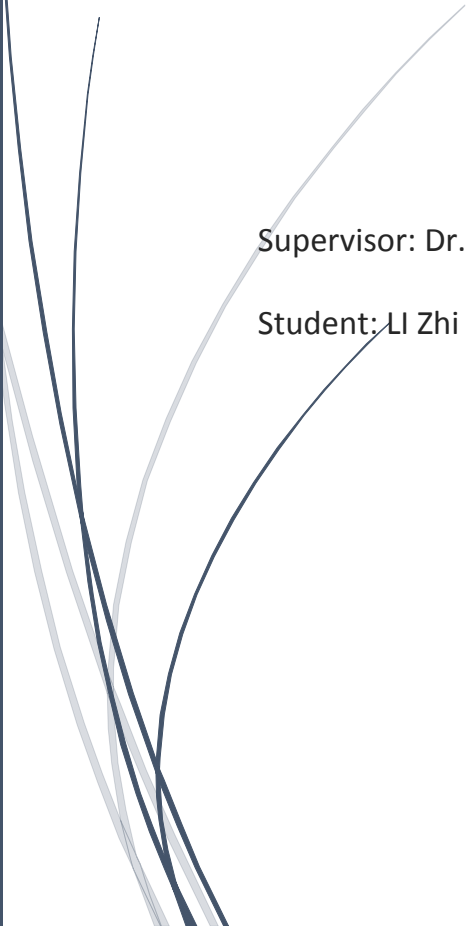




2014 spring semester

CSIT6910 Independent Project Report

—Implementation of Snake and its AI



Supervisor: Dr. David Rossiter

Student: LI Zhi

Table of Contents

ABSTRACT	2
1. INTRODUCTION	3
2. GAME DESIGN AND IMPLEMENTATION	5
2.1 Elements and their roles	5
2.2 Rules	8
3. AI DESIGN AND IMPLEMENTATION	9
3.1. Requirements	9
3.2. Ideas	9
3.3. Implementation	13
4. POSSIBLE FURTHER WORKS	17
5. SUMMARY	19
Appendix	20
Meeting Minutes	20

ABSTRACT

Traditional snake is a video game, which is easy to play. The rough idea is the player controls a 'snake' in the board to eat 'foods'. It is pretty interesting and owns a large number of fans. To make it accessible to most player, we implement it as a web game, so that at any time the game player just need a browser.

While, sometimes it could be quite boring if the player play a game without a competitor. So, to make the game more exciting, we add another 'snake' to the board, which is under control of computer.

Nevertheless, the game becomes harder and harder with the 'snake' keeps on growing, therefore a sophisticated artificial intelligence is required to make the computer more powerful, so that it can compete with the human beings. There are numbers of questions that we have to consider to design a nice artificial intelligence. For instance, does the shortest path to the food always means the best solution? We will

discuss these questions later on.

So, this project aims to implement the game – Snake, with two modes: single player mode and player VS AI mode. The artificial intelligence of the game is the most complex part of this project. It involves graph theory, gaming theory, decision tree and so on.

1. INTRODUCTION

Snake is a video game which originated during the late 1970s in arcades. To be precise, it is a concept, without details. Therefore there is no definitive version of the game. As a result, there are various versions of implementations for this game. Figure 1 shows a version of Snake. After it became the standard pre-loaded game on Nokia mobile phones in 1998, there was a resurgence of interest in the game as it found a larger audience. Thus, the Nokia version is used as the reference for the design of our Snake.

