EEE102 Assessment 3 Report

Yimian Liu 1717608

April 29, 2019

1. Introduction

In the previous lectures, we had obtained a basic understanding on C++ and had programed with several objects and classes, and also interact with files. In the last assignment, we had written a fraction class and also modified a game to make it work. During that process, we had got a overview of how a C++ project should look like. In this assignment, we are required to design and develop a real programming project, a monopoly game. In this game, players can obtain game experience and joy just as other games brought us. In other words, I decided to regard the assignment as an independent project and may require the method of project management.

This report will detailly classify and present how this assignment be conducted with the instruction of Software Development Process (SDP).

2. Problem Statement

Problems of this game development can be concluded into three parts, the Login part, the Game part and also the Data part.

2.1 Login part

The main purpose of the login part is to provide a login procedure for the game, which could contain the sign up for new players, the sign in for old players and also the creation and recovery of AI players.

2.1.1 For New players

- New players could set their user name, which is no more than 8 words.
- New players could set their password.
- The password should be retyped to confirm by the user.
- The password should be no more than 8 words.
- The password should not be displayed on the screen directly.
- At any time press ESC can go back to menu.

2.1.2 For Old players

- Old players could input their user name.
- Old players could input their password.
- The password should be checked.
- The password should not be displayed on the screen directly.
- The player's data should be recovered.
- At any time press ESC can go back to menu.

2.1.3 For AI players

- For existed AI, recover their data.
- For new AI, create their account.

2.2 Game part

The Game part provides the monopoly game. There are three modes of the Game.

2.2.1 Player-Player Mode

- Two players should login.
- Players play with each other.
- If these two players have a unfinished game, they can choose if to continue it.
- In other case, create a new map links to these two players.

2.2.2 Player-Computer Mode

- One player should login.
- One AI player should be generated or recovered.
- If the players have an unfinished game, he can choose to continue it.
- In other case, create a new map with player and a generated AI.

2.2.3 Computer-Computer Mode

- No player should login.
- Two AI player should be generated or recovered.
- If there is an unfinished game, user can choose to recover it.
- In other case, create a new map with two AI.

2.2.4 Overall Requirement

- Each player haves an ini $5000 balance.
- Map contains 38 sellable squares, 1 GO for gain $200 when in, 1 JAIL for waiting for another 1 round when in.
- Players cast dice to decide the steps to move. (1-6)
- When player at a unoccupied sellable square, he can choose to buy it.
- When player at his owned square, he can choose to upgrade it, by paying another 50% of its price. The price of the square will increase 5% after upgrading.
- A square can be unlimited upgraded, but only once for each round.
- When player at opponent's isolated square, he needs to pay for 10% of its price.
- When player at opponent's connected square, he needs to pay for 20% if its price.
- After one player bankrupt, the game over, the one who not bankrupt wins.
- A ESC can be pressed at certain time to go back to menu.

2.3 Data part

Data part take charge of all the data storage and data recovery process. Basing one this part, the information storage and recover of players and unfinished game can be achieved.

2.3.1 Data Package

- Data package provide a format to process data.
- Data package could operate with most types of format, such as string, int.
- Data should easily be pushed into a data package.
- Data can be easily getten from a data package.

2.3.2 Data Storage and Read (File Database)

- Database should accept a data package and an index key to storage data into files.
- Database should return a data package with an index key from files if the corresponded data exist.
- There is a method to know if a certain data package exist in database.
- The database should be relatively high efficient.
- The data in files should be encoded.

3. Analysis

3.1 Login Part

3.1.1 Variables

The login part needs many variables, especially temporary variables. For example, when login, we assumed that there should be a temp player object to deal with the input information. And also, as when login, the existence of the player is now certain, a temp data package should also be used to temporarily storage the search result from the database.

3.1.2 Input & Output

The input of login part is mainly the name and password the user typed in. And also, there are several command input such as an ESC should also be considered for returning to the menu page. Besides, situations such as user types a wrong password, user type nothing, user type in two different password in password confirm stage should print a suitable hint on the screen to instruct user's action.

3.2 Game Part

3.2.1 Variables

The variables in game section is mainly the map, and the two players. In the map, the position of these two players should be instantly storage. Besides, the ownership, price, owner type, level of each square should also be managed by the map. For the players object, it should manage the password, username, user's account balance, and also user's linked map.

3.2.2 Input & Output

There are a set of input and output in game part. The common form of input is to let user to choose an option to instruct a personal action such as buy a square and cast the dice. The output of this game are mostly achieved by a graphical interface draw by command characters. With this method, a game map can be print on the screen, and the movement and action of players can be displayed with anime, which is considered to be more vivid and friendly.

3.3 Data Part

3.3.1 Variables

The inner variable of data package is mainly in string format. I choose string because string is very stable and convenient in C++, and also the transformation form string to other form is considered very easy, while all the transform form other types to string can be achieved with a to_string() function in C++11 standard.

