and more people enjoy solving “Match Three” puzzles, there is a need for providing a better player experience for users. Most of players want to play against with an opponent of similar strength rather than be beaten every time [1]. One of the most significant aims when designing video games is to obtain a great player experience. Thus, it is important to generate puzzles and measure their difficulty levels. “Candy Crush Saga” is a classical and typical “Match Three” puzzle. This project involve formulating “Match Three” rules, implementing simulations of “Match Three” puzzles, and performing experiments which measure how well the difficulty measure lines up with real adult users who are studying in the University of St Andrews. 13 participants, spending 11.5 hours in total for my experiment, played five training games copied from the original “Candy Crush Saga” application and another five games which generated by this application. The win rates are collected and the class of the difficulty level which is judged by players after playing each game. There are three difficulty level classes, “Easy”, “Medium” and “Hard”. The AI player finds a most similar strategy on solving puzzles so that this AI player can predict the difficulty level as human players. Finally, the ranking of difficulty level towards each game was compared with the result did by AI player. Finally, the accuracy of difficulty level for five tested games was calculated. From the result, it can be concluded that the ranking of difficulty level by comparing the average win rates is same to the measured ranking by the AI player. In addition, the accuracy of difficulty level classes achieved 80%.

Acknowledgements

I would like to extend thanks to my supervisor of this project, Dr. Christopher Jefferson, for his support and for his highly efficient replies to my emails and questions. As a result of his insightful and timely directions and suggestions, I was able to gain the needed confidence to complete this project, which has been my main objective for embarking on an MSc degree programme in Computer Science.

I would also like to thank Dr. Jonathan Lewis Voss who provided valuable suggestions and assistance before and during the research.

Also thanks to my parents for their support along the way. Thanks to my room-mates and classmates for being there. Without these people, this work would not have been possible.

Last, I am really appreciate about 13 participants, they spent 11.5 hours in total for my experiment. It represents that they almost spent 52 minutes on average. Especially, one participant had spent 2.5 hours on solving 10 puzzles.

Declaration

I hereby certify that this dissertation, which is 12547 words in length, has been composed by me, that it is the record of work carried out by me and that it has not been submitted in any previous application for a higher degree. Credit is explicitly given to others by citation or acknowledgement.

This project was conducted by me at The University of St Andrews from June 2018 to August 2018 towards fulfillment of the requirements of the University of St Andrews for the degree of MSc under the supervision of Dr Christopher Jefferson.

In submitting this project report to the University of St Andrews, I give permission for it to be made available for use in accordance with the regulations of the University Library. I also give permission for the title and abstract to be published and for copies of the report to be made and supplied at cost to any bona fide library or research worker, and to be made available on the World Wide Web. I retain the copyright in this work.

Signed:_____HAN BAO_____ Date: _____August 27, 2018_____

Contents

List of Figures

List of Tables

1 Introduction

1.1 Project Description

There is a history of generating puzzles using A.I., from Chess problems to Sudoku levels. Sudoku is a classic puzzle which is also named as "the Rubik' s cube of the 21st century" and it became popular in Japan in 1986. Then Sudoku achieved international popularity in 2005 [2]. Sudoku, Chess, and other puzzles can be described by constraint problems and then the programmer can establish corresponding models based on these constraints [3]. Additionally, models could give an objective measure of the difficulty of a puzzle instance with grades such as from the easiest to the hardest. Then programmers apply this measuring method to problem instances for the public [3].

Recently there has been a lot of progress in generating puzzles, and also measuring the toughness of these puzzles, for a human player. In 2016, the AI player, Alpha Go, had beaten Lee Sedol and this remarkable competition also represents a result that it is nearly impossible to win against an AI player for puzzles [4]. Alpha Go greatly promoted the ability of AI players to solve puzzles. Alpha Go has since been modernized to the latest version Alpha Zero and this also represents the general trend is taking advantages of technology to solve problems [4]. In this report, I will introduce an effective way to solve puzzles and this method should be similar to the strategy used by human players. There are many puzzles which have been formulated as constraint problems and subsequently solved. This project aims to solve "Match Three" puzzles and measure the difficulty of "Match Three" puzzles.

1.2 Project Aim & Objectives

The main aim of this project is to automatically generate puzzles and create engines to solve puzzles. This project will involve formulating "Match Three" rules, implementing simulations for "Match Three" puzzles and performing experiments which measure how well the difficulty measure lines up with real users. Through implementing engines of solving "Match Three" puzzles, puzzles will be divided into three difficulty levels —- easy, medium and hard. Then, real users try to solve the same puzzles and record their time spent in solving puzzles and also divide puzzles into three hardness levels. The project will also involve comparing the results performed by computers and humans and evaluate the engine's ability to classify the puzzles' difficulties.

1.3 Motivations

As Schwab wrote in his book "AI game engine programming", there are two main goals for pursuing the development of AI [5]. The first is to understand what intelligent entities are and meanwhile understand our humankind better [5]. The second is to build intelligent entities, for fun and profit for most occasions [5]. This project may be seen to address aspects of both

goals mentioned above. The use of constraint programming to solve puzzle games is for fun and profit and how to get such an artificial engine is exactly what we need to know about the intelligent entities.

Russel and Norvig defined AI in an article "A modern approach" in 1995 as the creation of computer programs that emulate four things [6]:

1. thinking humanly
2. thinking rationally
3. acting humanly
4. acting rationally

From these four aspects, "thinking rationally" and "acting rationally" are considered as the more important aspect so that scientists could obtain AI programs as smart as possible such as Alpha Go. Besides, many trials Turing Test trials have been conducted to show that people cannot distinguish the differences between human and machines. For the "acting humanly" aspect, people are more likely to try to make a robot that is the same as a real person than the AI player that people never win for puzzles. Indeed, people have made great progress on these aspects year by year. Considering a situation as below, in a truly enjoyable game such as Chess, a human player may want to have fun playing against an automatic machine player. Due to the fast development of technology, it tends to be difficult for non-professional human chess players to win a Chess game against an automatic machine player, because the AI is too strong. The automatic machine player has been too much powerful than human in the aspect "thinking rationally". AI players are commonly able to consider many more combinations of moves than the human player can, thereby giving them an advantage. Therefore, many people play Chess is for entertainment instead of any academic learning or thinking training. Most of them want to play against with an opponent of similar strength rather than be beaten every time. One of the most significant aims of playing video games is to obtain a great player experience (PX) [1]. However, it is not easy to evaluate PX using any simple device or other physical experiments [7].

In Chess games, what normal players need is an automatic machine player with a suitable challenge rather than that automatic player overwhelm the human by always making the best move [5]. Such opponents would possible make mistakes occasionally which are more like humans or even worse than human normal levels. Then it may even aid human enjoyment, if an AI player could adjust its ability to that of its human opponent on occasions. Whereas in non confrontational games, such as "Candy Crush Saga", there is no automatic player so that we need to list the difficulty of initial boards and goals for normal human players. This project seeks to investigate aspects relating to using an AI to easily produce Candy Crush Saga game boards with suitable goals for human players that can be both challenging and enjoyable.

"Match Three" is a typical puzzle video game where the player manipulates tiles in order

to make them disappear according to a matching criterion. The matching criterion is that there are at least three tiles of the same type adjoin each other [8]. A field has an N×M matrix of spaces, where N and M are integers greater than three, and wherein each space of the matrix includes one of a plurality of different items. Then an object is allowed to swap to the nearest object once three are three or more identical items results in the items. Finally, these items would be removed from the field.

Because "Match Three" games are very simple games with a very limited number of rules, it can be easily understood by many different age groups and appears to appeal to both young and old. Over time "Match Three" puzzle games have become more and more popular. Bejeweled, Candy Crush Jelly Saga and Ruby Blast are the most famous "Match Three" puzzle games that the number of downloads from the app store has more than billions of times [9]. Many popular games are all based on this puzzle and the most famous series is "Candy Crush Saga". According to reports by Wall Street Journal in 2013, nearly 15 million people in Western countries were addicted to Candy Crush Saga [9] [10]. These data show that people enjoy playing "Match Three" puzzle games. Thus, in this project, I aim to use "Candy Crush Saga" as the typical "Match Three" puzzles and aim to implement processes and mechanisms to aid measurement of the difficulty of puzzle levels.

2 Context Survey

In this chapter, I will introduce background and basic rules of "Match Three", review existing methods on solving "Match Three" and how to establish the difficulty level and explain reasons why I choose these criteria.

2.1 Introduction of "Match Three"

Match Three games are a type of casual puzzle games. The major task consists in forming lines/chains/groups of 3 or more same identical tiles. The traditional game board is square-patterned and filled with various tiles which could shift, select or rotate. Eugene Alemzhin created the first Match Three game by swapping adjacent balls named "Shariki" in 1994 which was released for DOS [11]. If there are no more possible matches in the board, then the game is over with current score shown in the screen. Then this kind of influential game led to the popularity of "Match Three" puzzle games. Nintendo published "Panel de Pon" in 1995 and "Tetris Attack" in 1996 [12] [13]. The former one is the original version released and the following one is the first game in Puzzle League series. In 2000, "Pokémon Puzzle League" developed by Nintendo which features the same gameplay as in "Panel de Pon" [14]. However, different from the predecessors, it was developed with a 3D mode instead of the traditional 2D mode. With the development of mobile phones, people realized that they would like to play more mobile games, a convenient and relaxing kind of gameplay.

In 2001, "Bejeweled", the first most famous mobile "Match Three" game released by PopCap Games [15]. There are more than 150 million times of Bejeweled downloaded from App stores [15]. The objective of this game is to swap one gem with an adjacent gem to form a horizontal or vertical chain of three or more gems [16]. If there are more than three gems connected, it provides bonus points. Once these gems are connected, they would disappear and fill new gems in. That would lead to chain reactions, called cascades, are triggered, where chains are formed by the falling gems. The cascades bring more bonus points as well. Additionally, there are two main goals for players, including complete a certain score in a limited time or limited steps. Since the great popularity of Bejeweled, more and more "Match Three" puzzle games appear in App stores, such as Candy Crush Saga, Bubble Witch Saga, Jelly Splash. Those games have long since caught on all over the playing planet. The reasons why an enormous quantity of people is addicted to "Match Three" puzzle games as noted. It contains two main parts, including "Broader Context" and "Game Design" [8]. In the "Broader context" part, it has four prevailing features.

1. **Demographics.** Compared to traditional video games, casual games are more oriented towards women and the adults who are over 35. They have more free time and more patience playing the same game.

2. **Distribution.** The size of causal games currently tends to be smaller than 10MB, which only takes up a little space of mobile phones. Additionally, most of them do not need to

connect to a network. In other words, people are free to play when they take a subway without any network.

3. **Hardware.** These games do not tend to require advanced pieces of equipment or expensive external devices. Even a user's phone or computer is not the latest version, they could still play games fluently.

4. **Economic model.** "Match Three" games are often free to play. Even though they are some krypton gold in game, the cost is still quite small than most other video games.

Apart from the broader context part, the other significant part is game design [8] [17].

1. **Allow short playing sessions.**A game can often be finished in a few minutes. Most busy people, such as young child's mother or students who only have 10 minutes break during classes, would have more chances to play such games.

2. **Auto-save.** If players have some unexpected emergence cases have to deal with when they are playing, they can stop games by shutting the screen or closing the application. A commonly in-built auto-save feature makes it easy to resume playing the game.

3. **Easy control.** "Match Three" games do not require players to reply a fast response or any smooth operation. They can operate moves slowly.

4. **Very simple rules.** People tend to easily understand how to play without much much prior knowledge of the game, thereby making it.

5. **Moderate innovation.** For example, there are six regular candies in "Candy Crush Saga". However, there are more than ten kinds of different new candies or tools in "Candy Crush Saga". Thus, it has thousands of games and could develop more games as many as the company wants without having to make any substantial changes.

6. **Multiple levels of success.** Owing to many different levels of "Candy Crush Saga", players may succeed more easily than other normal video games. Meanwhile, it also provides some small missions during playing puzzles, such that players can get a small reward if the ranking of they are currently ranked top among all friends. While playing a game in "Candy Crush Saga", players can obtain three stars if the score is much higher than the expected. If the score is a little higher than the expected, players still can obtain one star. Those successes encourage players to play better every time.

7. **Much positive feedback.** Because players have to play "Candy Crush Saga" from the easiest to the hardest, players gain experience of victory pretty early on. If players cannot find the next move, a game may provide some tips or tools to help them overcome difficulties.

8. **Little negative feedback.** If players fail to pass, there is no punishment for players about their mistakes.

9. **The ranking of competition.** For example, "Candy Crush Saga" has more than thousands of games and players' accounts have been connected to their Facebook or

Twitter accounts. This way, players can see the ranking of all friends who are also playing "Candy Crush Saga". This encourages players to play more games than other competitors.

10. **Categorizing is human nature.** Whether types of coffee, colours of clothes, types of fruits, trends in popular music, there is a natural human desire to categorize objects and experiences [18]. Because categorizing is a kind of human nature and people tend to find it easy to match three objects without any training, people probably enjoy categorizing gems in games.

Thus, "Match Three" puzzle games become more and more popular.

2.2 Candy Crush Saga

This section introduces the "Candy Crush Saga" game along with its rules and characters. "Candy Crush Saga", which is developed by King Digital, is a "Match Three" puzzle game and is developed from the browser website game"Candy Crush". "Candy Crush Saga" has been adapted for various platforms, including IOS, Android, Windows Phone and Windows 10. The theme of "Candy Crush Saga" is different candies and appears to appeal to many different age groups over the world. The basic rule is to combine three, four or five candies together by swapping two adjacent candies. Then candies disappear and score rises up. Players need to put different candy groups together and detonate, which will produce different effects and different scores. When one candy is moved, other candies would have chain reactions when their positions are modified. See Figure 1, the chain reaction brings Combo and Combo brings bonus to improve the current score. Then, players could gain progressively more points. Each game stage has to be unlocked one by one, and each mode is different with randomly dropping candies. There are many kinds of candies in Table 1 shown [19]. First, the basic board is made up of regular candies. Additionally, there are six regular sweets, the red candy (jelly), the orange candy (lozenge), the yellow candy (the lemon sugar), the green candy (gum), the blue candy (lollipop) and the purple candy (grape flavour candy).