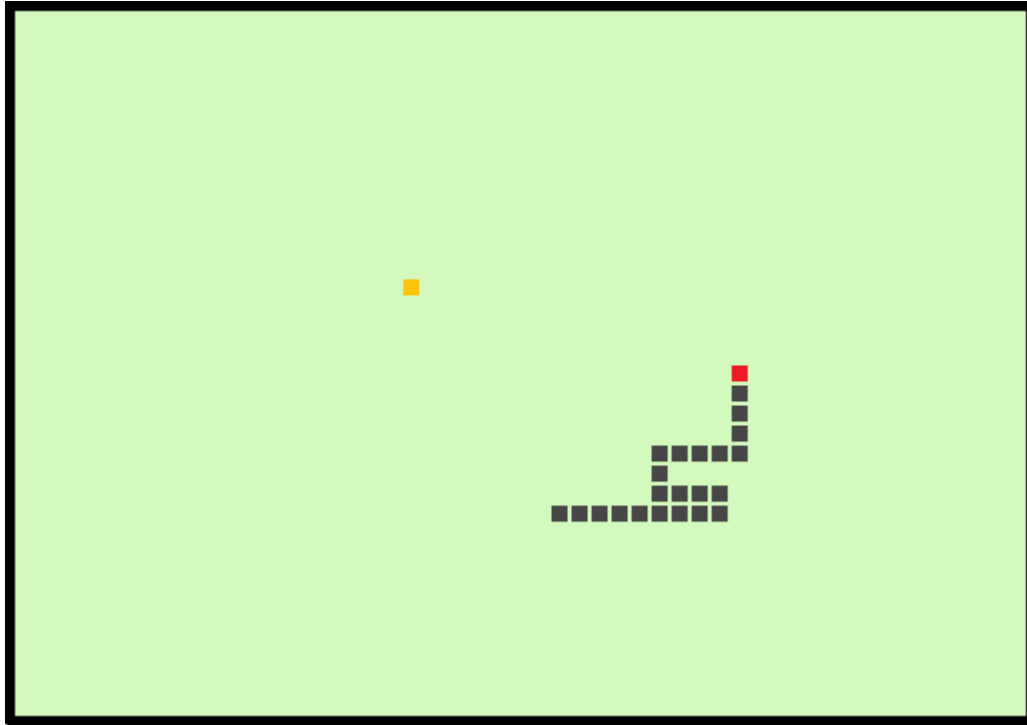


Figure 1: One version of Snake

With a quick glance, most people would take this game as a simple game, which does not require sophisticated strategies. Admittedly, it is quite easy for the player at earlier stage, as paths are obvious and obstacles are few. While, it does require advanced strategies when the ‘snake’ is long enough, due to inconspicuous solutions and increasing number of obstacles(the body of the ‘snake’ itself is also consider as obstacles, and with the game going, it increases, which means more obstacles are generated). As shown in Figure 2, the game is quite complex. So, to design a good AI will be the most challenge and important part of this project.