3.3.2 Input & Output

The input of database is packages into data packages format. The database can output these data to storage them into files. And also, the database can also read the data from files and package them into data package format.

4. Design

4.1 Overall Structure

This game was designed with mainly 12 parts. As it shown in Figure 1, in the main function, a controller takes charge all of the functions of the game. It includes a startinterface which provide an anime which be played when user open the game program. The map is achieved with square, player and also some library in ovo.h. The player mainly use some data process in ovo library. Besides, the controller also manages a hintbox, msgbox, scoreboard and also a roll (dice). In addition, there are also tools in controller for overall usage.

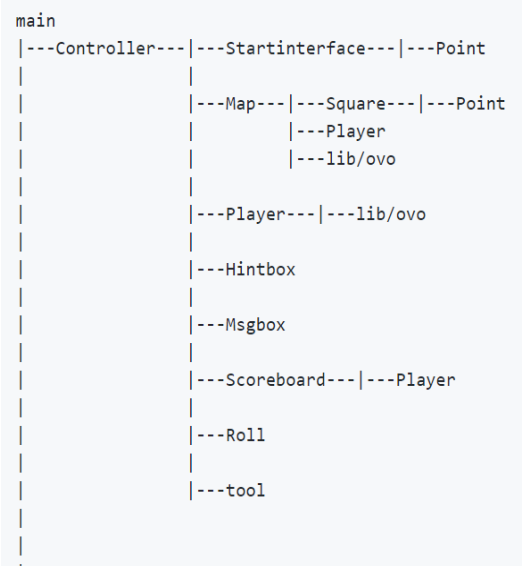

Figure 1 – Overall Structure

4.2 lib/ovo

Ovo library is my personal C++ library which was classified and developed mainly by me for personal usage. In this library, some math methods, a data package format and a file database are provided. In this game, I will use these three sections of ovo.h.
url: https://github.com/eeeneko/ovo

4.2.1 ovo::math

Ovo::math provides many encoding methods such as MD5, SHA256, Base64, AES and RSA. In this project, I used MD5 and SHA256 to generate the map ID and square ID. I also use Base64 to storage string data and to prevent the effect of special character such as space and '\n'.

4.2.2 ovo::data

Ovo::data provides a format for storage data. The inner process of ovo::data is basing on map. In other words, ovo::data is a further packing up of std::map. It can storage data with a key and a value. I include this for the convenience of interacting with database.

4.2.3 ovo::database

Ovo::database provides a file database. This is a no-sql data base and was designed to storage ovo::data with an index key. This means that, you can use an index key to storage data, and also get the data with the corresponded index key. With this file database, the interact with disk can be very easy and convenient.

4.3 Controller

Controller is a class which manage the game overall.

Controller	
Create a game - public	Map
Display welcome page - private	Player
Display selection page - private	Msgbox
User login - private	Hintbox
Draw game - private	Roll
Play game - private	Scoreboard
Player in game - private	lib/ovo
AI in game - private	Startinterface
Update score board - private	
Print hint - private	

Table 1 – Controller CRC card

As it shown in Table 1, a controller takes charge of the whole game process. The controller manages and coordinates objects in game such as the map, players, message box, score boards, dice, and even the welcome anime. Each of these objects are relatively independent, and it is the controller which combines them all together and finally present a monopoly game.

4.4 Player

Player	
Cost money - public	lib/ovo
Gain money - public	
Check password - public	
Link map - public	
Return linked map - public	
Get user name - public	
Get balance - public	
Reset - public	
Save data to file - private	

Table 2 – Player CRC card

Similarly, the CRC card of player are shown as the above table. The player class takes charge of the password check process. For each player, there is a account and also a linked map. The linked map can help player to find his old map when login thus he can continue his old unfinished game.

4.5 Map

Map	
Print map framework - public	Player
Move player - public	Square
Get player's position - public	Point
Get map ID - public	lib/ovo
Save data to files - private	

Table 3 – Map CRC card

The map class mainly have two tasks. The first one is to record the player's position. The second is to linked all the squares and also the players. The map in this game is like a telephone operator, who help the player to find their squares and can also help players to find their partners. The map takes charge of all the 40 squares.

4.6 Square

Square	
Print - public	Player
Buy - public	Point
Get square id - public	lib/ovo
Get price - public	Friend Controller
Get level - public	
Level up - public	
Save data to files - private	

Table 4 – Square CRC card

The square is very basic class in this game. As there are totally 40 squares, and each of them is a square object. All of the graphic action of the squares are also achieved by this class. The square use point to draw the interface.

4.7 Msgbox

Msgbox	
Set title - public	
Print - public	
Clear - public	

Table 5 – Msgbox CRC card

A msgbox is designed to display a message and also get feedback from users. The interact part of the game and also the login section are achieved base on this class.