Furthermore, there are some special candies, such as striped candies, wrapped candies, and chocolate candies [19]. Striped candy has two types that there are horizontal or vertical white stripes on a regular candy. If a player sweeps two adjacent candies when one candy is moved horizontally, then four candies disappear and a new horizontal striped candy appears with the same colour (see the first figure in Figure 12). If the movement is vertical, then the new striped candy is vertical striped candy with the same colour. A horizontal striped candy can destroy the whole column of candies and a vertical striped candy can destroy the whole row of candies. The second special candy is wrapped candy. There are two methods to make. If there are five same colour candies consisting the "L" or "T" shape (see the second figure in Figure 12), then five regular candies disappear and the same colour wrapped candy appear. A wrapped Candy could make the 3*3 explosions twice in the game board (see Figure 2), eliminating all nearby candies and obstacles. The last special candy is the most powerful candy named chocolate candy. Only if there are five same colour candies in a row or in a

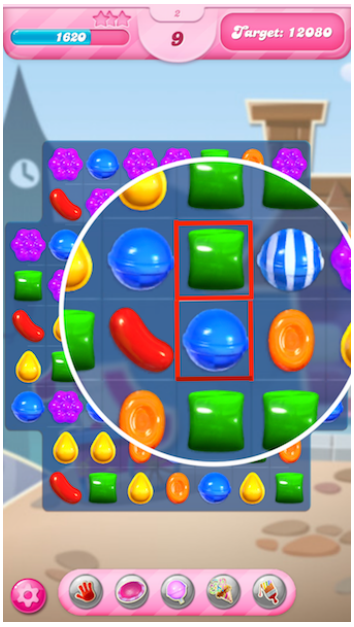

Figure 1: An Example of a chain reaction. If the player swap the selected Blue Candy and the selected Green Candy, and three blue candies in a row and blue candies would crush. Then the green candy would fall down and they will be in a column. It is an example of chain reaction process.

Image	Candy Name	The score if creates this candy	Image	Candy Name	The score if creates this candy
	Red Candy	0		Orange Candy	0
	Yellow Candy	0		Green Candy	0
	Blue Candy	0		Purple Candy	0
	Blue Horizontal Striped Candy	120		Blue Vertical Striped Candy	120
	Blue Wrapped Candy	200		Chocolate Candy	200

Table 1: Six Regular Candies And Four Special Candies.

column, they disappear and a new chocolate candy appears in their place (see the third figure in Figure 12). If the chocolate candy is swapped with any other regular adjacent candy, all the same colour candy on the board would be eliminated.

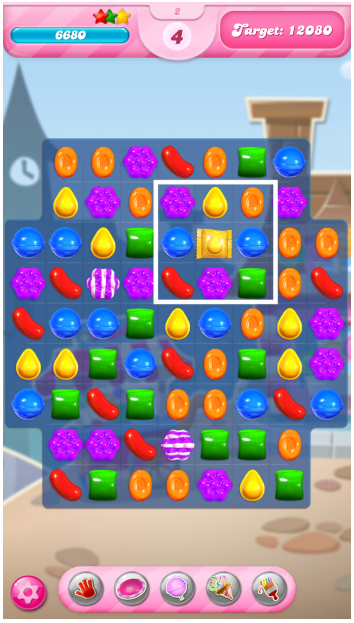

Figure 2: The explosion area of a Wrapped Candy crush. If the yellow Wrapped Candy crushes, the white rectangle area would explode twice.

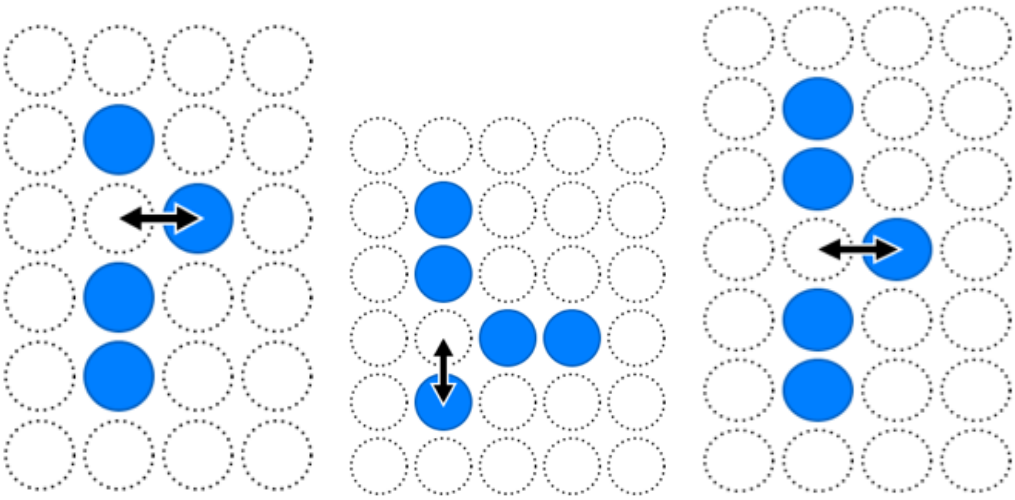

Figure 3: Except for matching three same candies in one row or in one column, there are three kinds of possible movements.

Except for these combinations, there are some hard combinations of special candies. They have high powerful effects and players can obtain much higher score after these combinations (see Table 2) [19].

One coin has two sides, there is no exception of "Candy Crush Saga". There are many doctors believe that indulging in playing the game would lead to a variety of health problems,

Image	Candy A	Candy B	Effect
	Striped Candy	Striped Candy	Eliminate an entire line, an entire row of candies.
	Striped Candy	Wrapped Candy	Striped Candy and Wrapped Candy turn into a huge candy and destroy three whole rows and three whole columns of candies.
	Wrapped Candy	Wrapped Candy	A mass explosion created. Clears 24 candies around (5*5 explosions).
	Striped Candy	Chocolate Candy	If the Chocolate Candy and Striped Candy exchange positions, then all the candies whose colour is as same as the colour of Striped Candy all randomly turn into the Horizontal or Vertical Striped Candy, then they all eliminate with the explosion.
	Wrapped Candy	Chocolate Candy	If the Chocolate Candy and Wrapped Candy exchange positions, then all the candies whose colour is as same as the colour of Wrapped Candy all turn into the Wrapped Candy, then they all eliminate with the explosion.
	Chocolate Candy	Chocolate Candy	All candies are cleared on the board.

Table 2: Combinations of two special candies. Special candies can be combined together to produce special amplified effects.

such as presbyopia, cervical displacement, muscle inflammation and so on [20]. And Fred Richmond, the executive director of behavioral health services at Mission Hospital, pointed out that excessive indulgence in playing the game can cause a variety of psychological and physiological problems and then have a negative impact on adolescents. However, there is no doubt that this puzzle game brings much enjoyment and satisfaction. The data published by King Digital in 2013, "Candy Crush Saga is played over 700 million times a day", also proves

the charming of this puzzle game [21].

2.3 Solve "Match Three" Puzzles

Based on the simple rules on "Match Three" puzzles, the falling candies are unpredictable. Daniel Hadar & Oren Samuel introduced three algorithms for solving "Candy Crush Saga" [22]. The first algorithm is Stupid Greedy Search (SGS). SGS algorithm finds all the valid swaps and computes their direct score and choose the highest score. The second method is Heuristic Greedy Search (HGS) which is similar to SGS. Besides, it combines with some heuristics, such as the amount of pairs in the current board after the swap, the number of possible moves in the resulting board and the row where the swap was performed (1-based counting from the top of the board) [22]. The last method mentioned is the Limited Breadth First Search(L-BFS) which is similarly to HGS. However, L-BFS chooses the best move using simulation of cascades and using heuristic scoring until the board is in a game-over state. Each heuristic has different weights and then combine all heuristics together with SGS, HGS or L-BFS. Finally, they find top 10 heuristics as Figure 4 shown and we could see the level of SGS is quite similar to the human level. Because what I need to develop is an AI player which can measure the difficulty of each board and most people should agree to the result conducted by this AI player. Thus, SGS is a great choice for my AI player.

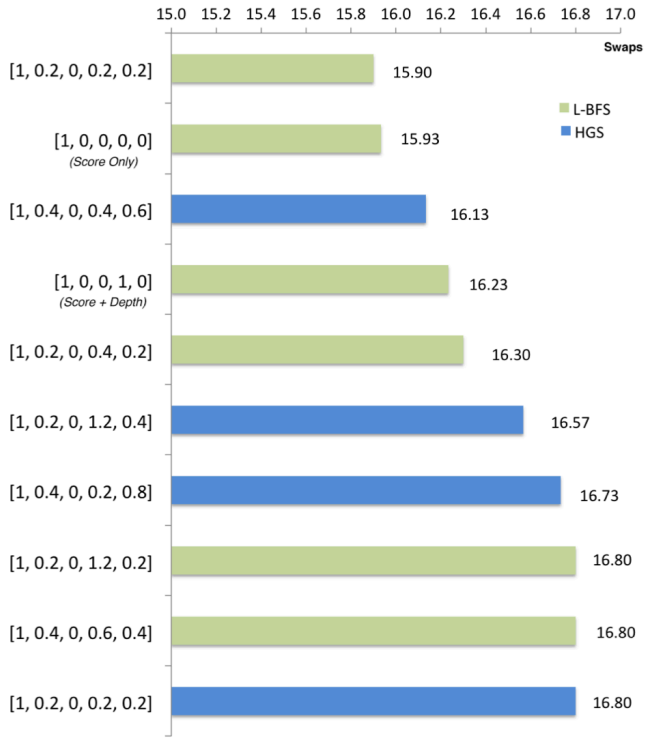

Figure 4: The top 10 heuristics and their average swap count. SGS and human results are too far out to show (at 25.2 and 28.8, respectively). [22]

2.4 Levels of Difficulty and Design

Though the background of "Candy Crush Saga" is important, level design and difficulty need to be considered as well.. This part would bring two questions:

1. Why do we need to design levels with different levels of difficulty, and what impact will this have on the game?

2. How to determine the difficulty of these levels? What criteria can be used as a reference standard?

For the first question, it has been mentioned in the "game design" part in Section 2.1. Because "Candy Crush Saga" does not only bring multiple levels of success, but also encourages players to gain a higher ranking of competitions. Moreover, great game design brings a better PX. Elisa and her group members have introduced a great experimental design and analytical method for measuring enjoyment which had been identified as a central component of the player experience [23]. However, PX has various, overlapping concepts so that it is difficult to develop valid measures and a common understanding of game enjoyment. Firstly, they structured the review of 87 quantitative studies into general methodological observations in an experiment. Then according to the "Purpose of the studies", "Participants", "Games and Genre", "Study setting, gameplay duration and game metrics", and "Measuring point of the critical element of experiment" aspects, more in-depth measures were developed by the institute [23]. Finally, the determining factors affecting the enjoyment of the game are Game System, Player and Context and Relationship between enjoyment and other PX components such as flow, presence and immersion. Thus immersion is a great critical factor of a great player experience [23]. In other words, the player has a desire to play a game, and it can prove that the player has a good game experience in the game. Immersion can be seen as the total time spent on "Candy Crush Saga". The more time players spend in a game, the stronger players' immersion. In other words, if players have an enjoyable experience in a game, they would like to spend more time and money on this game.

A similar experiment was conducted by Su Xue, Meng Wu and et al. on the relationship between difficulty and engagement [24]. They convinced themselves of a causal link between difficulty and engagement. Thus, they believed that a Dynamic Difficulty Adjustment framework with a global optimization objective of maximizing a player's engagement throughout the entire game [24]. Then they presented "Candy Crush Saga" and "Bejeweled" with Dynamic difficulty adjustment (DDA) implementations. Thereafter, they concluded two significant figures. The first shows that retained population (the red line) at a level is the number of players who have achieved this level as the highest one in Figure 5. There are players churned at each level, thus the retained population decreases as the level increases. The difficulty (the blue line) is measured by the average number of trials that are needed to win this level. The more trials it takes, the more difficult this level is. According to the win rate and the number of players, all levels have been divided into three difficulty levels, easy (<20), medium (21-80), and hard (>80). Firstly, we can easily find that the difficulty level is rising as the number of

level in figure 5, but the number of players is decreasing as the number of levels. Secondly, the win rate of each level tends to be volatile. For example, there are seven peaks between 40 level and 60 level. It represents that every three levels there is a much more hard level. After that, the difficulty of the level is restored to a relatively simple or medium difficulty. Figure 7 shows different initial candies board, which is affected by the random seed, have a relationship with the win rate. Because the random seed of board initialization is different, the initial candies would be different. The researchers used random seeds from 0 to 99 for the experiment, Figure 7 shows how the difficulty factor of the same game is also changing, from as low as 0.15 to as high as 0.75. This difficulty range can be said to have crossed from easy to hard. And by publishing the experiments after the game, they concluded that while existing DDA systems adapt game difficulty in a greedy manner for local benefit, their method maximizes the player engagement throughout the entire game.