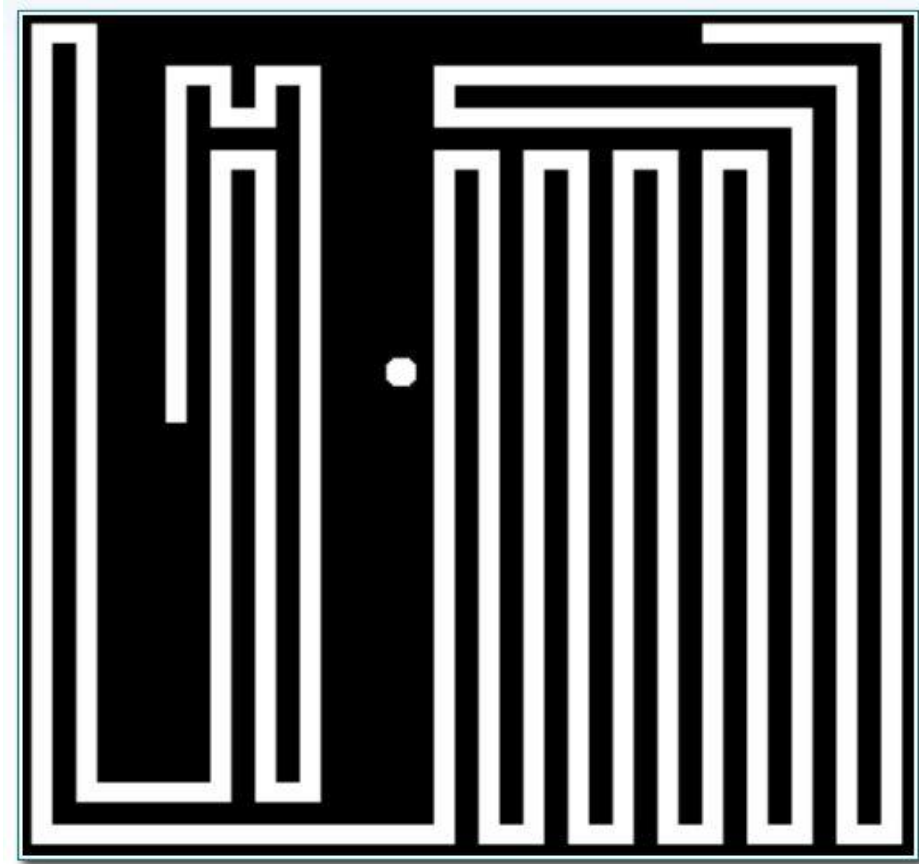


Figure 2: a complex situation of the game.

2. GAME DESIGN AND IMPLEMENTATION

2.1 Elements and their roles

a) *Cell:*

- **Description:** a cell is the basic unit in this game. It defines the minimal space that an element should take.
- **Implementation:** it is a concept. While in the game, it is equivalent to a point, which contains information of its

coordinate.

b) Board:

- **Description:** an $N \times M$ board, with N stands for width and M stands for height (typically, $N = M$), defines the place that the game plays on.
- **Implementation:** it is implemented using a two dimensional array. Each element in the array is a cell, so it contains $N \times M$ cells.
- **Actions:** **1. Refresh:** with the game goes on, it will keeps on updating the information of the game; **2. Clear:** while restart the game, board will clear all other elements on it and initialize the game.

c) Food:

- **Description:** at each time of the game, there is a food in the board. Typically, it occupies a cell.
- **Implementation:** as described it occupies a cell, therefore, we use point to describe it. That is to say, a food is implemented as a point.
- **Actions:** **Generating:** 1. in the beginning, a food is generated; 2. when a food is eaten by the snake, another

food will be generated.

d) Snake:

- **Description:** a snake is the leading actor of the game. It is under the control of player or computer. It has a body front with head and will occupies n cells, where n is the length of the snake.
- **Implementation:** a one dimensional array 'a', which records cells that the snake takes. The head of the snake is 'a [0]'.
- **Actions:**
 - 1. Moving:** A snake has to keep on moving at a speed and each step take a cell;
 - 2. Eating:** A snake can eat the food in the board;
 - 3. Growing:** A snake will increase its length by one, after it eat a food;
 - 4. Die:** A snake will die if it hits an obstacle or edge of the board.

e) Obstacle:

- **Description:** an obstacle is a cell or a list of cells. It cannot be pass through. Thus, if a snake tries to pass through

obstacles, it will die.

- **Implementation:** similar to snake, it is implemented by a one dimensional array o. By definition, the body of the snake is also consider as obstacle.
- **Action: *Generating:*** with different level of difficult, the patterns of obstacles are diverse. (Note: obstacles, except body of the snake, are not generated dynamically. It is generated by the pre-set pattern)

2.2 Rules

- a) ***When does snake die:***
1. Hit obstacles
 2. Hit itself
 3. Hit the edge (also called wall)
 4. Hit the other snake (player VS computer mode)
- b) ***When win:***
1. in single mode, exceed a given score
 2. in player VS computer mode, the one that exceed a given score
 3. In player VS computer mode, if one snake die, the other one wins the game.