4.8 Hintbox

Hintbox	
Set title - public	
Print - public	
Clear - public	

Table 6 – Hintbox CRC card

Similar to the msgbox, the hintbox provide the similar function as the msgbox do, but in different shape.

4.9 Scoreboard

Scoreboard	
Print - public	Player

Table 7 – Scoreboard CRC card

The score board is to show the balance of the players on the screen. In the game, there are two scoreboard objects, one for player1 and another for player 2.

4.10 Roll

Roll	
Cast - public	
Print - private	
Clear - private	
Anime - private	

Table 8 – Roll CRC card

The roll class provide a dice object for the game. Thanks to this class, it is much easier to add a anime for the dice casting.

4.11 Point

Point	
Print - public	
Print circular - public	
Print star - public	
Print row line - public	
Print column line - public	

Print left line - public	
Print right line - public	
Clear - public	
Change position - public	
Get position - public	
Set color - public	

Table 9 – Point CRC card

The point class provide different operations and shapes of points on screen. Basing on this class, the anime can be easily achieved. And also, this class let me to focus on drawing a more user-friendly interface.

4.12 Startinterface

Startinterface	
Print Cartoon - private	Point
Clear Cartoon - private	
Print Text - private	
Anime - public	

Table 10 – Startinterface CRC card

This class is to display a welcome anime at the beginning of the program.

5. Implementation

This project includes four folders. The bin folder contains an exe file of this game, which is just in case. In the docs folder, there is a back-up of this report. Besides, the code files are all in the lib folder and the src folder.

There are 11 .H files and 12 .cpp files in the src folder, which means the source code. The code of the ovo library is in the lib folder.

Since I have written some codes that belong to C11, it is recommended to use a complier such as gcc. It is strongly recommended to use the **click-me.bat** to compile the project.

This project is on Github. Project url: https://github.com/IoTcat/monopoly

src/main.cpp
This file includes the main function of this project.
url: https://github.com/IoTcat/monopoly/blob/master/src/main.cpp

src/controller.h
This file includes the declaration of Controller class of this project.
url: https://github.com/IoTcat/monopoly/blob/master/src/controller.h

src/controller.cpp
This file includes the functions of Controller class of this project.
url: https://github.com/IoTcat/monopoly/blob/master/src/controller.cpp

src/hintbox.h
This file includes the declaration of Hintbox class of this project.
url: https://github.com/IoTcat/monopoly/blob/master/src/hintbox.h

src/hintbox.cpp
This file includes the functions of Hintbox class of this project.
url: https://github.com/IoTcat/monopoly/blob/master/src/hintbox.cpp

src/map.h
This file includes the declaration of Map class of this project.
url: https://github.com/IoTcat/monopoly/blob/master/src/map.h

src/map.cpp
This file includes the functions of Map class of this project.
url: https://github.com/IoTcat/monopoly/blob/master/src/map.cpp

src/msgbox.h
This file includes the declaration of Msgbox class of this project.
url: https://github.com/IoTcat/monopoly/blob/master/src/msgbox.h

src/msgbox.cpp
This file includes the functions of Msgbox class of this project.
url: https://github.com/IoTcat/monopoly/blob/master/src/msgbox.cpp

src/player.h
This file includes the declaration of Player class of this project.
url: https://github.com/IoTcat/monopoly/blob/master/src/player.h

src/player.cpp
This file includes the functions of Player class of this project.
url: https://github.com/IoTcat/monopoly/blob/master/src/player.cpp

src/point.h
This file includes the declaration of Point class of this project.
url: https://github.com/IoTcat/monopoly/blob/master/src/point.h

src/point.cpp
This file includes the functions of Point class of this project.
url: https://github.com/IoTcat/monopoly/blob/master/src/point.cpp

src/roll.h
This file includes the declaration of Roll class of this project.
url: https://github.com/IoTcat/monopoly/blob/master/src/roll.h

src/roll.cpp
This file includes the functions of Roll class of this project.
url: https://github.com/IoTcat/monopoly/blob/master/src/roll.cpp

src/scoreboard.h
This file includes the declaration of Scoreboard class of this project.
url: https://github.com/IoTcat/monopoly/blob/master/src/scoreboard.h

src/scoreboard.cpp
This file includes the functions of Scoreboard class of this project.
url: https://github.com/IoTcat/monopoly/blob/master/src/scoreboard.cpp

src/square.h
This file includes the declaration of Square class of this project.
url: https://github.com/IoTcat/monopoly/blob/master/src/square.h

src/square.cpp
This file includes the functions of Square class of this project.
url: https://github.com/IoTcat/monopoly/blob/master/src/square.cpp

src/startinterface.h
This file includes the declaration of Startinterface class of this project.
url: https://github.com/IoTcat/monopoly/blob/master/src/startinterface.h

src/startinterface.cpp
This file includes the functions of Startinterface class of this project.
url: https://github.com/IoTcat/monopoly/blob/master/src/startinterface.cpp

src/tools.h
This file includes the declaration of tools functions of this project.
url: https://github.com/IoTcat/monopoly/blob/master/src/tools.h

src/tools.cpp
This file includes the functions of tools of this project.
url: https://github.com/IoTcat/monopoly/blob/master/src/tools.cpp

lib/ovo.h

This file includes the declaration ovo.

url: https://github.com/IoTcat/monopoly/blob/master/lib/ovo.h

lib/ovo.cpp

This file includes the functions of ovo.

url: https://github.com/IoTcat/monopoly/blob/master/lib/ovo.cpp

6. Test

6.1 Welcome Interface

Figure 2 – Welcome interface

As it shown in Figure 2, after a welcome cartoon, the screen pause at this page.