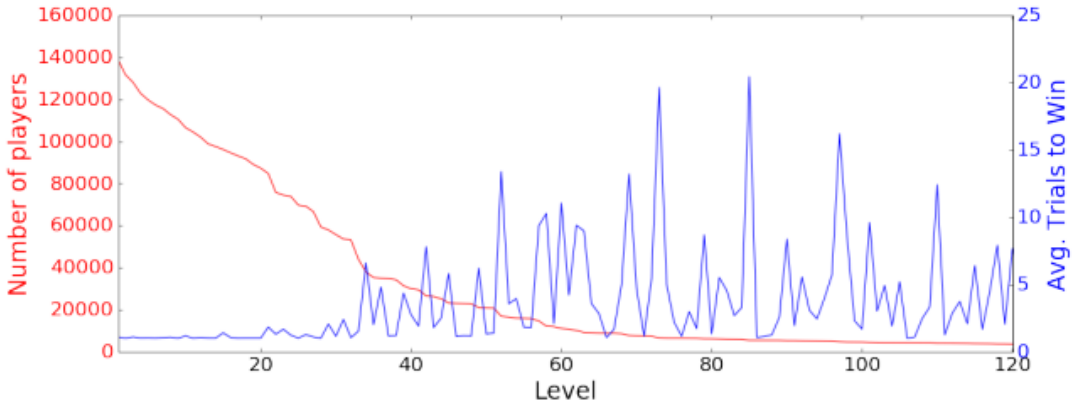

Figure 5: The retained population of players (red line) versus difficulties (blue line) by level. The red line represents, for each level, the number of players who have ever achieved it. The blue line represents the level difficulty, which is measured by 1/win rate, i.e., the average trials needed to win this level. We can observe the strong impact of difficulty on population retention, in particular for middle stage levels. [24]

In order to increase the interests of the puzzle game, it is necessary to design the game more ingeniously, such as adding some game props or changing the grid of the game board. Modifications would have an impact on the difficulty of levels. Four significant criteria, which would affect the difficulty level, should be noted [25]. The first is the complexity of the terrain, the goals of the level. The complexity of the terrain is the most basic element of a game. This involves two core parts, static complexity and dynamic complexity. The static complexity means that the complexity of the basic terrain itself, independent of the game process.The static complexity has a core factor: the average connected number. It is derived from the function: Grids Difficulty = Total Connected Number / Effective Number Of Grids [25]. The Connected Number means how many possible moves on the board regardless of the colours of candies. For example, there are six possible connected "Match Three": A3-B3-C3; B3-C3-D3; C3-D3-E3; C1-C2-C3; C2-C3-C4; C3-C4-C5. The maximum connected number is 6 for such

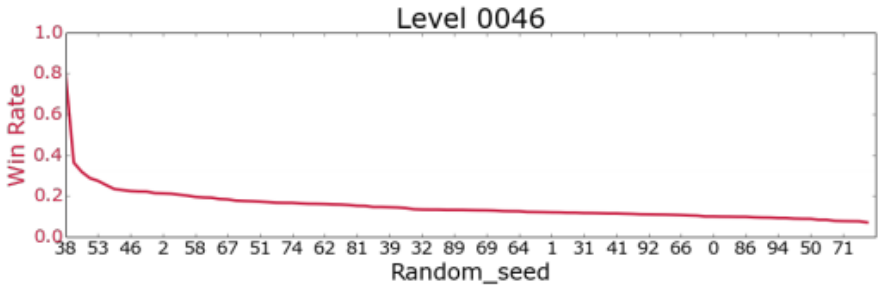

Figure 6: Difficulties of various random seeds at a level of the match-three game. The difficulty is measured by the win rate of a certain seed, i.e., the percentage out of all trials with this seed are actually wins. The variance of difficulties across seeds is large. We can see that the easiest seed (leftmost, seed 38) shows a win rate up to 0.75. In contrast, the hardest seed (rightmost, seed 71) has a win rate as low as 0.15 [24].

cell. Effective Number Of Grids means how many candies on the grid. And as such, if the

	1	2	3	4	5	6	7
A	0	0	1	0	0	0	0
B	0	0	2	0	0	0	0
C	1	2	6	4	3	2	0
D	0	0	4	4	4	3	0
E	0	0	3	4	4	3	0
F	0	0	2	3	3	2	0
G	0	0	0	0	0	0	0

Figure 7: Connected Number: Owing to the function: Grids Difficulty = Total Connected Number / Effective Number Of Grids. Effective Number Of Grids means how many candies on the grid. White cell means there is no candy on the grid, and blue cell means there is a candy. "6" in C3 means there are six possible connected "Match Three": A3-B3-C3; B3-C3-D3; C3-D3-E3; C1-C2-C3; C2-C3-C4; C3-C4-C5 [25]

level of elimination of the level is the same, he can divide the terrain difficulty of the game into three levels:

- High: The average connected number is >=3.6. Such a level is generally smoother and Combo is frequent.
- Medium: The average connected number is between 3.2 and 3.6. Although the number of Combos is not too large, it is not too difficult to play.
- Low: The average connected number is <3.2. In such a level, the player often has to

spend some effort to find a few operational combinations on the disc, because resetting the situation is relatively frequent because there is no operation.

However, there may be a high degree of static complexity, but the actual game operation is very difficult, because it involves dynamic complexity. The dynamic complexity means that the complexity characteristic of the process of elimination and falling behaviour during the actual game of the player. Dynamic complexity also has a core factor named the Combo problem. When a candy is eliminated, the fallen candies or the remaining candies will cause the chain reaction to occur. The appearance of these Combos will greatly change the difficulty of the game. These Combos are hard to predict in the game. Apart from the complexity of the terrain, the goals of the level, the special elements in the level and the restrictions of the level are also significant. The goal of the level may be a specific score, or the number of candies eliminated, or a combination of the two [25]. Then the higher the target number or score, the more difficult it is. The special elements in the level, such as some novelty items, will increase the difficulty of the game, such as chocolate candies. Level restrictions such as the number of steps and time will affect the difficulty of the game. However, these four things only affect the difficulty of the game, and can not be used as a direct basis for judgement. Consequently, a game which is designed to optimize the level of difficulty will have a profound impact on PX. This is why we need to design a reasonable game difficulty for the player.

After understanding the importance of game design, programmers could generate many different levels of games [17]. Meanwhile, here is another question about how we can identify these levels. Because the computer could generate thousands of different games artificially and mark the difficult level automatically. If the hardness of a game is determined by human beings, it may lead to more problems. On the one side, individuals have to spend a lot of time to judge the difficulty level of each game. It is inefficient, wasting time and possible to be a one-sided testing method. On the other side, different people have different standards for the same game. Thus, it must be the program itself that determines the difficulty of these different level games. This involves another matter whether the difficulty of a game determined by the program is the same as that of human beings. Therefore, an important experiment is needed for this project. The experiment requires two elements. Human and programs should divide all games into three levels, easy, medium and hard. The experiment should compare the consequences of the two groups.

Christopher Jefferson, Wendy Moncur and Karen E. Petrie [26] had performed an experiment on an automated generation of puzzles named Combination solved by constraints, which indicated that the fun and immersing computer games could be generated by constraints. They explained how all the levels of Combination were generated, checked for correctness and rated for difficulty completely automatically through the use of constraints. Then they found that running the Constraint Programming a number of times using different variable orderings then averaging the result could provide a more satisfactory player experience. Finally, this application was released on iTunes and gained a great commercial success and received good reviews.

2.5 Summary

To summarize, there is a substantial body of work that focuses on finding the reason why "Match Three" games so popular and how to solve "Match Three" puzzles. However, limited work about how to evaluate the difficulty level of game.

Firstly, "Match Three" has two main reasons that lead to its popularity among the world, "Broader Context" and "Game Design". It allows people of all ages to play with a most basic device. The download file is small and it does not need Internet connection either. Because of these four items, the "Context" of "Match Three" games is "Broader" [8]. Players could gain multiple levels of success and much positive feedback with a few minutes. The features, such as"Auto save" and "Short playing session", allow users to play games in some gaps and stop playing when they have some emergency cases. Programmers could develop many games based on multiple special tools such as "Chocolate Candy" so that players always have a sense of freshness [8]. Then players are addicted to play such games with great immersion. The immersion is one of most significant factor on evaluating the player experience on a game [23]. Consequently, people have a great PX on "Match Three" games.

Secondly, a normal player could have a better PX if he/she has an opposite player with a suitable challenge rather than that opposite player overwhelm this normal player by always making the best move [5]. See Figure 7, "Candy Crush Saga" does not design such levels that the difficulty of level must be increased by the rising number of levels. Players would play a "Hard" difficult level after playing some "Easy" or "Medium" levels [24]. However, there is no such opposite player for "Match Three" games. Consequently, the grid map and task requirements of each level could be seen as the opposite player. This is why evaluate the difficulty of each level is important for players.

Thirdly, Daniel Hadar & Oren Samuel discussed three algorithms on solving puzzles [22]. Stupid Greedy Search (SGS), Heuristic Greedy Search (HGS) and Limited Breadth First Search(L-BFS) are introduced to solve puzzles. Then HGS and L-BFS can combine with heuristics. Each heuristic would have different weight from 0 to 1. SGS is the most closest to Human level [22].

Fourthly, There are two criteria which would be used to evaluate the difficulty of these levels. The first one is the win rate mentioned in Su's experiment [24]. In addition, the second is the time spent on solving a puzzle mentioned by Christopher Jefferson [26].

3 Requirements specification

This chapter gives the detailed problem statement. Then the requirements summarized are listed through background chapter. These requirements have an greatly impact on the project objectives and the design of experiments, and also provide guidelines for designing and developing the applications. There are two parts including functional requirements and non-functional requirements. The last section contains the use case diagram for this project.

3.1 Functional Requirements

My project main aim is to automatically generate puzzles and create engines to solve puzzles and then evaluate the difficulty levels of puzzles. Then real users try to solve the same puzzles and record their time spending in solving puzzles and also divide puzzles into three hardness levels. Finally, compare two results performed by computer engine and human players and evaluate the engine ability on classifying puzzles difficult levels. Consequently, there are three main tasks. The first is to develop a "Match Three" application that generate puzzles automatically. The second is to build an intelligent engine. The last one is to perform experiments on human players and the engine player. Based on these three tasks, functional requirements and non-functional requirements are listed separately according to their different uses:

3.1.1 "Candy Crush Saga" Java Application

1. This application shows the grids of candies, target, current score and the steps left. Once players move successfully, the board should be refreshed and data should be updated.

2. This application allows players to move candies if the move is valid. The move must be completed by two adjacent candies, and this move must meet at least one of these conditions:

   - if there are three or more than three same colour candies in a row or in a column.
   - if there are five same colour candies consisting the "L" or "T" shape.
   - if two adjacent candies are special candies or one of them is "Chocolate Candy". The result of the combination of two special candies shown in Table 2.

3. If the current score is no less than the target score when the left steps is zero. A message box would ask human players to continue this game or give this level up. Human players could click one "YES" to play current level again or click "CANCEL" button to play next level game.

4. Players can gain score if candies crush. The specialized score of each crushed candy as following table 3:

Candy Crush or Combination Crush	Score	Bonus
Three regular candies	60	None
Four regular candies	120	Striped Candy with same colour
Five regular candies	200	Chocolate Candy or Wrapped Candy
Regular candies + Vertical Striped Candy	120	Candies in the whole row of this horizontal striped candy would crush as well. Each regular candy score is 60.
Regular candies + Horizontal Striped Candy	120	Candies in the whole column of this horizontal striped candy would crush as well. Each regular candy score is 60.
Regular candies + Wrapped Candy	1080	As the Wrapped Candy as the centre, 5*5 candies around this candy would crush twice. Each regular candy score is 60.
Striped Candy + Wrapped Candy	720	Three row and three column candies crush. Each regular candy score is 60.
Chocolate Candy + Regular Candy	0	Chocolate Candy would be seem as the regular candy. All candies which colour is same to the regular one would crush. Each regular candy score is 60.
Chocolate Candy + Striped Candy	0	Chocolate Candy would be seem as the striped candy. All candies which colour is same to the striped one would change to striped candy (It could be Horizontal and Vertical Striped Candy. This is a random choice). Each striped candy score is 120 and would lead to whole row or whole column crush. Each regular candy is 60.
Chocolate Candy + Wrapped Candy	0	Chocolate Candy would be seem as the wrapped candy. All candies which colour is same to the Wrapped candy would change to Wrapped candy. Each Wrapped candy score is 200 and would lead to 3*3 crush twice. Each regular candy is 60.
Chocolate Candy + Chocolate Candy	0	All candies on the board crush and chocolate candy is seen as a regular candy. Each regular candy is 60.
Combo[1]	0	The original Score * Combo Times

Table 3: The specialized score of each crushed candy.

3.1.2 AI Player

1. The AI player should be developed based on the "Candy Crush Saga" application and simulate the strategy which applied by human players.
2. The AI player should choose the best move according the current board using SGS algorithm.
3. The AI player should record time spent on playing games and the final score.

3.1.3 Project Constraints

1. The project has to be accomplished by 27th August 2018.
2. The programming language used should be Java.
3. The application should be runnable on different operating systems, including Windows, Linux and OS X.

3.2 Non-functional Requirements

The application should have good usability. There are four items mentioned by Nielsen [27] should be considered more carefully:

1. Learn-ability: Games should be organized so that new users are able to accomplish basic tasks. For example, new players are not familiar with special candies and the results of combinations of special candies. These messages should be told to them before the experiments.
2. Efficiency: The operation should be very simple that players can accomplish tasks quickly. Each level game should not last for more than 5 minutes.
3. Memorability: This application cannot allow users back to previous games or read the history when users return to the design after a period of not using it. However, players can easily remember the rules in a short time.
4. Errors: When players make moves, the application should check each move so that the application could prevent severe errors.
5. Satisfaction: The design of the application should bring users sense of pleasure rather than frustration or aggravation. Player experience is the most significant element for a game. Players could skip current level game off or exit the whole game as they want.

3.3 Use Case Diagram

The basic use case diagram of "Candy Crush Saga" Application is shown in Figure 8. It allows human players and the AI player to play games on the board. Each game has one board which is full of candies and blocks. It requires players to collect scores by swapping

candies, and then reach the target score with limited steps. Human player could use Mouse to select candies to accomplish tasks. AI player should be able to try every possible moves and then select the best move.

