07 MAY, 2018

Tecnológico de Monterrey

FINAL PROJECT

ADVANCED GRAPHIC BLACKJACK

PRESENTED BY: **SEUNG HOON LEE, A01021720 & SAMUEL TORRERO, A01361063**

ADVANCED PROGRAMMING

GILBERTO ECHEVERRÍA FURIÓ

THE PROGRAM

Our final project is a complex blackjack game. It has a graphical SFML component to draw cards, actions and text. The program works with a server and up to 4 different players that connect to it. Our program must use threads to handle each client connection as well as each graphical component within the client. It also must use mutexes to handle the variables that are shared between the threads, so there is no variable that gets modified by two treads at a single time.

Program also uses dynamic memory which allocates memory for the server information. It uses signals to handle the exit and stops for the server, so the clients can know when a server is disconnected.

Finally program uses pointers to save information of every player.

PROGRAM METHODS

GAME SERVER PROGRAM

USAGE

It shows the user how to initiate the server if he misses the correct syntax.

SETUPHANDLERS

Handles signals to notify all clients when the server is being closed.

INITGAMESERVER

Initializes all the information needed, allocates the memory for the players and mutexes.

WAITFORCONNECTIONS

Creates a loop to wait for incoming connections. Waits certain time for players, if there are no players closes the game. If a player connects it waits 15 seconds for other players, after that time it start with the players connected.

ATTENTIONTHREAD

Main server method. It handles all player requests. Has a main loop which sends and receives all information (cards, movements, bets) and shows scores.

CLOSEGAMESERVER

Receives the structure of server data and locks. Frees memory used.

CHECKVALIDACCOUND

Receives the player number. Checks that the number of players is within the limit.

ONINTERRUPT

Receives a signal. Handles the Ctrl-C signal, advises all players that the server is being closed.

ONSTOP

Receives a signal. Handles the Ctrl-Z signal, advises all players that the server is being stopped.

CHECKWINNER

Receives an array with scores and the total number of players. Compares the different scores and returns the player number for the winner.

PRINTCARDS

Receives the size of the deck for the server and the deck of cards itself. Iterates through it to print each card.

STARTDECK

Receives the size of the deck for the server and the deck of cards itself. Iterates through it to initialize it, so it delivers cards between 1 (ace) and 11.

HITCARD

Receives the size of the deck for the server and the deck of cards itself. Generates a new card for the deck and increments the deck size.

CHECKLOSE

Receives the size of the deck for the server and the deck of cards itself. Checks if the total score with current cards is greater than 21, if true it returns the lose flag.

TOTALSCORE

Receives the size of the deck for the server and the deck of cards itself. Iterates through the deck and sums the card scores. Returns the total score.

GAME PLAYER PROGRAM

USAGE

It shows the user how to initiate the client if he misses the correct syntax.

BLACKJACKOPERATIONS

It handles all the player logic. Receives the amount for bets, delivers cards to the player and shows him all the possible options:

1. Players sets his initial bet and receives his two first cards
2. After the player has his first two cards he can
    a. Hit: receive another card
    b. Stand: keep his current cards and move to next phase
3. When he ends hitting because he stood player can
    a. Settle with his current bet
    b. Double down his bet
    c. Withdraw from the game (half of the money is lost)

The method then checks the player total score, sends it to the server and receives a winner.

PRINTCARDS

Receives the size of the deck for the player and the deck of cards itself. Iterates through it to print each card.

STARTDECK

Receives the size of the deck for the player and the deck of cards itself. Iterates through it to initialize it, so it delivers cards between 1 (ace) and 11.

HITCARD

Receives the size of the deck for the player and the deck of cards itself. Generates a new card for the deck and increments the deck size.

CHECKLOSE

Receives the size of the deck for the server and the deck of cards itself. Checks if the total score with current cards is greater than 21, if true it returns the lose flag.

TOTALSCORE

Receives the size of the deck for the player and the deck of cards itself. Iterates through the deck and sums the card scores. Returns the total score.

RUN_SFML

This method handles all the graphic interface. It creates all the objects which are:

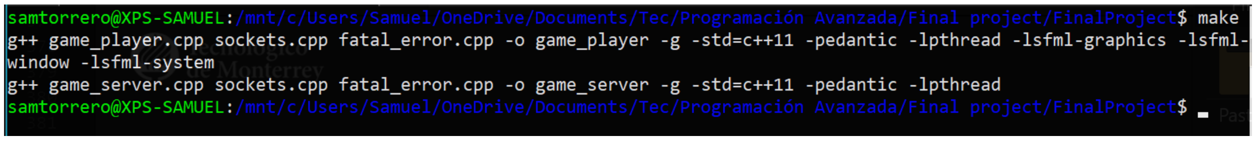
1. Buttons
2. Text
3. Inputs
4. Cards
5. Background

It also has a main loop which handles in a separate threat. This loop stays while a window of SFML is open, and it updates each element within it. Inside this loop there is the button color change effect as well as the validation for the click of each of them. When a button is pressed a function of the game is called. There is certain logic inside to handle whether the objects are visible or not.