After we press a key, we got the following screen.

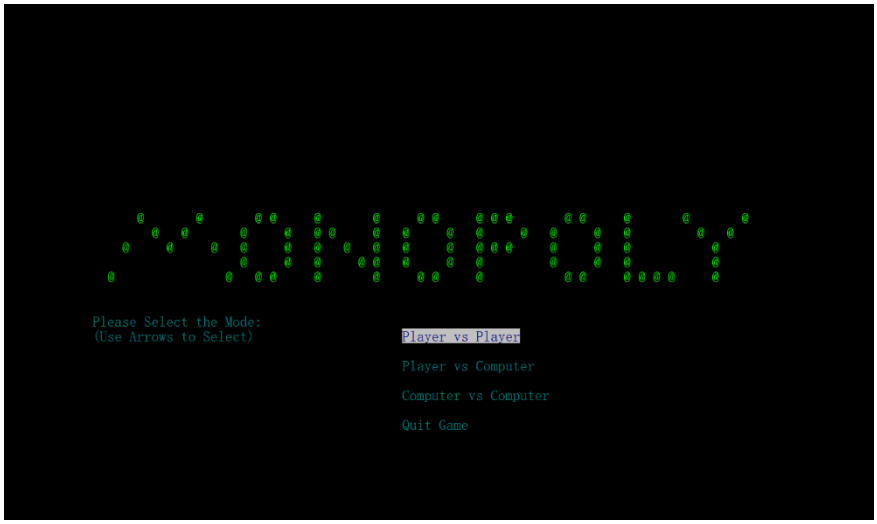

Figure 3 – Mode Selection

As it shown in Figure 3, the mode selection part was displayed as wished. At this stage, we tested to use arrow to move the selections. And it worked well. Then we test the quit option. It quit as wish.

6.2 Login Interface

Sign up

At first, we chose the player-computer mode. Then it come to the following screen.

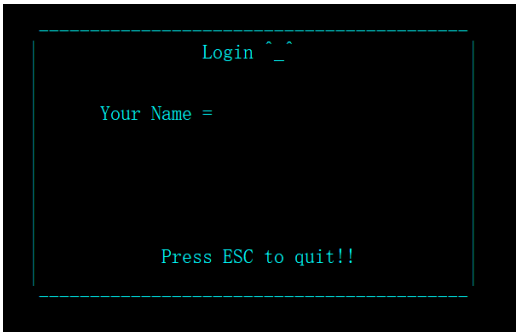

Figure 4 – Login Box

As it shown in Figure 4, there is a login box, and waiting for input a name.

We test ESC here and find it go back to menu successfully.

The we back to this page and type a name.

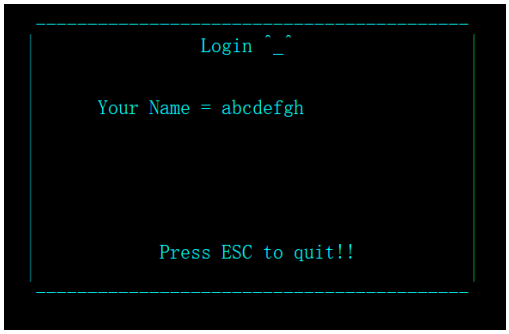

Figure 5 – Name Input Test

As it shown in Figure 5, when we trying to type more than 8 words, it can not be typed in.

Then we press Enter and go to the next page.

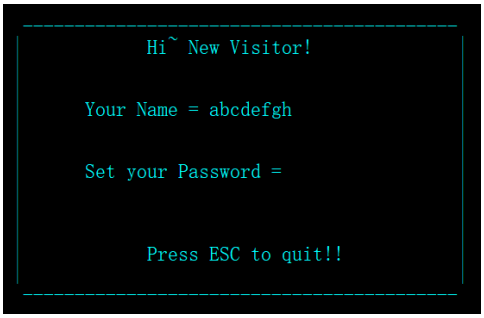

Figure 6 – Password Set Page

As it shown in Figure 6, it recognized that I am a new visitor and let me to sign up a new account by set a password.
We type nothing as password.