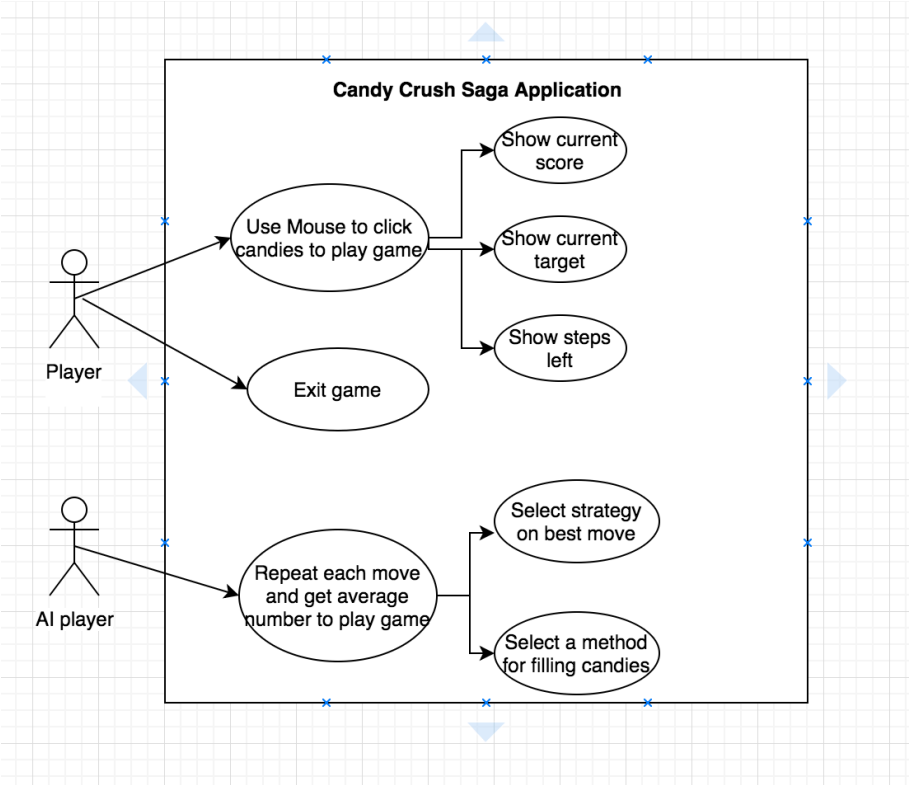

Figure 8: The basic use case diagram of "Candy Crush Saga" Application.

4 Ethics

Because this project needs human players involved, ethical considerations should be considered. This project involve formulating "Match Three" rules, implementing simulations of "Match Three" puzzles, and performing experiments which measure how well the difficulty measure lines up with real adult users who are studying in the University of St Andrews. Participants will be internal to the University, and will be asked to provide anonymous feedback on the effectiveness, design and utility of a project-based artifact. Questions are limited to the artifact and opinions of the artifact. Through implementing engines of solving "Match Three" puzzles, puzzles would be divided into three difficult levels. Then, participants would try to solve the same puzzles and record their time spending in solving puzzles and also divide puzzles into three difficult levels in Computer Science Lab at St Andrews University. They can stop or exit this experiment if they want. The data will be used to compare the results performed by computers and humans and evaluate the engine ability on classifying puzzles difficult levels. Participants will indicate informed consent by reading and signing a form. All participants are anonymous and only their spending time on solving problems would be saved. I would use an "Timer" in the program and record this data in the Excel file. All raw data will be destroyed after submission of the project. No personal data is needed nor collected. The Ethical Approval Document is shown in Appendix A.

5 Design

This chapter describes the designs of the application "Candy crush Saga" and experiments. The structure of application is introduced first, including game part and player part. Most basic requirements of the application has been written in Section 3 "Functional Requirements" part. However, these requirements are not specified enough and design part would introduce more detailed design content. Game part contains user interface and game board. Player part contains two modes that human players and AI player could perform their operation on the same application. This part will use a class diagram to help understand. After that, the experiments design will be introduced. It mainly has three parts, including how to generate random "Match Three" puzzles, how to choose suitable boards with a suitable target and limited steps, and how to compare the results of human players and the AI player.

5.1 Application Design

This chapter presents the graphical user interface and the architecture design for the application. High-level and Low-level architectural designs are presented. "Candy Crush Saga" is a typical game of "Match Three" puzzles. Thus, this application would be developed based on the real mobile game "Candy Crush Saga".

5.1.1 Top-level Architecture

The top-level architecture of this application is an N-part design in which concerns are separated into game and player part. Game mainly has two parts, including the user interface and game board. Players could be divided into human players and the AI player. Each part has its own usage and Figure 9 presents the top-level logical architecture of this application.

5.1.2 Low-level Architecture

This section will explain in detail the two modules, Game Part and Player Part, and other important classes they contain. Specifically, this section should explain their internal connections and the reasons for choosing them.

- **Game Part**

  Game Part is introduced firstly because this part is the fundamental part of whole application. It mainly consists of two important classes, one is the game itself and the other is the game board. Besides, the game user interface is determined by the board, and the board is controlled by the interface. See Figure 10, this "Game" class means one whole level game. The parameter "level" in parentheses refers to the level number. The information for each level is stored in a "Level_N.txt" file and each level file has different information, and then players could have different games to play. The board

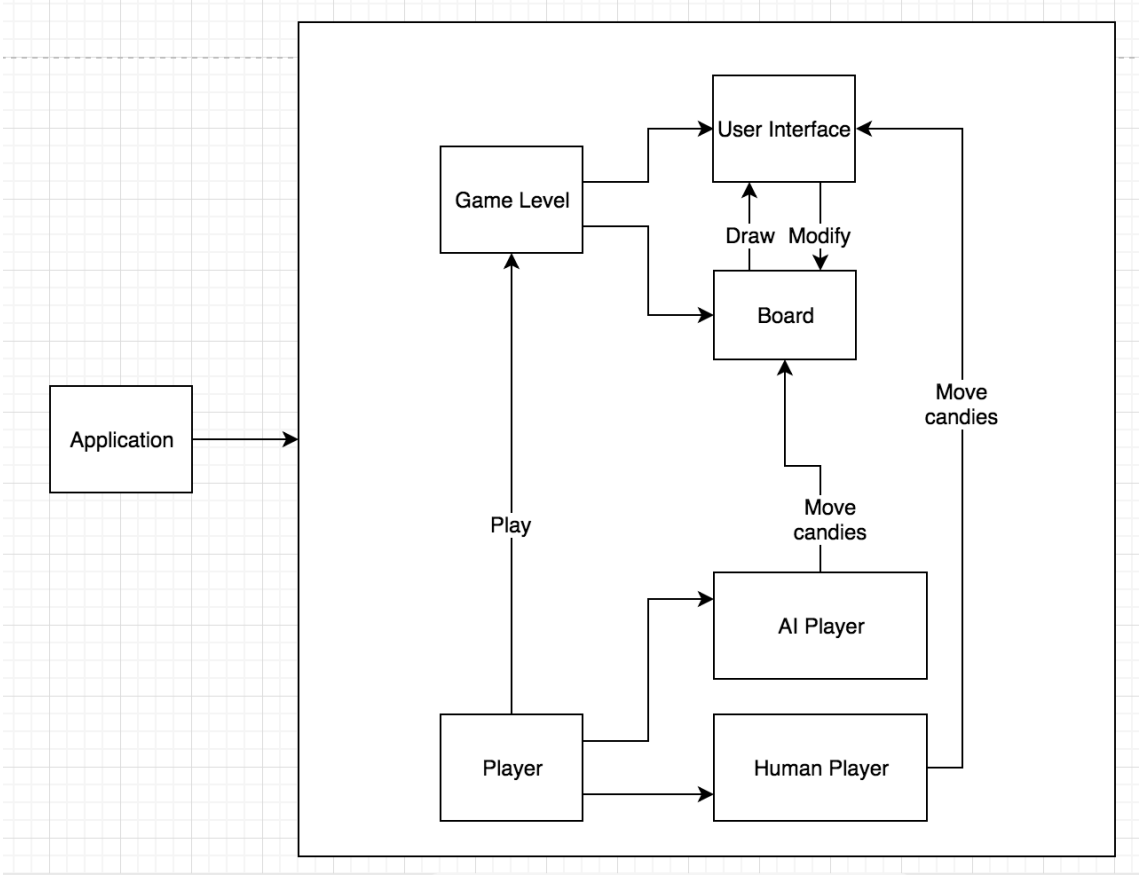

Figure 9: The Architecture of application “Candy Crush Saga”.

information of each level needs to be read from the specific files. It requires the ability to read “Level_N.txt” files and write important messages in the log file. The GUI only has the ability to draw, which requires the ability to monitor and execute the modifications of UI. According to these detailed design requirements, it can be divided into three classes.

1. Game Class

   Game Class is mainly the construction of the initial user interface and the new board. Because the layout on the board is to be read by a randomly generated “Level_N.txt” file, this class should check whether this file is existing. If there is no such file, it should create a such file, which contains the step number needed, the target score and a random board, for Board Class. However, the initial board generated randomly may be not suitable for a game level. For example, if there are multiple same colour candies in one row or one column. Thus, it is necessary to check the board to ensure the reasonableness of this initial board. If there are some same colour candies in a row or column, crush them, fill in the new random candies and reset the current score.

2. Board

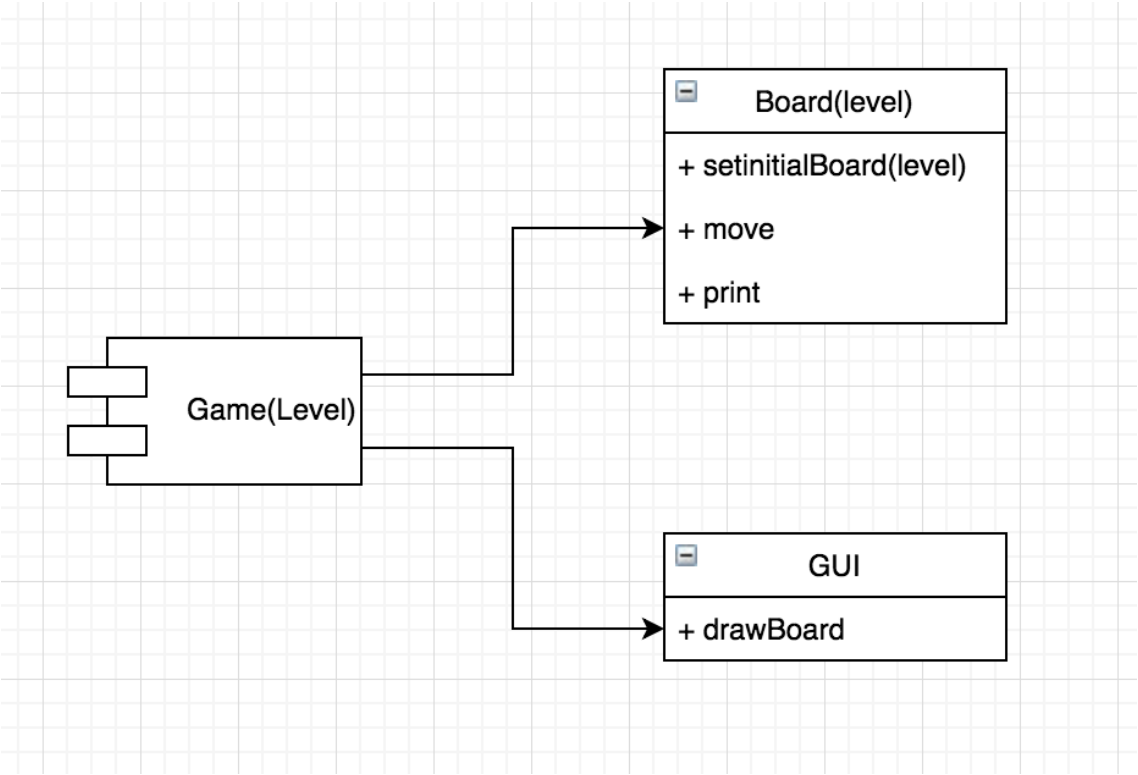

Figure 10: The Architecture of Game. It contains two parts.

From above part mentioned, Board Class should have four significant methods. The first is to read the initial board information and record that. The second method is to swap two selected candies. The third significant method is to crush "Match N" candies. The last method is to add candies to the blank cells.

As for the first method, it is observed form the original "Candy Crush Saga" game that most of the boards are N*M in "Candy Crush Saga" and N and M are usually numbers of <=9 [19]. Consequently, all the boards generated by this application are 9*9 in size. If the designed board is 6*6 in size, the other spare places would be filled with a special candy. This special candy is regarded as a block that this cell has no candy and can not be selected.It is an efficient way to standardize the initial state of all boards. This is not only benefit for generating initial boards, but also good for other methods such as moving and crushing candies. Owing to the board is a 9*9 board in size, it is easy to store data by using a simple 2D-array. Apart from the size of board, the another important matter is to fill candies in the board. Mentioned in the background part of "Candy Crush Saga", there are six colours of regular candies, six colours of striped candies, six colours of wrapped candies, the chocolate candy and the blocks (a special candy). Consequently, the candy can be replaced directly with numbers. For instance, use "0" as the colour number for the yellow candy. In Figure 11, a candy needs a colour number, and the specific position of the board (the row number and the column number). Then

through “extends Candy”, it is easy to create six basic types of candies. “Regular Candy”(named “RegularCandy” in java files) is the first kind and is also the most basic candy category. It contains six different colours and shapes. “Vertical Striped Candy” (named “XCandy” in java files) comes from four same colour candies in a column. Once it crushes, the entire row can be eliminated. The third type of “Horizontal Striped Candy” (named “YCandy” in java files) which is similar to “Vertical Striped Candy”. It comes from four same colour candies in a row, and the entire column to be eliminated once it crushes. The fourth category is “Wrapped Candy” (named “BombCandy” in java files), which can cause the 3*3 cells explosion twice. The fifth one is “chocolate candy” (named “ChocolcateCandy” in java files), which has no colour. However, this category of candies use “-1” as the colour number for uniform format. The last one is not a candy category but it is regarded as a candy category. It has no colour indeed, but I apply “-10” as the colour number for uniform format as well.

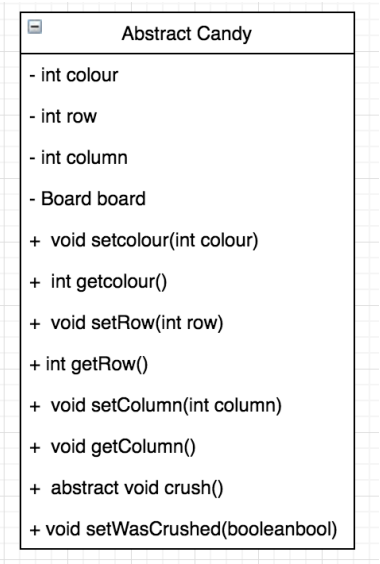

Figure 11: The Class Diagram of Candy. It is an abstract class.