3. AI DESIGN AND IMPLEMENTATION

3.1. Requirements

There are mainly 2 aspects that we should take into consideration in the design of AI:

1. *Safe*: the AI should guarantee the snake lives for a long time. Which means each step should be safe not only temporally, but also for a long run.
2. *Competitive*: the AI should also be able to make the snake more competitive, while competing with human being. For instance, typically a shortest path strategy is consider better than longest path strategy.

3.2. Ideas

To meet the requirements of design, there are several strategies that are provided:

- 1) *Shortest path*: the shortest path to the food should be consider as a base for the AI to make decision, especially at the earlier stage of the game. At the earlier stage, typically, the best solution is the shortest path to the food, as it is

unlikely that these movements will lead the snake into a dangerous situation.

- 2) **Safe path:** sometimes the pure shortest path strategy would lead the snake die. As shown in figure 3, if the snake using shortest path, it will die, as it blocks itself. So a safe guaranty path would play a role in this situation. As you can see, the snake find that if it eats the food, it would be extremely dangerous, so it decide to ‘give up’ the food.

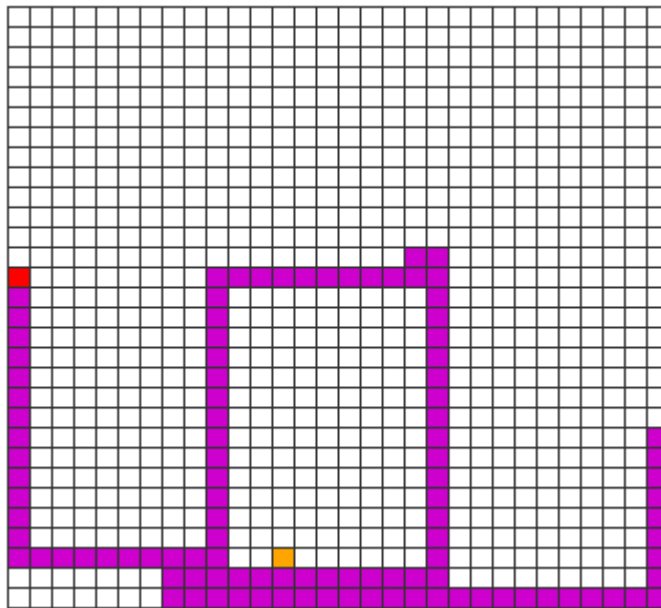


Figure 3. Avoid blocking

- 3) **Game and evaluation function:** a more advanced AI should also take the competitor in to consideration. Meaning that, it should not only make itself into a nice situation, but also tries

to make its competitor into an embarrassed stage. Suppose there is an evaluation function EVA (*snake & s*). So the aim is increase EVA (*AI*) and decrease EVA (*PLAYER*). However, to design an evaluation function could be extremely hard and sometimes the result is not that good as the evaluation could consider too many issues, while in different situation, the weight of different aspects may also change. To simplify the problem, we come up with several strategies:

a) ***Strategic give up***: it is similar to gambit. A gambit is a chess opening in which a player, more often White, sacrifices material, usually a pawn, with the hope of achieving a resulting advantageous position. Although Snake is not as complex as chess, this methodology can be adopted. As shown in figure 4, there are two snakes S1 with green head, and S2 with red head, and the food F is in orange. So obviously, S2 can reach F before S1, if S1 keeps on moving towards F it does not make sense. So at this stage the best decision for S1 is to increase the number cells that are more close to S1 than S2. In other words, it tries to increase the probability that it can eat the next food, as the food are generated randomly, more cells, that are more close to S1 than S2, means higher

probability. So in this situation the shortest path is not the best solution.