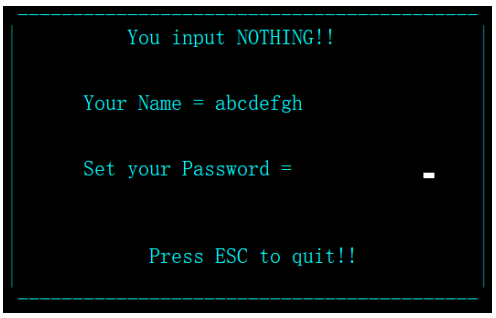

Figure 7 – Input Nothing as Password

As it shown in Figure 7, the game refused the empty input.
Now we type 1234567890 as password.

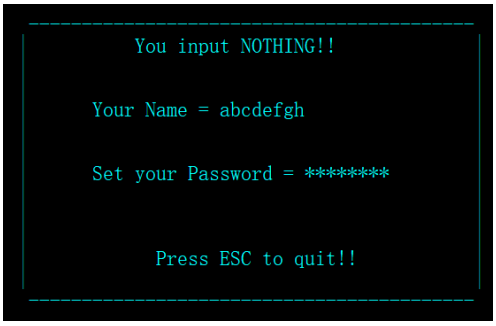

Figure 8 – Set Password

Figure 8 shows that the password only accept the first 8 input which is “12345678” as its password. Also it can be seen that the password was hided. And the function of backspace was also confirmed. Then we press Enter to go to the next page.

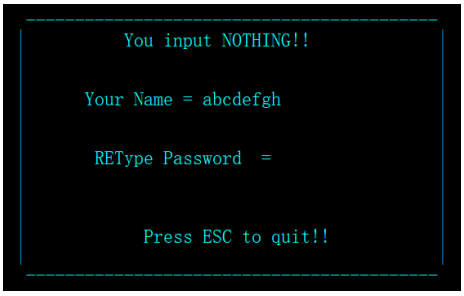

Figure 9 – Password Confirm

As it shown in Figure 9, it need me to retype the password to confirm.
We input a different one.

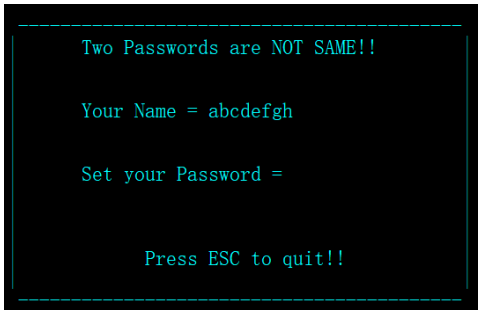

Figure 10 – Password Not Same

It told me that the two passwords are not the same.
Then we try to type two same password.
Then we go into the game.

Sign in

We restart the program and select the player-computer mode to login.
We type the username we just set.

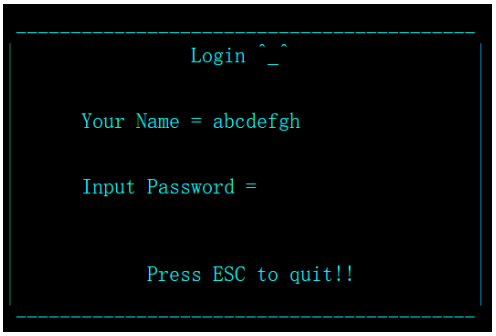

Figure 11 - Login

As it shown in Figure 11, the program recognize that I am a old user and let me to input my password.
We input a wrong password.

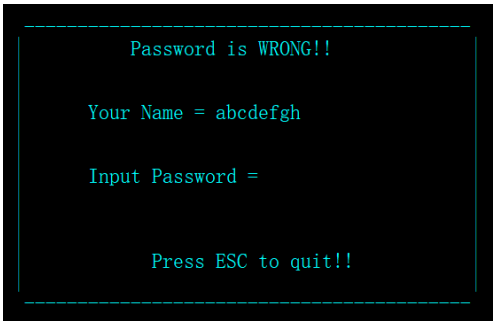

Figure 12 – Wrong Password

It told me that the password is wrong.
Then we input a correct password and then we go into the game.

6.3 Game Interface

Player-Computer Mode

We login into the player computer mode and start the game.

Figure 13 – Game Interface 1

As it shown in Figure 13, there are two players, me and a generated AI, stay at the GO square. The balance of all players are $5000. Each square have a random price from 10 to 300 and at lv.0 now.

Now we select cast.

Figure 14 – Game Interface 2

As it shown in Figure 14, the dice return a 5. So the yellow point which is me go forwards for 5 squares and it then let me to choose if to buy this square.

I choose to buy it.