Swapping two selected candies requires several steps as shown in Figure 12:

(a) Select two adjacent candy

(b) Judge whether the movement is valid or not

(c) Swap two candies

(d) Crush candies

From above steps, it is necessary to check six conditions:

(a) Selected candies are not null.

(b) Selected candies cannot be blocks.

(c) Their positions are adjacent.

(d) After the movement, it makes three same colour candies in a row or in a column.

(e) If they are “Striped Candy”, “Wrapped Candy” or “Chocolate Candy”, (d) situation could be ignored.

(f) If one of them is “Chocolate Candy”, (d) and (e) situations can be ignored.

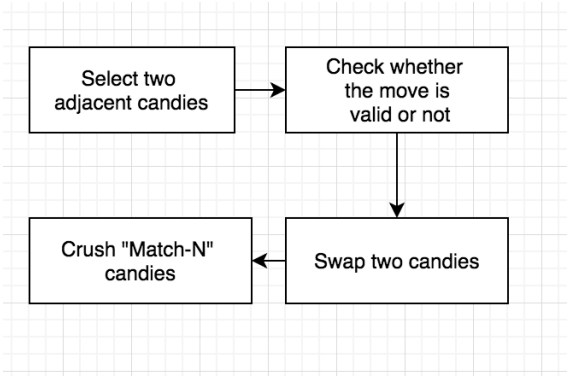

Figure 12: The Process of “Move” method.

The third method is to crush candies. The second method is only to change positions and every movement leads to candies crush, this action should be written separately. Besides, this action makes the current score higher. According to the rules of each candy category, I also design five classes to crush candies. The figure 13 shows the explosion result of each candy crush itself. Besides, the combination of two special candies would lead to huge crush, the procedure code is shown in the figure 14.

The last method in Board Class is to add random candies. It is easy to get a random number from 0 to 5 when applying “(Math.random()*6)” to this function.

3. GUI

If Class GUI “extends JPanel”, then it can change the display of the entire board. Because the board is an 2D-array and the size is 9*9, the GUI display can also be regarded as an 2D-array of 9*9 JButtons. In this way, add a candy image to each JButton and all buttons compose the whole board. Beside, each button needs to be monitored so that players can easily find the selected location and the corresponding candy. In this part, the most significant matter is to recognize the category of each candy. Because one candy, whose colour number is from 0 to 5, can be “RegularCandy”, “Xcandy”, “YCandy” or “BombCandy”.

- **Player Part**

Daniel Hadar & Oren Samuel wrote in their reports that “Stupid Greedy Search”(SGS) and Human Players had the similar ability to solve puzzles. We also can conclude that

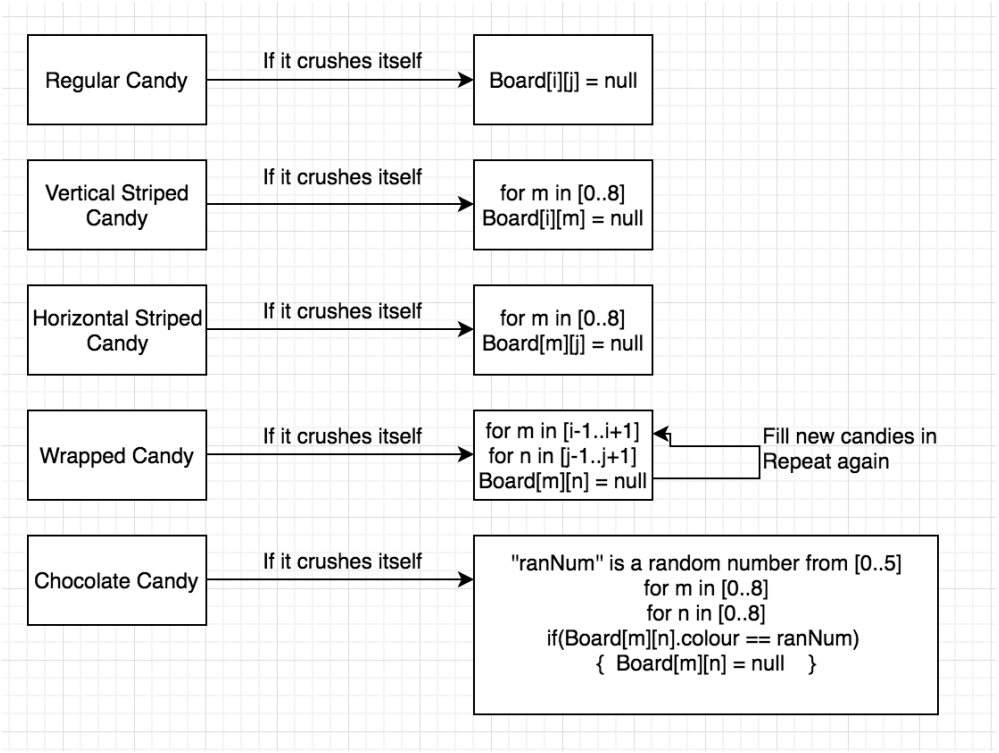

Figure 13: The procedure code of crushing candies themselves.

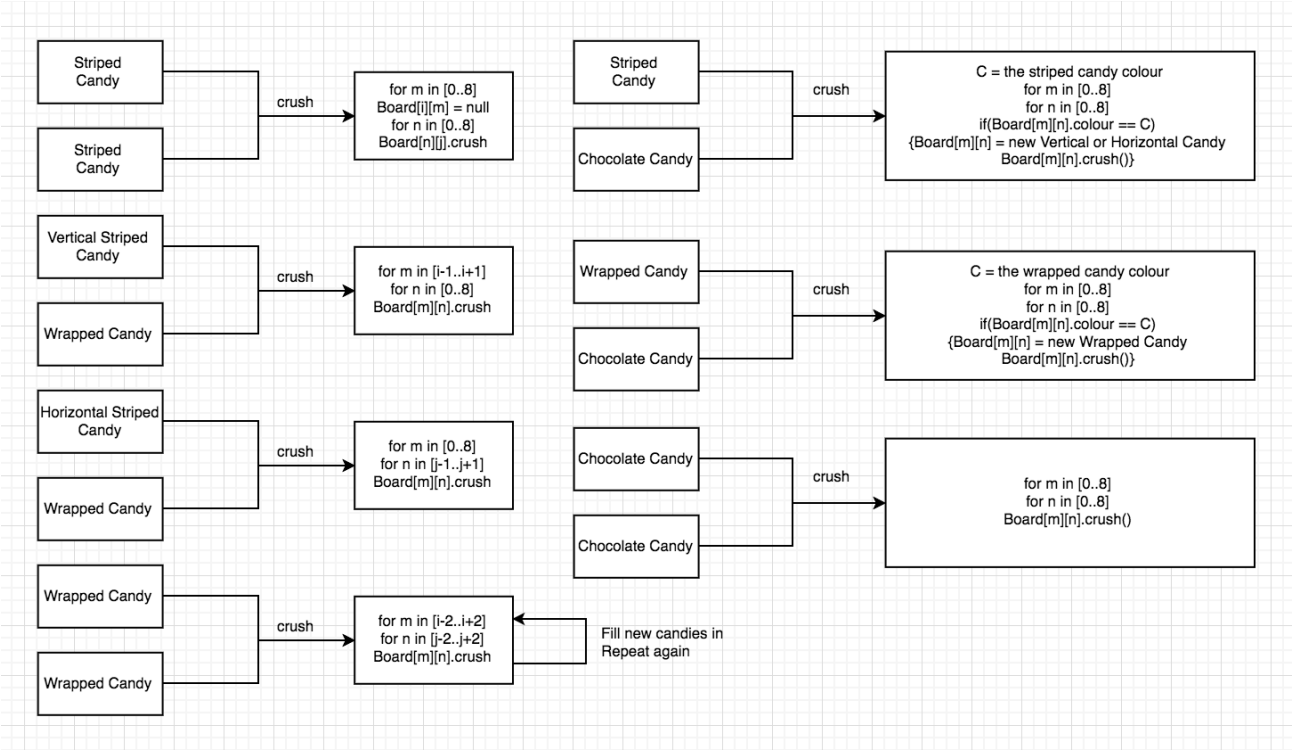

Figure 14: The procedure code of crushing candies combinations.

conclusion from Figure 4. In that article, SGS serves as a good baseline and it gives a good model for the behaviour of human players. If I want to measure difficulty level of each board, it would be a good model for measuring as well. The design of AI player is

to build an AI player using "Stupid Greedy Search" algorithm.

Based on this conclusion, there are four important criteria influencing the SGS best move. The first is the score, which is most basic criteria. It is also related to the final target straight. From my experience on solving "Match Three" puzzles, I find that special candies would have a greatly influence on the final result. Thus, the AI player should have different strategies on choosing candies. Find the highest score or make more special candies are both important for "Candy Crush Saga". Score, the number of chocolate, the number of BombCandy and the total number of XCandy and YCandy are all taking consideration on choosing moves. As a result, there are 24 possible strategies. For example, list a possible strategy which is from the most important criteria to the least, it can be [score, Chocolate Candy number, Wrapped Candy number, Striped Candy number]. Then apply "1" to "4" four integers to represent the importance, [score, Chocolate Candy number, Wrapped Candy number, Striped Candy number] is transformed as [1234]. If this move brings higher score, then choose this move. If several moves bring same scores but one of them creates an Chocolate Candy, choose the move which creates an Chocolate Candy. Other possible strategies: [1243], [1324], [1342], [1423], [1432], [2134], [2143], [2314], [2341], [2413], [2431], [3124], [3142], [3214], [3241], [3412], [3421], [4123], [4132], [4213], [4231], [4312], [4321]. All possible strategies are compared with each other and that one which is most closest to the behaviour by human players will be applied for further implementation.

5.2 Experiment Design

There are four tasks should be completed in this experiment. First, ensure that the application runs well. Second, the initial boards of "Match Three" games can be randomly generated, including the target score and the number of steps required to pass through. Third, each player can complete or abandon each level game within a limited number of steps. Then the application records the time spent and final score obtained. Similarly, the AI player will perform the same tests and record data. Finally, compare the results of human participants and the AI player and calculate the accuracy. Thus, the design of this experiment needs to involve following things:

1. Game Board Design, Selection and Ordering.
2. The number of participants and what they need to train before this experiment.
3. What kind of data should be collected and how to process the collected data.
4. The method to compare the results of human participants and the AI player.

Next, the design of this experiment will be divided into four parts to explain.

5.2.1 Level Design, Selection and Order

As for the number of games involved in this experiments, more data can provide more reliable conclusions. However, participants cannot provide too much time on my experiment. Each game will spend a human player about a few minutes (less than 3 minutes) as I observed. As a result, 10 games for each participants are reasonable. In addition, new players are not familiar about "Candy Crush Saga" rules, so the first few level games should be designed for learning. Thus, I would copy the five games from the five games in real "Candy Crush Saga". Because the first five boards teach players how to create special candies and the results of combination of special candies. In this way, participants are getting familiar with the rules and acquire some skills to play games better. Consequently, the first five games would be thought by participants more difficult than it should be. Thus, I only use the later five games as the test games.

Secondly, some target score for players are too easy that there are not enough "Medium" or "Hard" games among 10 games. For instance, the original targets of the first five score are very easy to complete. Thus, the five targets should be reset so that participants can acquire skills as soon as possible. Besides, other five random boards also should be tested before the experiments.

5.2.2 Participants

Participants are real adult users who are studying in the University of St Andrews. First, they have the great ability to understand rules of a new game well. Secondly, this experiment needs more than 10 participants. Thirdly, the basic rules, such as the combination of special candies, should be taught before the experiment. They also have some chances to try the first five games and then start the experiment to make sure that they have totally understand rules and operations. They can stop or exit this experiment if they want.

5.2.3 Data Collection & Data Processing

The output of this application should be the time spent and the final score of each game. Human players need to record their personal opinions on each level. The majority of their opinions would be regarded as the difficulty level of human players for such game. Then collect these data in a excel file.

Then calculate the win rate of each participant for each game and calculate the time spent on per step. Normalize the win rate and the time spent on per step. Finally, according to this function: final_result = win rate / the time spent on per step. If human players have lower final_result, it represents that this game is more difficult. Otherwise, if human players have higher final_result, it represents that this game is easier. Because AI player does not need user interface but human participants need UI to help, they may have totally different time

spent on average. If the time spent after normalization is still too difficult to compare with each other, only select win rates as the representation of the difficulty level.

5.2.4 Comparison Method

For these data, ranking the data or clustering are good methods to find the difficulty levels for each game. I only need to rank the win rate or the win rate / time spent on per step if I want to obtain the ranking the difficulty levels of games. This method is the best method to know the difficulty levels. If I want to get a difficulty tag for each game, there are two options by using clustering. The first is "K-mean" using R. The final comparison result should be drawn in a table. The x-axis is the win rate / time spent on per step or the win rate only. The y-axis is the id number of each game. The goal of the "K-mean" algorithm is to put n observations in the middle of k clusters so that each observation is placed in the closest cluster to it [28]. "The closest cluster" is measured by the distance between the mean of the corresponding cluster and this observation [28]. Second method is "Support Vector Clustering" [29]. It requires labelled training data so that I can apply the opinions of participants on each levels as the labels [29]. The first method can be clustered without the opinions of participants. The second method has higher accuracy on clustering [30]. Thus, I would apply "Support Vector Clustering" to measure the difficulty level of each board.

6 Implementation & Result

This section describes every important trial on the application and the experiment. Explain the reasons why some design are added, altered or deleted. According to the design of the application and the experiment, the implementation part is also divided into two parts.

6.1 "Candy Crush Saga" Application

The programming language for this project is Java, same as what was used to develop the "Candy Crush Saga" application with the help of Eclipse SDK. This application has two main parts, because the user interface part can be included into Game part.