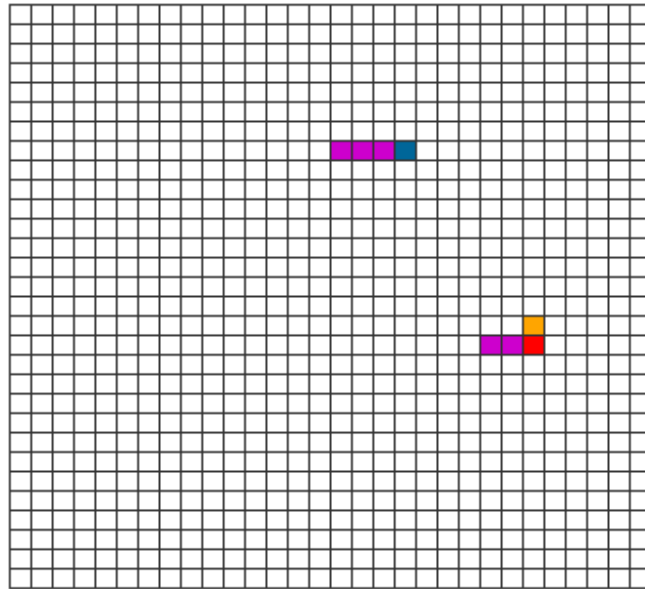


Figure 4. A situation that one snake will definitely reach the food before the other one.

b) ***Vicious competition:*** the idea is that the decision of the snake could be not optimal for itself, as long as the decision could make its competitor into an even worse situation. This is based on the idea that the snake itself can be an obstacle, so proper movement could disturb the competitor a lot. There are 2 ways to disturb the competitor:

- ***By blocking the food:*** it could be great if the snake can

let its competitor away from the food.

- *By blocking the competitor:* this is a more advance way. As said in the rule, if a snake will win automatically if its competitor die. So trying to make your competitor die could be a good way to win the game.

3.3. Implementation

Shortest path: using A* algorithm or Dijkstra algorithm we can find the shortest path easily.

Safe path: the idea is that if the snake moving towards its tail it will never die, so if there is a path between the head and the tail it means that the snake is safe, as in the worst case it can still chase its tail to guaranty its safety. So if a movement leads to a situation that there is no path between head and tail, this action is not allowed. Figure 5 & figure 6 show these two situations. So the algorithm is:

```
Safe path (snake, food) {  
Head = snake [0];
```

```

Tail = snake [length (snake) - 1];
S = shortest path (head, food);
If path (head, tail) exists {
    Decision = S;
    Safe = true;
}
Else
    For each possible moving M {
        If path (head, tail) exists {
            Decision = M;
            Safe = true;
        }
    }
If doesn't safe
    Decision = randomly decision;

Return decision;
}