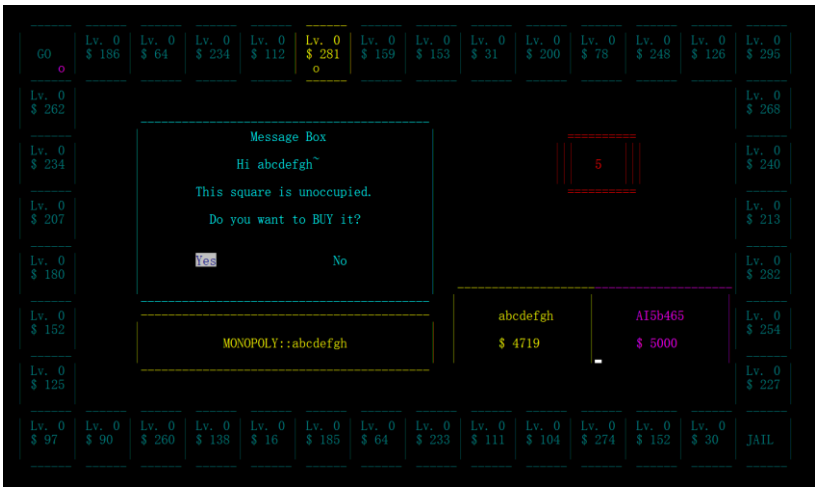

Figure 15 – Game Interface 3

As it shown in Figure 15, the square was brought by me and the color of the square changed. It can also be noticed that the balance of me decrease 281.

Then it's AI's turn.

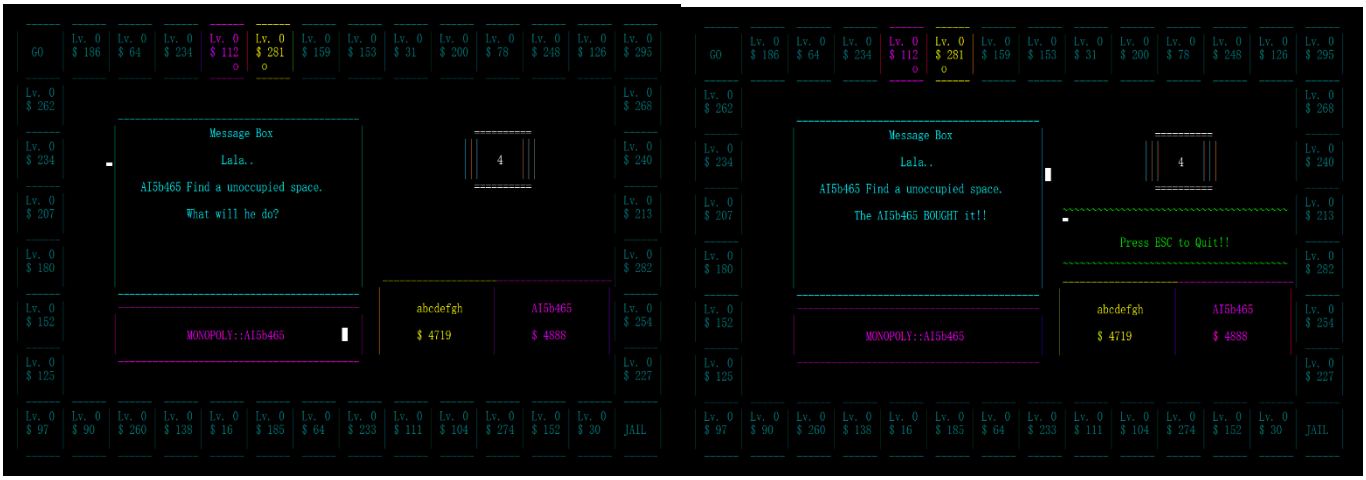

Figure 16 – Game Interface 4

As it shown in Figure 16, the AI roll 4 and it buy the place.

Then its my turn again.

Figure 17 – Game Interface 5

Figure 17 shows the situation that the AI go into the JAIL. As it shown in the message box, it need to wait for one more round.

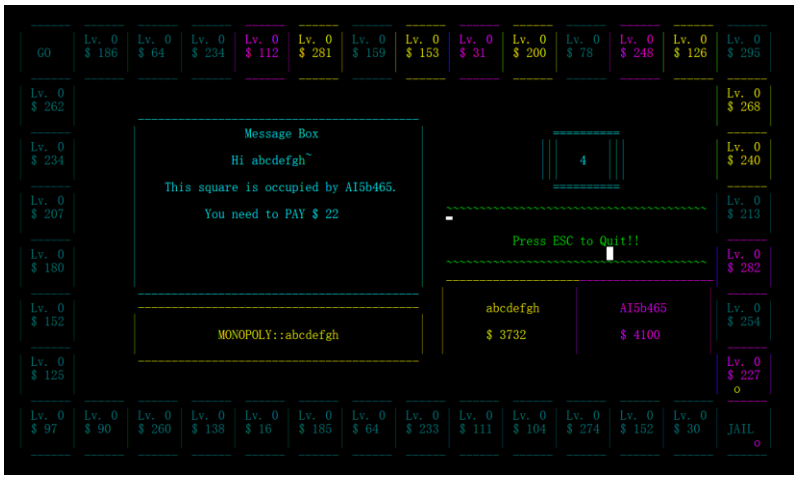

Figure 18 – Game Interface 6

Figure 18 shows the situation that I step into the AI's isolated square. As it shown, I was fined for $22, which is 10% of the square price. The balance of mine then change to $3710.