6.1.1 Game

Because user interface part is included in Game Part and the method to swap candies in Board class also needs the location of the GUI, the designed user interface is implemented with Java Swing toolkit. Thus, Class "Game" and Class "GUI" both reply on Java Swing toolkit.

- **Create and Read board information files**
  Because the first initail boards are copies from the original "Candy Crush Saga" game, I create five "levelN.txt" as the information files. See Figure 15, this file contains required step number, required target and the initial board. Number 0 to 5 stand for the colour of each candy. "block" means block (they also can be represented by -10). Capital letter "C" stands for Chocolate Candy. If there are special candies such as Figure 16, a capital letter will follow the integer. "B" means "BombCandy" (Wrapped Candy), "V" means "XCandy"(Vertical Striped Candy) and "H" means "YCandy"(Horizontal Striped Candy). If players play Level 10, the information file is not created and this application should generate the information file by itself. Firstly, the step number, target score and 9*9 board are random. There are two issues causing that Java random always returns the same number when I set the seed or use "Math.random" directly. The first is that the code sets a seed value for a Random instance, and the second is that the instance method "random" instances a new Random object and then immediately sets its seed with the same seed every time. Combine two issues and then this random method always returns a same sequence values. Thus, the random seed will be changed if the current time is changed. See the method "writeRandomLevel(int LEVEL)" in Figure 17, this seed would related to the step number, target score and the initial board.

- **Game GUI**
  Owing to the design of this application, it should have two main components. The first component is an area for showing the current score, the target score and the number of steps left. The second component is the board area. "JPanel" will separate into two components, a northern area and a southern area. Then in the board area, a new Class "GUI"

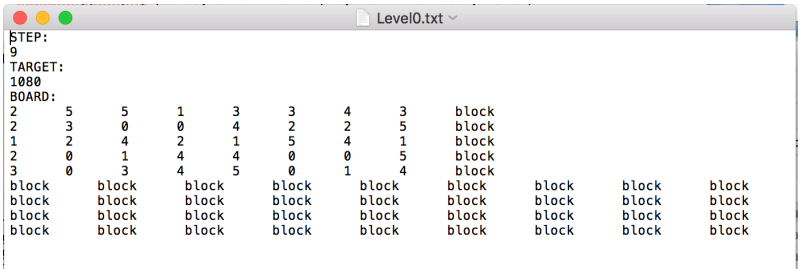

Figure 15: An example screen-shot of a board information file. This is the information of Level 0 (the first game).

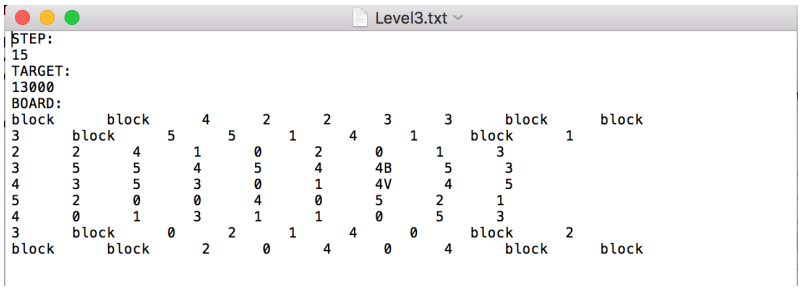

Figure 16: An example screen-shot of a board information file which contains special candies. This is the information of Level 3 (the fourth game).

is developed to update all candies' images. The source of these images is from website: https://github.com/casyazmon/CandyCrush/tree/master/resources/images. What I need is "PNG" files but images downloaded from website are "JPG" files. When I trasnfor "JPG" files into "PNG" files, two images are broken. In this way, "pngcheck" and "pngfix" could help to check broken images and fix them (see Figure 18). Then based on the current board, add candies' images to each area. If this candy is selected, "JButton" is a good choice that players can easily find a rectangle around. After these implementations, the final GUI is shown in Figure 19.

- **Swap Candies On The Board**
  According to the design of the method which swap two adjacent candies on the board, this method is related to other four methods, checking the validation of moves, swapping candies' positions, crushing candies and adding new candies. The first method is to check whether the swap is valid or not. Because if the swapping is valid, candies must be crushed after the swapping. Thus, swap their positions before crushing. Thirdly, according to the basic rules about what the result effect of each candy crushing, write some methods to crush candies. After crushing candies, there are at least two empty cells on the board, this application should add new random candies. I write three random methods for choosing. This random method is not similar to Figure 17. Because I want the filling candies for each board is same, so I use the default seed number rather than use current system time as the seed number.

- **Finish A Game**
  When the step number is equal to 0, we need to check whether players have achieved

```
void writeRandomLevel(int LEVEL) {
    try {
        File writename = new File("Level"+LEVEL+".txt");
        BufferedWriter out = new BufferedWriter(new FileWriter(writename));
        out.write("STEP:" + "\r\n");
        int seed = (int) System.currentTimeMillis();
        Random random = new Random(seed);
        int m = Math.abs(random.nextInt(seed) % 21);
        int a = m+5;
        out.write(a + "\r\n");
        out.write("TARGET:" + "\r\n");
        m = Math.abs(random.nextInt(seed) % 10);
        int b = (m+1)*100*a;
        out.write(b + "\r\n");
        out.write("BOARD:" + "\r\n");
        for(int i = 0; i < BOARDSIZE; i++) {
            for(int j = 0; j < BOARDSIZE; j++) {
                m = Math.abs(random.nextInt(seed) % 7);
                int n = Math.abs(random.nextInt(seed) % seed);
                if(n == (seed-1)) out.write( m + "B" + "      ");
                if(n == (seed-2)) out.write( m + "V" + "      ");
                if(n == (seed-3)) out.write( m + "H" + "      ");
                else if(m == 6) out.write( "block      ");
                else out.write( m + "      ");
            }
            out.write( "\r\n");
        }

        out.flush();
        out.close();

    } catch (Exception e) {
        e.printStackTrace();
    }
}
```

Figure 17: The screen-shot of "writeRandomLevel" method which generates the step number, the target score and a random initial board. Seed number is collected from the current time.

```
cd XXX(filepath)
brew install pngcheck
pngcheck
pngfix XXX(BrokenImageName)
```

Figure 18: How to fix broken "PNG" images in terminal.

the target score. If the player has achieved, the application should open next level game for the player with the notification in Figure 20. Otherwise, the player should play this level again with the notification in Figure 21.

6.1.2 AI player

This AI player should choose two candies to swap and then finish a game. Because SGS which is introduced in Content Survey part is a baseline for human level, I also apply SGS algorithm in my project to predict the difficulty level of each board. Adding random candies will lead to "Combo" in games. Consequently, I repeat each move for 100 times and calculate the average number and then find the highest score. SGS has 24 possible strategies and the final code is shown in Figure 22. 24 possible strategies all will be tested with 5 games which are copied

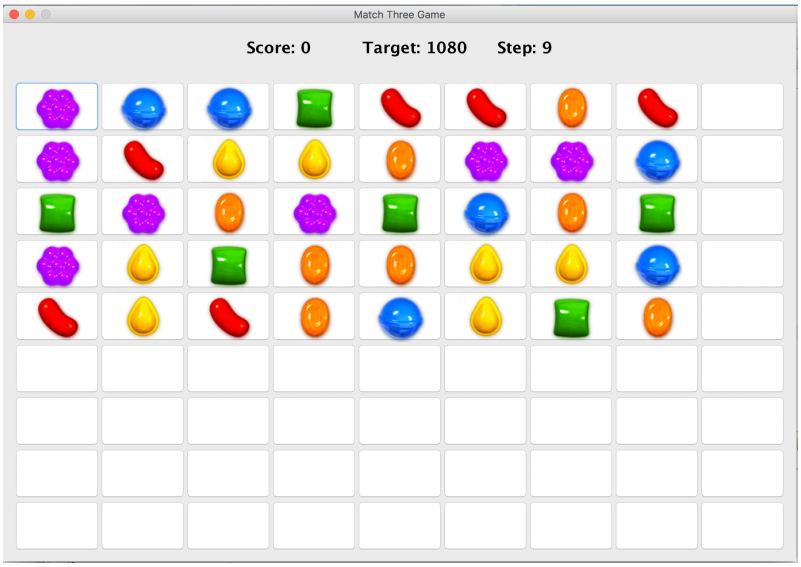

Figure 19: An example screen-shot of the Application. This is the Level 0 screen-shot (the first game).

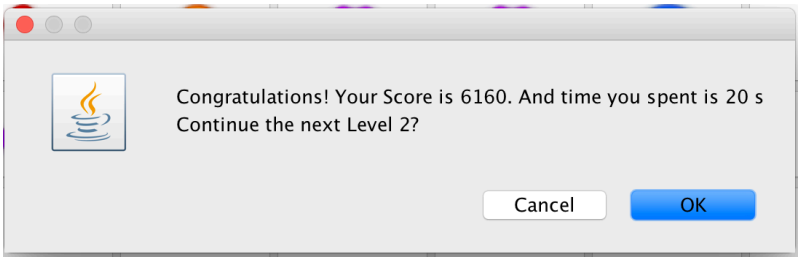

Figure 20: An example screen-shot of the Application. This is the Level 0 screen-shot (the first game).

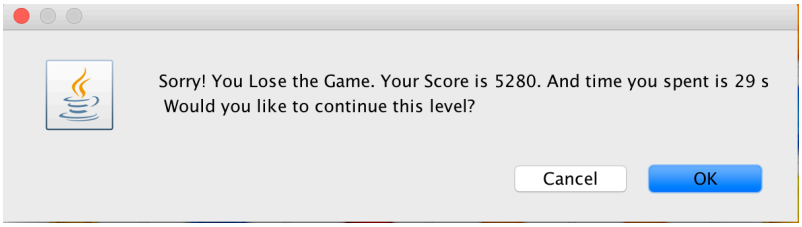

Figure 21: An example screen-shot of the Application. This is the Level 0 screen-shot (the first game).

from the original "Candy Crush Saga" application.

6.2 Experiment

This experiment was completed by students who studying at the University of St Andrews to play 10 games. As the Design Part, there are 13 participants played this application and then record their time spent and final scores in a "log.txt" (see an the example player' "log.txt" in Appendix B). Before this experiment, only one person has not played similar games before

```
//In total, there are 24 methods:
//1234 1243 1324 1342 1423 1432
//2134 2143 2314 2341 2413 2431
//3124 3142 3214 3241 3412 3421
//4123 4132 4213 4231 4312 4321
SGS(game.STRATEGY,  i, j);
```

Figure 22: The screen-shot of choosing different strategies by using SGS algorithm.

and I speed several minutes and a few games to make sure that this player understands rules and learns the basic operations well. Each participants should write their opinion about the difficulty level of each level game ("Easy", "Medium" or "Hard"). A my own experience, it should be 30 minutes or so to complete whole experiment for each participant because they can give some hard games up. However, most participants chose to play multiple times for passing all games. Thus, the total time spent by 13 participants on my experiment is 11.5 hours and thus each participant spent 52 minutes on average. Especially, one participant had spent 2.5 hours on solving 10 puzzles. Their performance helps this experiment greatly from the aspect of data collection.

6.3 Results

After collecting all data from participants, an Excel file records all information together (see Figure 23), including their player id, win rate and total time spent on each level.

Level/Names	Player1	Player2	Player3	Player4	Player5	Player6	Player7	Player8	Player9	Player10	Player11	Player12	Player13
0	Easy	Medium	Medium	Easy	Easy	Easy	Easy	Easy	Easy	Easy	Easy	Easy	Easy
1	Easy	Easy	Medium	Easy	Easy	Easy	Easy	Easy	Easy	Easy	Medium	Easy	Medium
2	Easy	Hard	Hard	Hard	Medium	Medium	Easy	Medium	Medium	Easy	Hard	Hard	Hard
3	Medium	Medium	Medium	Medium	Easy	Easy	Hard	Easy	Medium	Easy	Medium	Easy	Medium
4	Easy	Medium	Medium	Easy	Medium	Easy	Easy	Easy	Easy	Easy	Easy	Medium	Easy
5	Medium	Hard	Hard	Hard	Medium	Easy	Easy	Hard	Easy	Easy	Easy	Hard	Easy
6	Easy	Hard	Medium	Hard	Easy	Easy	Medium	Easy	Medium	Easy	Easy	Easy	Medium
7	Hard	Medium	Hard	Hard	Medium	Medium	Hard	Hard	Hard	Hard	Hard	Hard	Hard
8	Easy	Easy	Hard	Easy	Hard	Medium	Easy	Easy	Easy	Easy	Easy	Easy	Easy
9	Medium	Hard	Medium	Medium	Easy	Easy	Easy	Easy	Medium	Easy	Easy	Medium	Easy
win rate													
0	1	1	1	1	0.5	1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1	1	1	1	1	1	0.333333333
2	1	0.25	0.5	0	0.333333333	0.043478261	0	0.333333333	0.5	1	0.166666667	0.142857143	0
3	0.333333333	1	1	0.5	0.5	1	1	0.5	0.25	1	0.333333333	1	0.333333333
4	1	1	1	1	0.333333333	1	1	1	1	1	1	0.5	1
5	1	0.142857143	0	0	1	0.166666667	0.5	0.142857143	0.5	1	1	0.1	0.5
6	1	0.2	1	0	0.5	0.5	0.5	0.5	0.25	1	0.5	1	0.333333333
7	0	1	0	0	0.142857143	1	0.2	0.25	0.041666667	0.166666667	0	0	0
8	0.5	1	0	1	1	0.5	1	1	1	1	1	1	1
9	1	0	0.5	0	0.5	1	1	1	0.333333333	1	1	0.5	1
time total(s)													
0	54	52	95	103	82	51	64	47	31	40	46	82	40
1	119	62	167	111	46	66	72	69	90	51	193	172	232
2	76	358	516	481	166	1905	488	413	229	54	1206	1270	1523
3	301	93	178	304	121	115	61	158	343	44	1674	131	367
4	124	128	280	285	289	194	64	83	114	46	253	280	157
5	55	237	99	712	28	268	46	363	76	17	271	573	114
6	55	340	78	575	98	128	126	112	229	34	194	107	278
7	548	99	241	253	709	100	567	549	2531	340	921	5232	1160
8	92	84	91	158	51	148	48	63	80	26	93	143	48
9	73	119	91	99	149	76	67	77	242	28	140	193	130

Figure 23: Participants' Experiment Data: The opinions towards each game, win rate and total time spent of each game.