```

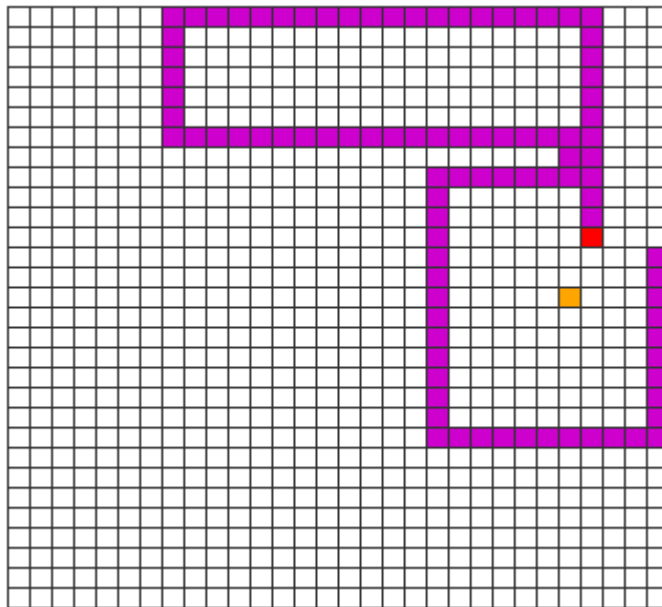


Figure 5: there is a path from head to tail, then it's safe

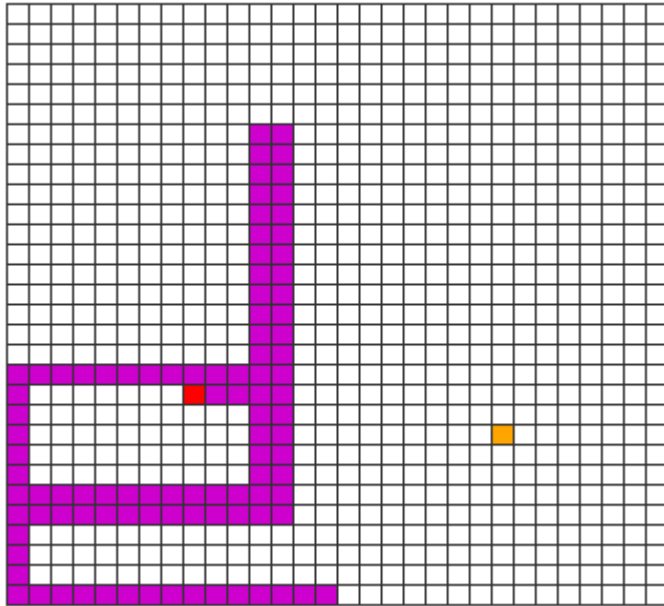


Figure 6. No path from head to tail, going to die!

Game:

1) For strategic give up, suppose AI is S1, competitor is S2, food is F. Then the algorithm is:

```

Give up (S1, S2 and food)
// Give up that food
Head1 = S1 [0];
Head2 = S2 [0];
If Distance (Head1, F) + threshold > Distance (Head2, F)
    Return -1; // Do not need to give up
Else
    //Number of close cells
    N = 0;
    For each possible moving M {
        //Number of close cells for that move

```



```

N_M = 0;
For each free cell C {
    If Distance (head1, C) < Distance (head2, C)
        N_M++;
    }
If N <= N_M {
    N = N_M;
    Decision = M;
    }
}
Return Decision;

```

2) Actually blocking food is not very useful, since most of the time, if the snake is able to block the food, it can eat that food directly. So the powerful one is block the competitor. The algorithm adopts the idea used in safe path, instead of find safe path, the AI tries to make its competitor lose safe path. The algorithm is:

```

Block competitor (S1, S2) {
    Head1 = S1 [0];
    Head2 = S2 [0];
    Tail1 = S1 [length (S1) - 1];
    Tail2 = S2 [length (S2) - 1];

    For each possible move M1 of S1
        For each possible move M2 of S2
            If path (Head2, Tail2) doesn't exists
                Return M1;
    }

```

4. POSSIBLE FURTHER WORKS

There are several possible improvements that could be applied to this project.

- 1) In the game part, we suppose that the probability of each possible movement for the competitor to take is equivalent, but in fact, they can differ much. Therefore, if we can predict the most possible move for the competitor, it could help the AI a lot. While, the problem for me is that, so far I do not have a model to calculate these probabilities properly.
- 2) The user interface can be improved if I have sufficient time. Right now, it is not very user friendly.
- 3) I was trying to build a deeper decision tree. (Right now, we just consider one more step, which means the decision tree just has 2 level) However, it make the game quite slow to have the height of the decision tree large, meanwhile, almost no improvement can be observe if the height is not large enough. So, if there are some smart ideas to prune the decision tree, so that we can deduce the

search space tremendously, then the AI could be improved a lot.

- 4) Some patterns can be observed while playing the game, so if we can record these patterns then it may help the AI make decision. For example, figure 7 shows a pattern that the snake blocks itself. This kind of patterns can be observed frequently. (It's very typical in the game) It seems that the snake is going to die, while with proper movements, the snake can save itself. So if we can record this kind of pattern. Then in the decision tree, we don't need to evaluate the situation, the response time will be much less and also proper movements can be taken.