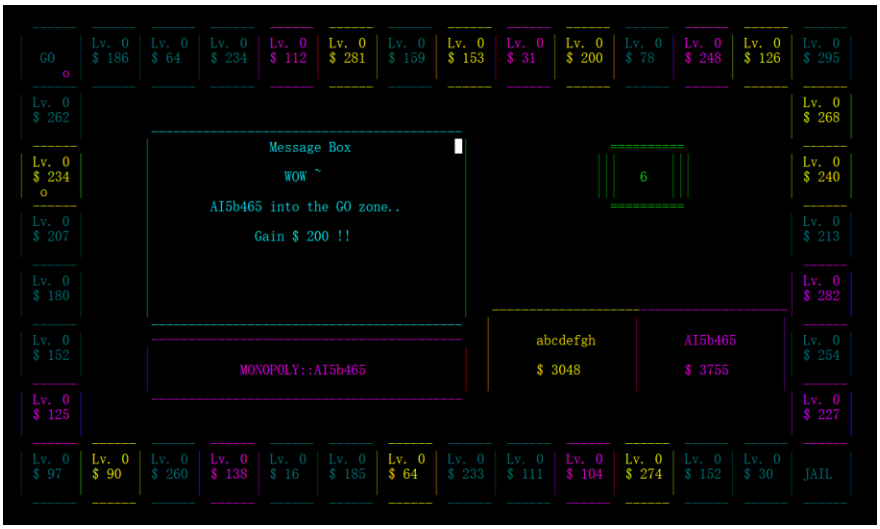

Figure 19 – Game Interface 7

Figure 19 presents the situation that the AI get into the GO square. As it shown, the AI had obtain $200. Its balance then increased to $3955.

Figure 20 – Game Interface 8

Figure 20 shows the situation that I step into a square belong to me. I can choose to upgrade it.

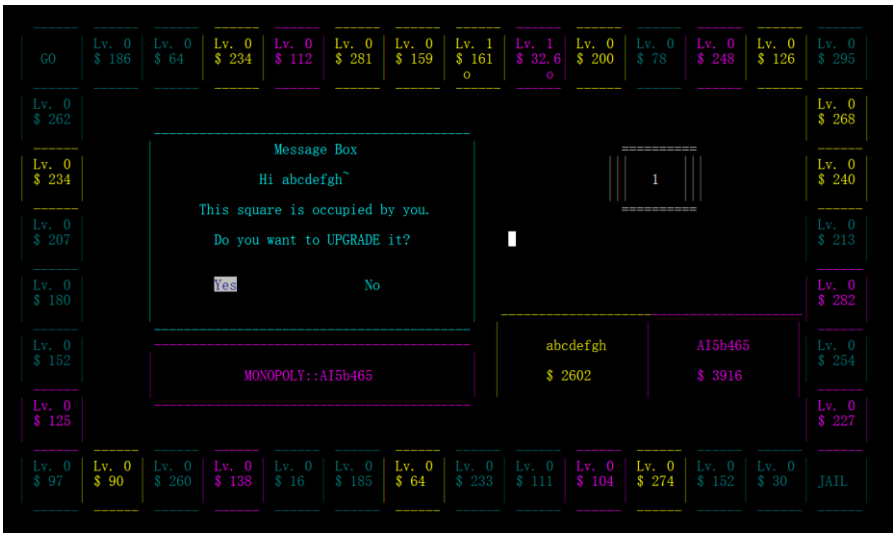

Figure 21 – Game Interface 9

As Figure 21 indicated, after I choose to upgrade it, the square become lv.1 , the price increased from $153 to $161, which is 5%. And also, my balance had decreased from $2678 to $2602, which is 50% of $153.

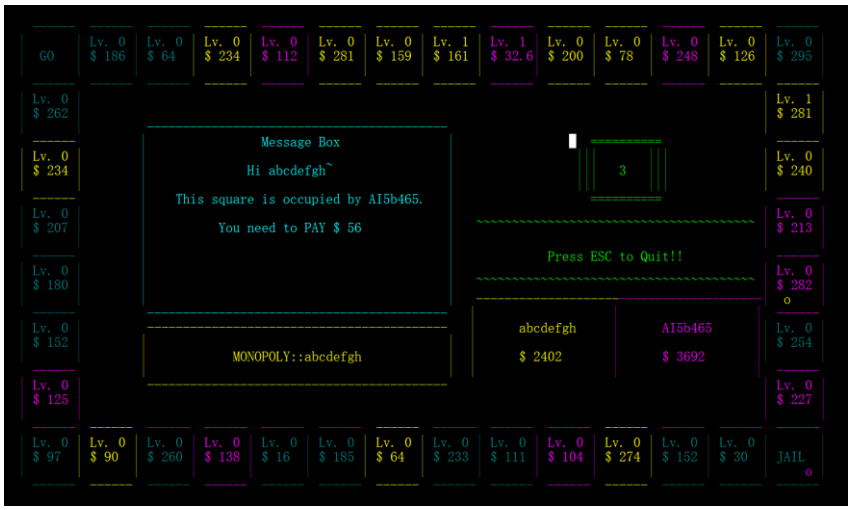

Figure 22 – Game Interface 10

As it shown in Figure 22, I step into the connected squares of AI. I was fined for $56, which is 20% of its price.