It should be noticed that files would record 10 games but only the last five games are used because of this experiment design. Human participants have different ability and strategies on solving puzzles. As a result, some participants spend much longer time on solving puzzles but the win rate is high. Even after normalizing human participants' data, the time spent

still has too much large difference. Thus, in the later experiment, I will only use "win rate" as the only criteria of games' difficulty level. From this table, we can collect the major opinion among 13 participants and the histogram diagram show the result (see Figure 24). Then calculate a function to combine all attitudes of each game. However, the ranking from this figure can not be concluded directly. Thus, there are two options for ranking the difficulty level. The first is to compare the average win rate. The final ranking from the easiest to the hardest is 4, 3, 5, 1, 2 (see Table 4).

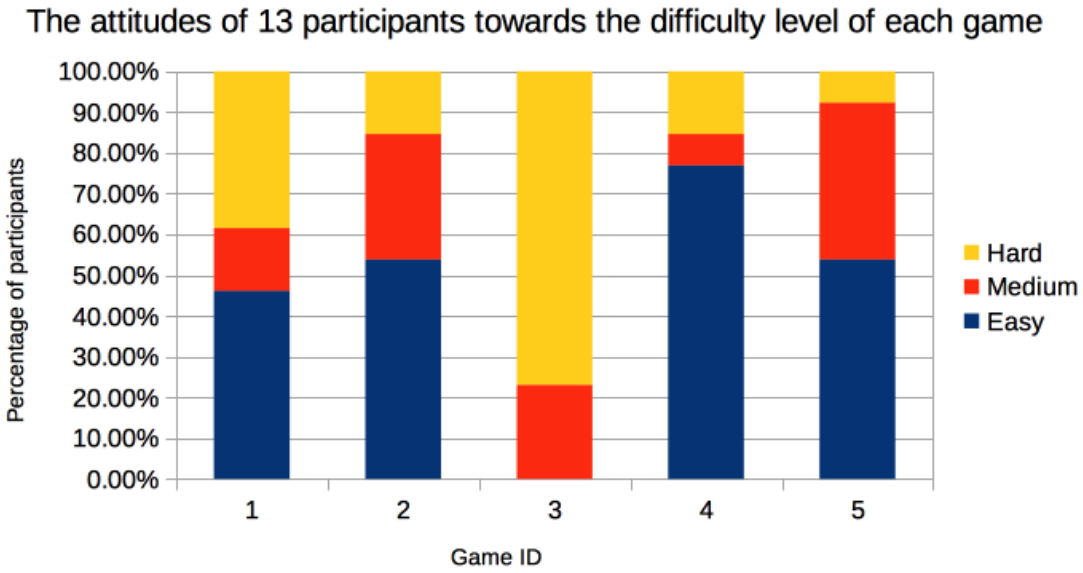

Figure 24: The attitudes of 13 participants towards the difficulty level of each game. The major attitudes are "Easy", "Easy", "Hard", "Easy", "Easy".

Game ID	The Average of 13 participants' Win rate(2 decimal places)	Ranking
1	0.47	4
2	0.56	3
3	0.22	5
4	0.85	1
5	0.68	2

Table 4: The Ranking of difficult level from the easiest to the hardest through comparing the average win rate of 13 participants. The ranking of difficulty level from the aspect of major attitudes (from the easiest to the hardest) is 2, 3, 5, 1, 4.

Secondly, apply different weights to three classes and get the sum. In this way, I suppose that the weight of "Easy" is 0, "Medium" is 0.5 and "Hard" is 1. The final results are 6, 4, 11.5, 2.5, 3.5 for Game 1 to Game 5 (see Table 5). It can be concluded from the results that the difficulty level ranking of major attitudes is 4, 3, 5, 1, 2 (from the easiest to the hardest). Compare the ranking of two tables, the results are totally same so that I could use the ranking

Game ID	“Easy”	“Medium”	“Hard”	Sum	Ranking
1	6	2	5	6	4
2	7	4	2	4	3
3	0	3	10	11.5	5
4	10	1	2	2.5	1
5	7	5	1	3.5	2

Table 5: The ranking of difficult level through comparing the sum after weighting three difficulty classes. The weight of “Easy” is 0, “Medium” is 0.5 and “Hard” is 1. The ranking of difficulty level from the aspect of major attitudes (from the easiest to the hardest) is 2, 3, 5, 1, 4.

to represent the difficulty level directly. It indicates that 13 participants are carefully and fairly evaluating each game which also adds credibility to the following experiment.

Then AI player should choose its strategy among 24 possible methods. All data are collected in an Excel file (see Figure 25). Win-rates of 24 AI players should be compared with the average win-rates of 13 participants and the results are shown in Figure 26.

STRATEGY/winrate	1	2	3	4	5
1234	0.92	0.86	0.33	1	0.22
1243	0.97	0.92	0.3	1	0.26
1324	0.97	0.86	0.29	1	0.28
1342	0.95	0.91	0.34	1	0.22
1423	0.96	0.88	0.21	1	0.25
1432	0.97	0.9	0.28	1	0.28
2134	0.96	0.78	0.12	0.36	0.93
2143	0.96	0.78	0.12	0.39	0.92
2314	0.96	0.71	0.14	0.26	0.33
2341	0.91	0.9	0.1	1	0.58
2413	0.86	0.68	0.12	0.24	0.29
2431	0.88	0.84	0.03	1	0.55
3124	0.93	0.78	0.11	0.36	0.89
3142	0.82	0.77	0.07	0.31	0.93
3214	0.93	0.73	0.14	0.1	0.26
3241	0.89	0.83	0.06	1	0.44
3412	0.87	0.71	0.15	0.16	0.18
3421	0.9	0.8	0.08	1	0.57
4123	0.22	0.29	0.39	0.32	0.25
4132	0.24	0.31	0.24	0.23	0.28
4213	0.22	0.22	0.17	0.16	0.07
4231	0.14	0.18	0.09	0.1	0.1
4312	0.88	0.6	0.22	0.26	0.19
4321	0.84	0.83	0.07	1	0.51
human	0.96	0.95	0.33	0.67	0.91

Figure 25: The win-rates of each game by 24 different AI players and the average win rates of 13 human players.

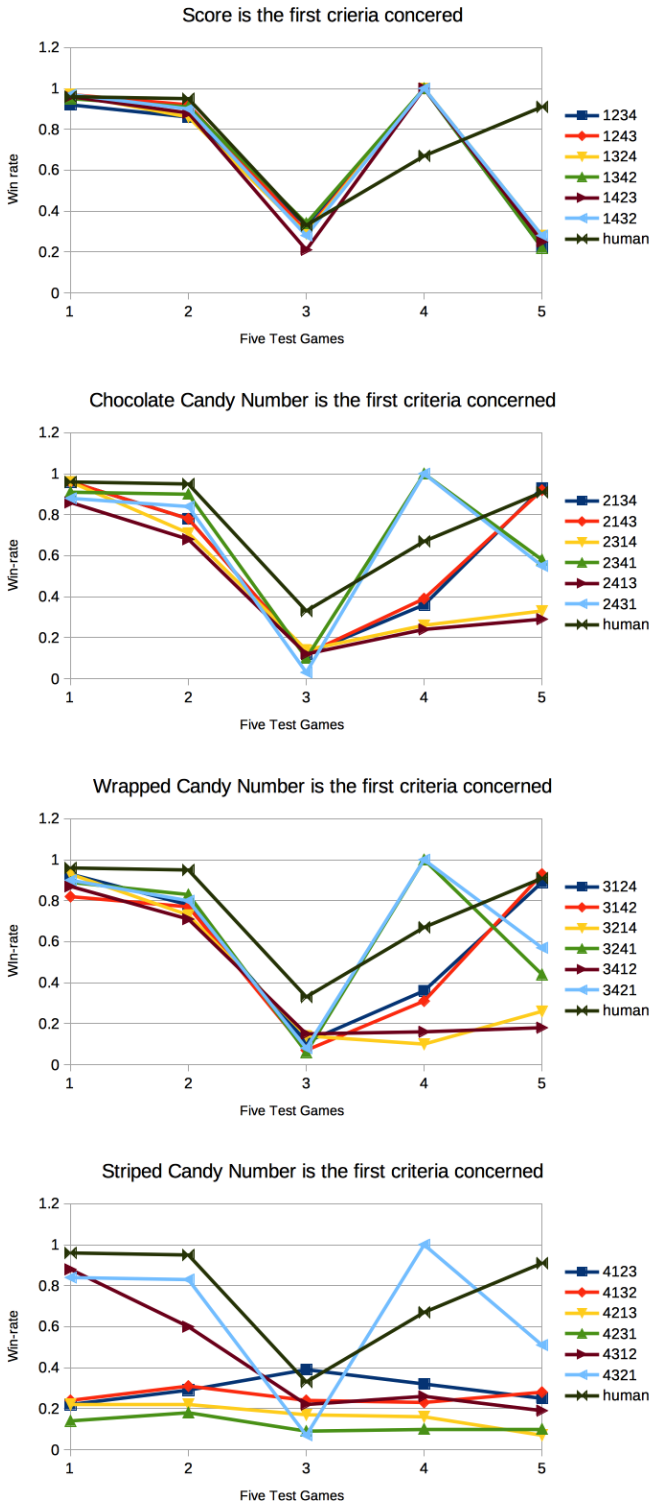

Figure 26: The trends of the win-rates among 24 different AI players and 13 human players.

From four figures, it can be found that [2134], [2143], [3124] and [3142] have the most similar trend lines compared to the human trend line. After calculating the differences by standard deviation formula, the smallest one is [2143] and [2134] is very closest to the result as well. Thus, AI player should choose [2143] as its strategy and completing later experiments. This result also represents one possible conclusion that most human players concern "Chocolate Candy" first and then concern the total score second.

Then repeat same games for AI player for 100 times and also record its win rate of each level. In addition, AI player does not need GUI part which saves a lot of time so that this experiment only spends a few minutes. The result of win rate played by AI player is shown in in Figure 27. From this figure, the ranking of difficulty level is 4, 3, 5, 1, 2 (from the easiest to the hardest). This ranking is same to that in participants' experiment. Overall, this application can measure the difficulty level.

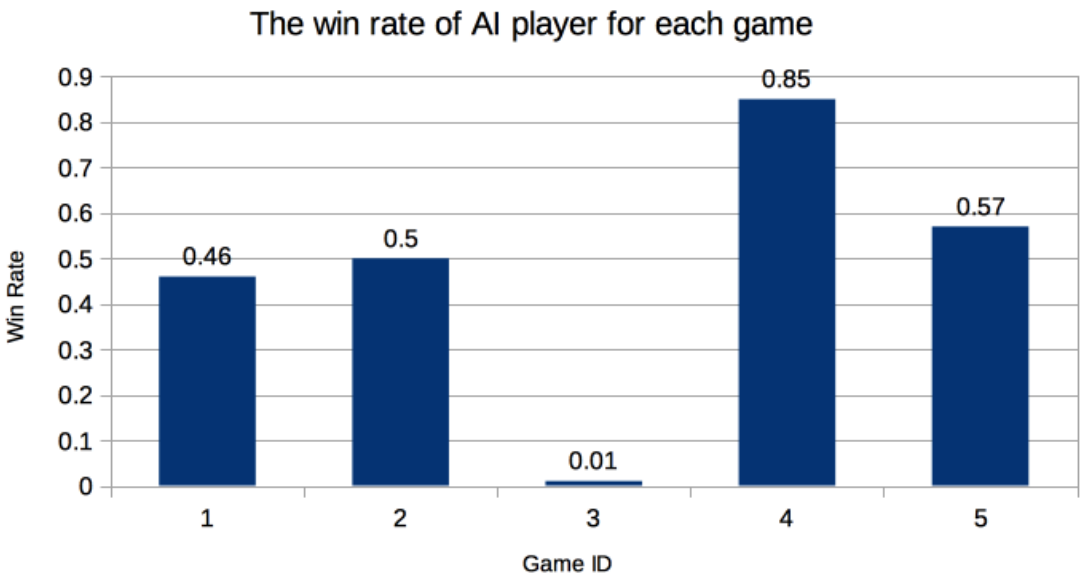

Figure 27: The win rate of AI player for each game. Each Game has been played for 100 times.

Apart from the ranking of difficulty level, I also divide these games into three classes, "Easy", "Medium" and "Hard". The win rates of each participants for each levels are applied as training data and their opinions are applied as the training labels. In order to decrease individual differences, the average win rate is significant for this clustering experiment. Then take the average win rates of human players and the win rates of AI player for each levels as the test data and use them all in SVC to divide ten games into three difficulty levels. When completing the clustering, the test label is shown as Figure 28. The first five numbers belong to human participants and the following five numbers belong to AI player. According to the result, AI player thinks that the difficulty level ranking of five games are shown in Figure .

Thus, the accuracy of the measured difficulty level classes of the AI player compared to the major attitudes of 13 participants for 5 games is 80%. Only the last game is different and this result also can be seen that this application can measure the difficulty level.

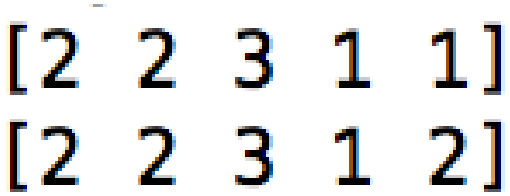

Figure 28: The result of AI player after applying SVC. “1” stands for “Easy”, “2” stand for “Medium” and “3” stands for “Hard”.

7 Evaluation

The design of this application has been implemented successfully that this application has been achieved three aims of this project, including generating puzzles, solving puzzles by AI and Human players and measuring the difficulty level. As what was planned in Chapter 3 Requirements specification, all the functional and non-functional features have been satisfied and the application has fulfilled the five usability quality components. In addition, this application can be run in all platforms such as Windows, Mac OSX and Linux. Human players can easily control candies in the screen through the Mouse and play “Candy Crush Saga” games endlessly because this application can generate random boards if the player wants to play. Besides, the AI player finds the best move based on “Stupid Greedy Search” algorithm and finally calculates the win rate. According to the experiments on 13 participants and the help of “SVC”, it is easy to measure the difficulty level of each game.