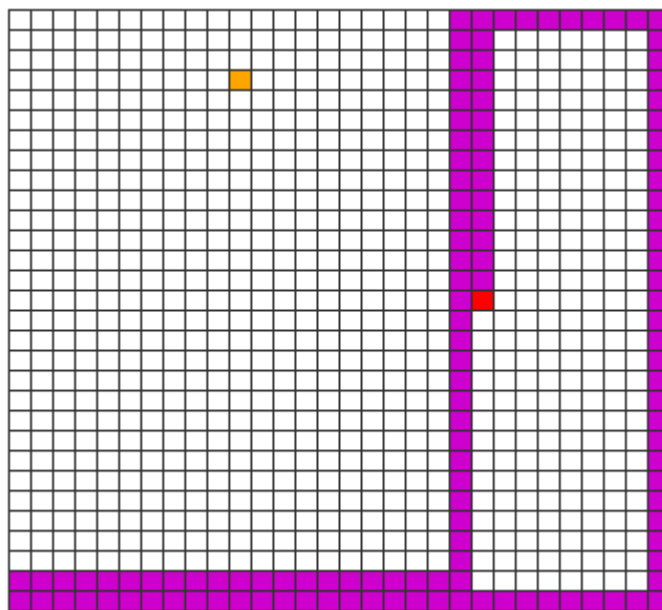


Figure 7. A situation that happens often.

5. SUMMARY

This project implements one version of snake, it designs and implements artificial intelligence for this game so that players can play with computer. The artificial intelligence is integrated with greedy algorithm, graph theory, gaming theory, probability and so on. It is very competitive, while there are still so many possible improvements for the AI.

Appendix

Meeting Minutes

I. Minutes of the 1st Project Meeting

Date: 21 February, 2014

Time: 9:30am

Place: Rm. 3512

Attending: LI Zhi, Dr. David Rossiter

Absent: None

Recorder: LI Zhi

1. Approval of minutes

This is first formal group meeting, so there were no minutes to approve.

2. Report on Progress

This is the first meeting, so there was nothing to complete.

3. Discussion Items

Things to do

- Design details of the game
- Decide technics that will be used
- Implement one-player mode.

4. Meeting adjournment and Next meeting

The meeting was adjourned at 10:00 PM.

II. Minutes of the 2nd Project Meeting

Date: 21 March, 2014

Time: 9:45am

Place: Rm. 3512

Attending: LI Zhi, Dr. David Rossiter

Absent: None

Recorder: LI Zhi

1. Approval of minutes

The minutes of the last meeting were approved without amendment.

2. Report on Progress

All things discussed in last meeting are complete.

3. Discussion Items

Things to do

- Adding more features into the game
- Design and implement the basic AI algorithm

4. Meeting adjournment and Next meeting

The meeting was adjourned at 10:15 PM. The next meeting will be held in Apr.

III. Minutes of the 3rd Project Meeting

Date: 22 April, 2014

Time: 2:00 am

Place: Rm. 3512

Attending: LI Zhi, Dr. David Rossiter

Absent: None

Recorder: LI Zhi

1. Approval of minutes

The minutes of the last meeting were approved without amendment.

2. Report on Progress

All things discussed in last meeting are complete.

3. Discussion Items

Things to do

- Design and implement advance AI
- Starting final report

4. Meeting adjournment and Next meeting

The meeting was adjourned at 2:30 PM. The next meeting will be held in May.

IV. Minute for the 4th Project Meeting

Date: 14 May, 2014

Time: 3:00 am

Place: Rm. 3512

Attending: LI Zhi, Dr. David Rossiter

Absent: None

Recorder: LI Zhi

1. Approval of minutes

The minutes of the last meeting were approved without amendment.

2. Report on Progress

All things discussed in last meeting are complete.

3. Discussion Items

Things to do

- Record video for the project
- Finalize report for the project

4. Meeting adjournment and Next meeting

The meeting was adjourned at 3:30 PM.