In the condition that player meet a connected three squares which belong to opponent, the player is fined for 20% of the square's price.

Player-Player Mode

As game part of player is using the same code as the Player-computer mode, we only test the part that are different.

Figure 23 – Game Interface Player-Player

After login, we come into a map which contains two players. Now its 111's turns.

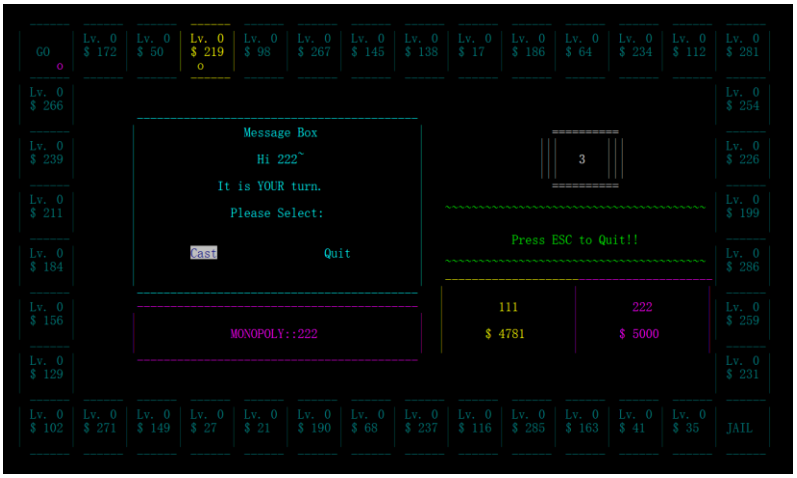

Figure 24 – Game Interface Player-Player 2

As it shown in Figure 24, after 111's operation, 222 can also do as what 111 do to select it's options.

Computer-Computer Mode

The computer-computer mode is just like the other two modes, only the players use two AI.

Figure 25 – Game Interface Computer-Computer

Figure 25 shows the situation that two AI play for more than 3 hours. As it shown, it functions well.

6.3 Data Storage

The data of user and map can be storage into files so that every time when they open the game they can choose to continue the unfinished game.

Now we use the old account 'abcdefgh' to open the player-computer mode.

Figure 26 – Message Box for Continue Game

As it shown in Figure 26, I can choose if to recover a old map.

I choose yes.

Figure 27 – Old Map

As it shown in Figure 27, I got the old map I unfinished last time.
If I choose no in Figure 26.

Figure 28 – New Game

As it shown in Figure 28, I then start a new game with a new AI in a new map.

As the other two mode are use the same code for this part, so they will get the same result as the Player-Computer mode.

The data are all storage in the data folder.

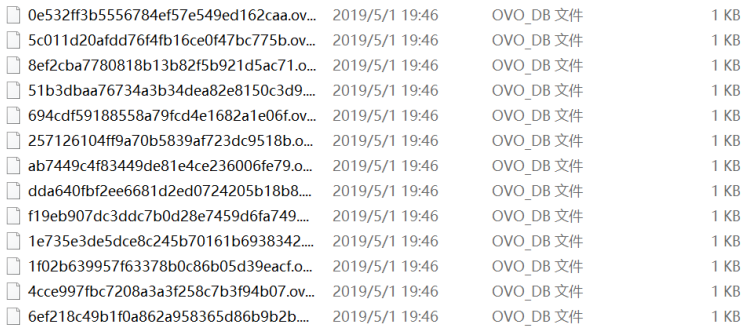

Figure 29 – Data Files

As shown in Figure 29, there are a number of data files in data folder. Each of them storage one package of data.

Figure 30 – Data File Content

As you can see in Figure 30, the data in data file are encoded.

7. Conclusion

In this report, the problem statement, analysis, design, and also the test process of developing a monopoly game had been detailly discussed. In this assignment, I had cultivated a sense of project management to manage more than 20 code files and libraries and finally compiled them into an exe file. In this process, I also realized the importance and convenience of having a personal library. Before this assignment, I had developed the ovo.h personal library and classify many methods and functions into it with the format I familiar with. Thanks to this library, in this assignment I can focus on how to achieve a graphic interface using command lines and concentrate on the game logic and game experience without warried about the file process.