However, compared to other similar studies on the experiments and how to measure the difficulty level of “Candy Crush Saga”, my project has a huge difference from their methods. For example, this experiment has 13 participants and each participant has to play 10 games. Then compare the difficulty level by the average win rates or weighting the amounts of three difficulty level classes. Finally, collect win rates and the attitude towards each game as training data and the average win rates as the final result. In this way, this application can measure the difficulty level. While Daniel Hadar & Oren Samuel did a similar test on their application which has only 5 human subjects [22]. Then they compare the average swap count when using different heuristics. They provide many possible heuristics for my AI player, and I choose “SGS” which result is the most closest to human result. In another paper, Purmonen SamiWe explored the usage of Monte Carlo tree search (MCTS) and deep learning on predicting game level difficulty in “Candy Crush Saga” by the win rate [31]. He also programmed a bot for a simplified “Candy Crush Saga” application. However, he applied a deep neural network (DNN) or MCTS to predict the following moves and the DNN can make estimations of game level difficulty[31]. He found that DNN could predict the difficulty level in substantially

shorter time if there are enough levels training. The The standard of difficulty is based on the accuracy of predicted attempts per success for each bot. Then he tested three methods by playing multiple times as figure 29. Finally, Purmonen SamiWe concluded that DNN could predict the difficulty level. However, there is no standard or any output for each difficulty level which is providing by Purmonen SamiWe. These works are very similar to my project but we have different strategies and different aims. Overall, my project complete all aims even through this AI player is much weaker than the bot using MCTS and DNN .

Method	Attempts	Mean absolute error	Standard deviation
MCTS	100	3.02	4.12
MCTS + DNN	100	2.96	4.06
DNN	100	3.28	4.41
DNN	1000	2.85	3.85

Figure 29: Accuracy of predicted attempts per success for each bot.

8 Conclusions and Further Work

To summarized, this project has developed a “Candy Crush Saga” application which could generate puzzles directly. Human and AI players both could play games through this application. In addition, AI player could measure the difficulty level. The ranking of difficulty level conducted by the average win rates of 13 participants is same to the ranking measured by AI player. Or the ranking of the difficulty level after weighting the amount of three difficulty level classes “Easy”, “Medium” and “Hard” is also same to the ranking measured by AI player. Besides, win rates and these attitudes of each participant are used as training data and then the average win rate of participants and the win rates of AI player are used as the test data. After clustering data, the final prediction of AI player is 80% accuracy compared to the average attitudes of 13 participants towards 5 games. Overall, this project has completed three main aims have been completed. However, this project still has some problems and some further work to do. The first is that the AI player cannot predict the best moves well. On the one hand, there is not enough training data, which represents the amount of participants, that “SVC” may be not trained well enough. Secondly, there is no standard for the difficulty levels. Different people have different opinions. Thirdly, this experiment does not consider learning transfer because of the order of five games. Each participant played 5 games in a same order and this may be affected by learning transfer. Thus, the order should be the interleaved problem order. For example, Player A plays five games by the order: 1, 2, 3, 4, 5. Then Player B plays five games by the order: 2, 3, 4, 5, 1.

References

[1] Katherine Isbister and Noah Schaffer. *Game usability: Advancing the player experience*. CRC Press, 2008.

[2] Inês Lynce and Joël Ouaknine. Sudoku as a sat problem. In *ISAIM*, 2006.

[3] Helmut Simonis. Sudoku as a constraint problem. In *CP Workshop on modeling and reformulating Constraint Satisfaction Problems*, volume 12, pages 13–27. Citeseer, 2005.

[4] David Silver, Julian Schrittwieser, Karen Simonyan, Ioannis Antonoglou, Aja Huang, Arthur Guez, Thomas Hubert, Lucas Baker, Matthew Lai, Adrian Bolton, et al. Mastering the game of go without human knowledge. *Nature*, 550(7676):354, 2017.

[5] Brian Schwab. *AI game engine programming*. Nelson Education, 2009.

[6] Stuart Russell, Peter Norvig, and Artificial Intelligence. A modern approach. *Artificial Intelligence. Prentice-Hall, Egnlewood Cliffs*, 25(27):79–80, 1995.

[7] Lennart Nacke, Anders Drachen, Kai Kuikkaniemi, Joerg Niesenhaus, Hannu J Korhonen, Wouter M Hoogen, Karolien Poels, Wijnand A IJsselsteijn, and Yvonne AW De Kort. Playability and player experience research. In *Proceedings of DiGRA 2009: Breaking new ground: Innovation in games, play, practice and theory*. DiGRA, 2009.

[8] Jesper Juul. Swap adjacent gems to make sets of three: A history of matching tile games. *Artifact*, 1(4):205–216, 2007.

[9] Cheng Chen and Louis Leung. Are you addicted to candy crush saga? an exploratory study linking psychological factors to mobile social game addiction. *Telematics and Informatics*, 33(4):1155–1166, 2016.

[10] WSJ Staff. Why 15 million people are addicted to candy crush. Available at: https://blogs.wsj.com/digits/2013/05/22/why-15-million-people-are-addicted-to-candy-crush/. Accessed July 28, 2018.

[11] Michael Thomsen. How should a cloned app be? the ethics of artistry in an age of exploitation. Available at: https://www.forbes.com/sites/michaelthomsen/2014/04/08/how-should-a-cloned-app-be-the-ethics-of-artistry-in-an-age-of-exploitation/#b993d2c14ede. Accessed July 29, 2018.

[12] Nintendo. Panel de pon. Available at: http://nintendo.wikia.com/wiki/Panel_de_Pon. Accessed July 29, 2018.

[13] Nintendo. Tetris attack. Available at: http://nintendo.wikia.com/wiki/Tetris_Attack. Accessed July 29, 2018.

[14] Nintendo. Pokémon puzzle league. Available at: https://www.nintendo.com/games/detail/ptXgPEzxomPG7ko1NYa5UnjSb4291Nn-#game-info. Accessed July 29, 2018.

[15] Mark Ward. Casual games make a serious impact. Available at: http://news.bbc.co.uk/1/hi/technology/7301374.stm. Accessed July 28, 2018.

[16] PopCap. Bejeweled stars. Available at: https://www.ea.com/games/bejeweled/bejeweled-stars?isLocalized=true. Accessed July 28, 2018.

[17] Evangeline Marlos Varonis and Maria Evangeline Varonis. Deconstructing candy crush: what instructional design can learn from game design. *The International Journal of Information and Learning Technology*, 32(3):150–164, 2015.

[18] Morven Leese Brian S. Everitt, Sabine Landau and Daniel Stahl. *Cluster Analysis, 5th Edition*. Wiley Series in Probability and Statistics, 2011.

[19] Fandom. Candies in candy crush saga. Available at: http://candycrush.wikia.com/wiki/Candies. Accessed July 29, 2018.

[20] Fred Richmond. Why you can't stop playing candy crush. Available at: https://www.stjhs.org/healthcalling/2015/august/why-you-cant-stop-playing-candy-crush/. Accessed July 29, 2018.

[21] King Digital. The world' s most popular game candy crush saga launches on 'kakao talk' in korea. Available at: https://discover.king.com/wp-content/uploads/2016/05/091313__the_worlds_most_popular_game_candy_crush_saga_launches_on_kakao_talk_in_korea-1-.pdf. Accessed July 29, 2018.

[22] Hadar Daniel and Samuel Oren. Crushing candy crush - an ai project. Available at: http://www.cs.huji.ac.il/~ai/projects/2014/crushingCandyCrush/report.pdf. Accessed July 29, 2018.

[23] Elisa D Mekler, Julia Ayumi Bopp, Alexandre N Tuch, and Klaus Opwis. A systematic review of quantitative studies on the enjoyment of digital entertainment games. In *Proceedings of the 32nd annual ACM conference on Human factors in computing systems*, pages 927–936. ACM, 2014.

[24] Su Xue, Meng Wu, John Kolen, Navid Aghdaie, and Kazi A Zaman. Dynamic difficulty adjustment for maximized engagement in digital games. In *Proceedings of the 26th International Conference on World Wide Web Companion*, pages 465–471. International World Wide Web Conferences Steering Committee, 2017.

[25] 偶尔喝咖啡. 三消游戏关卡设计教程（初级篇）——基本地形设计. Available at: http://youxiputao.com/articles/5465. Accessed July 29, 2018.

[26] Christopher Jefferson, Wendy Moncur, and Karen E Petrie. Combination: Automated generation of puzzles with constraints. In *Proceedings of the 2011 ACM Symposium on Applied Computing*, pages 907–912. ACM, 2011.

[27] Jakob Nielsen. Usability 101: Introduction to usability, 2003.

[28] Kiri Wagstaff, Claire Cardie, Seth Rogers, Stefan Schrödl, et al. Constrained k-means clustering with background knowledge. In *ICML*, volume 1, pages 577–584, 2001.

[29] Asa Ben-Hur, David Horn, Hava T Siegelmann, and Vladimir Vapnik. Support vector clustering. *Journal of machine learning research*, 2(Dec):125–137, 2001.

[30] Marina Marino and Cristina Tortora. A comparison between k-means and support vector clustering of categorical data. *Statistica applicata*, 21(1):5–16, 2009.

[31] Sami Purmonen. Predicting game level difficulty using deep neural networks, 2017.

A Ethical Approval Document

University Teaching and Research Ethics Committee

12 August 2018

Dear Han Bao,

Thank you for submitting your ethical application, which was considered by the School of Computer Science Ethics Committee on Wednesday 6th June, where the following documents were reviewed:

1. Ethical Application Form
2. Participant Information Sheet
3. Anonymous Consent Form

The School of Computer Science Ethics Committee has been delegated to act on behalf of the University Teaching and Research Ethics Committee (UTREC) and has granted this application ethical approval. The particulars relating to the approved project are as follows -

Approval Code:	CS13784	**Approved on:**	26.06.18	**Approval Expiry:**	26.06.2023
Project Title:	Automated Creation of Puzzle games with Constraint Programming				
Researcher(s):	Han Bao				
Supervisor(s):	Chris Jefferson				

Approval is awarded for five years. Projects which have not commenced within two years of approval must be re-submitted for review by your School Ethics Committee. If you are unable to complete your research within the five year approval period, you are required to write to your School Ethics Committee Convener to request a discretionary extension of no greater than 6 months or to re-apply if directed to do so, and you should inform your School Ethics Committee when your project reaches completion.

If you make any changes to the project outlined in your approved ethical application form, you should inform your supervisor and seek advice on the ethical implications of those changes from the School Ethics Convener who may advise you to complete and submit an ethical amendment form for review.

Any adverse incident which occurs during the course of conducting your research must be reported immediately to the School Ethics Committee who will advise you on the appropriate action to be taken.

Approval is given on the understanding that you conduct your research as outlined in your application and in compliance with UTREC Guidelines and Policies (http://www.st-andrews.ac.uk/utrec/guidelinespolicies/). You are also advised to ensure that you procure and handle your research data within the provisions of the Data Provision Act 1998 and in accordance with any conditions of funding incumbent upon you.

Yours sincerely

Wendy Boyter

School Ethics Committee Administrator

B An Example Record for Each Player

```
1LEVEL 0 START!
TIME: 1533668951361
LEVEL 0 FINISH!
TIME: 1533669005599
Score 3480     RUNNING TIME: 54 s

LEVEL 1 START!
TIME: 1533669010250
LEVEL 1 FINISH!
TIME: 1533669130146
Score 8580     RUNNING TIME: 119 s

LEVEL 2 START!
TIME: 1533669143499
LEVEL 2 FINISH!
TIME: 1533669220106
Score 19940     RUNNING TIME: 76 s

LEVEL 3 START!
TIME: 1533669224810
LEVEL 3 FINISH!
TIME: 1533669322938
Score 12800     RUNNING TIME: 98 s

LEVEL 3 START!
TIME: 1533669332184
LEVEL 3 FINISH!
TIME: 1533669429870
Score 11100     RUNNING TIME: 97 s

LEVEL 3 START!
TIME: 1533669431845
LEVEL 3 FINISH!
TIME: 1533669538301
Score 26120     RUNNING TIME: 106 s

LEVEL 4 START!
TIME: 1533669541426
LEVEL 4 FINISH!
TIME: 1533669665661
Score 9680     RUNNING TIME: 124 s
```

```
LEVEL 5 START!
TIME: 1533669667883
LEVEL 5 FINISH!
TIME: 1533669723505
Score 3140     RUNNING TIME: 55 s

LEVEL 6 START!
TIME: 1533669729511
LEVEL 6 FINISH!
TIME: 1533669785509
Score 8420     RUNNING TIME: 55 s

LEVEL 7 START!
TIME: 1533669788200
LEVEL 7 FINISH!
TIME: 1533669932355
Score 4080     RUNNING TIME: 144 s

LEVEL 7 START!
TIME: 1533669936992
LEVEL 7 FINISH!
TIME: 1533670031438
Score 3480     RUNNING TIME: 94 s

LEVEL 7 START!
TIME: 1533670474774
LEVEL 7 FINISH!
TIME: 1533670625312
Score 13600     RUNNING TIME: 150 s

LEVEL 7 START!
TIME: 1533670694458
LEVEL 7 FINISH!
TIME: 1533670533915
Score 4000     RUNNING TIME: 160 s

LEVEL 8 START!
TIME: 1533670586121
LEVEL 8 FINISH!
TIME: 1533670629357
Score 1860     RUNNING TIME: 43 s
```

```
LEVEL 8 START!
TIME: 1533670631117
LEVEL 8 FINISH!
TIME: 1533670680984
Score 2880     RUNNING TIME: 49 s

LEVEL 9 START!
TIME: 1533670900225
LEVEL 9 FINISH!
TIME: 1533670974161
Score 12260     RUNNING TIME: 73 s
```