PROGRAM COMPILING AND RUNNING

The game has a "makefile" to compile all the programs. We skipped the `-Wall` flag in the player file because it throwed warnings about all the different modules of SFML not being handled in the event switch. We are not using those modules at all, so the warnings had no sense. However, we do are using the flag to compile the server program.

To compile both programs we must first locate in the folder where the programs are, then the user just needs to type `make` into the console and hit enter. The output of typing that should look like this:



```
samtorrero@XPS-SAMUEL:/mnt/c/Users/Samuel/OneDrive/Documents/Tec/Programación Avanzada/Final project/FinalProject$ make
g++ game_player.cpp sockets.cpp fatal_error.cpp -o game_player -g -std=c++11 -pedantic -lpthread -lsfml-graphics -lsfml-
window -lsfml-system
g++ game_server.cpp sockets.cpp fatal_error.cpp -o game_server -g -std=c++11 -pedantic -lpthread
samtorrero@XPS-SAMUEL:/mnt/c/Users/Samuel/OneDrive/Documents/Tec/Programación Avanzada/Final project/FinalProject$
```

The next step is to run the program. For playing you will need at least two different terminals, one for the server and one for the client. More terminals (3) can be added if more players want to join the game.

There are a few things to consider before running the player program:

1. Any player program must run within the next 10 seconds after the server program is running, otherwise the timeout will run out and the server will need to be run again.
2. The player program uses SFML to run the graphical environment, so it needs to be installed before compiling it. To install SFML in a Linux environment run the following line:

```
sudo apt-get install libsfml-dev
```

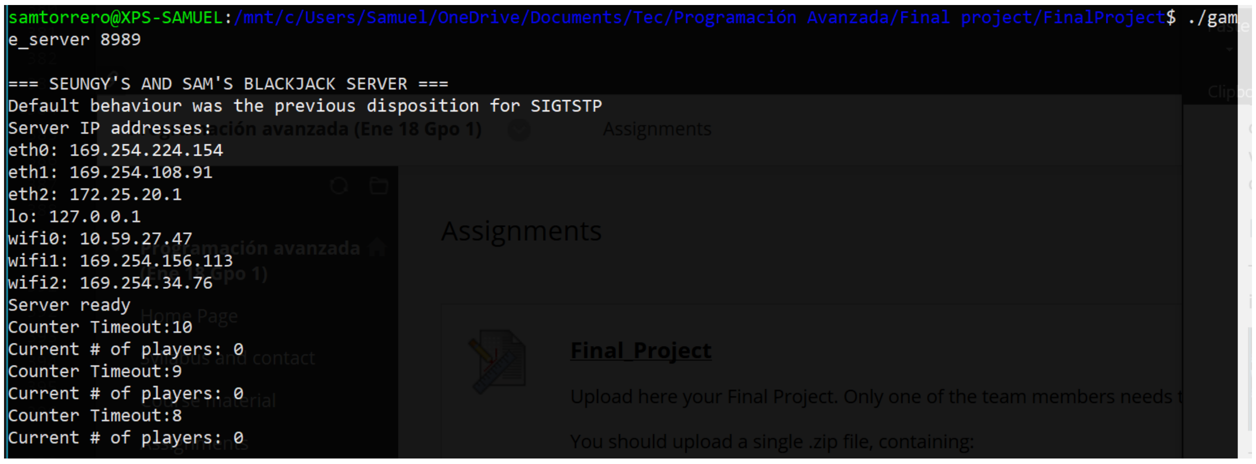
3. SFML works with Linux environments, so if the user has a Windows PC it is needed a graphic server (Xming in this case) for the game to show and the Linux Subsystem for Linux, to compile SFML. To make the graphical server run it is necessary to tell the terminal to use that as a display to output the graphical SFML using the following line:

```
export DISPLAY=:0
```

To run the server program the following line must be typed in the terminal:

```
./game_server <port_number>
```

Then hit enter. The result is to get the server code running, and should look like this:



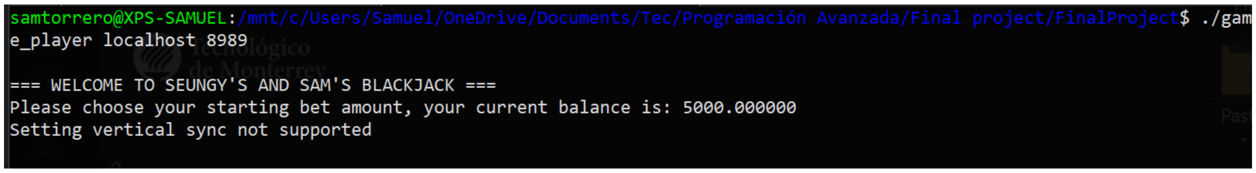
```
samtorrero@XPS-SAMUEL:/mnt/c/Users/Samuel/OneDrive/Documents/Tec/Programación Avanzada/Final project/FinalProject$ ./gam
e_server 8989

=== SEUNGY'S AND SAM'S BLACKJACK SERVER ===
Default behaviour was the previous disposition for SIGTSTP
Server IP addresses:
eth0: 169.254.224.154
eth1: 169.254.108.91
eth2: 172.25.20.1
lo: 127.0.0.1
wifi0: 10.59.27.47
wifi1: 169.254.156.113
wifi2: 169.254.34.76
Server ready
Counter Timeout:10
Current # of players: 0
Counter Timeout:9
Current # of players: 0
Counter Timeout:8
Current # of players: 0
```

Then we need to move out to the second terminal to get the player program running. There's no need to compile the programs again. To run the player program the following line must be typed into the terminal:

```
./game_player localhost <port_number>
```

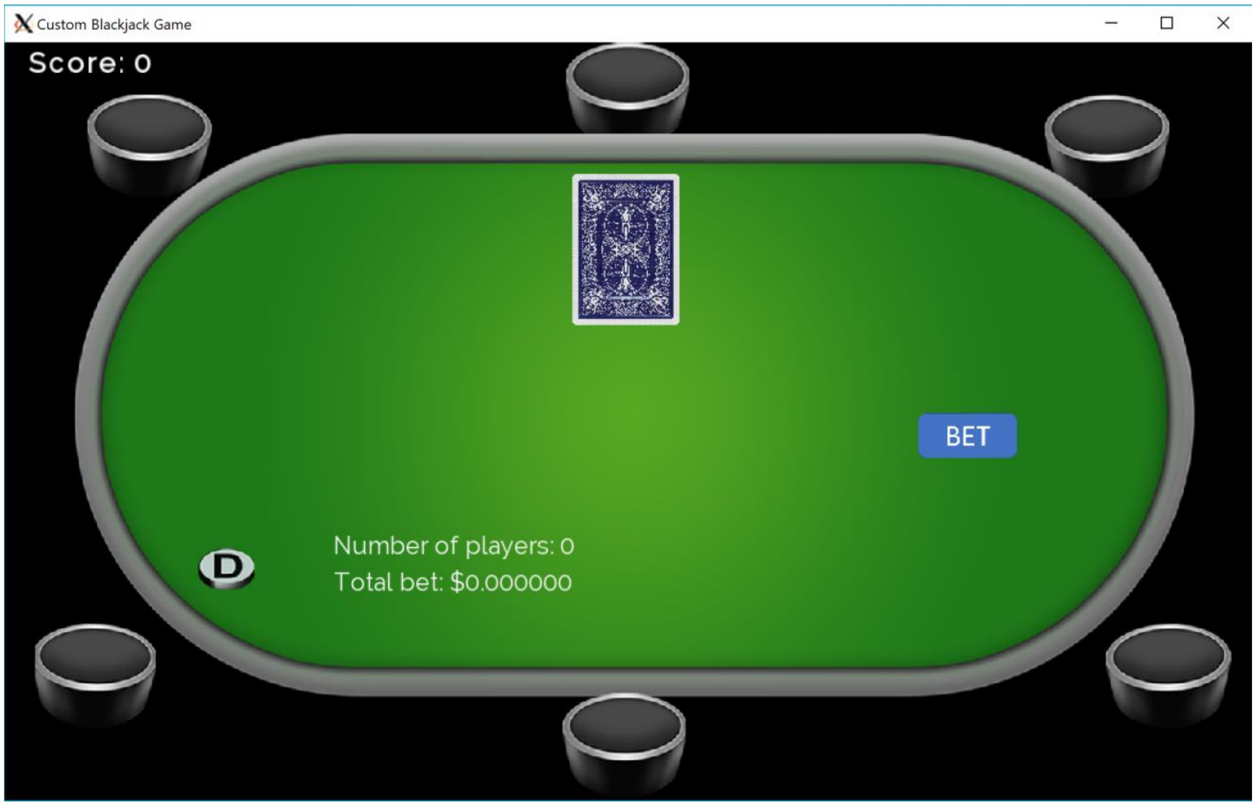
Then hit enter. The output should be the following:



```
samtorrero@XPS-SAMUEL:/mnt/c/Users/Samuel/OneDrive/Documents/Tec/Programación Avanzada/Final project/FinalProject$ ./gam
e_player localhost 8989

=== WELCOME TO SEUNGY'S AND SAM'S BLACKJACK ===
Please choose your starting bet amount, your current balance is: 5000.000000
Setting vertical sync not supported
```

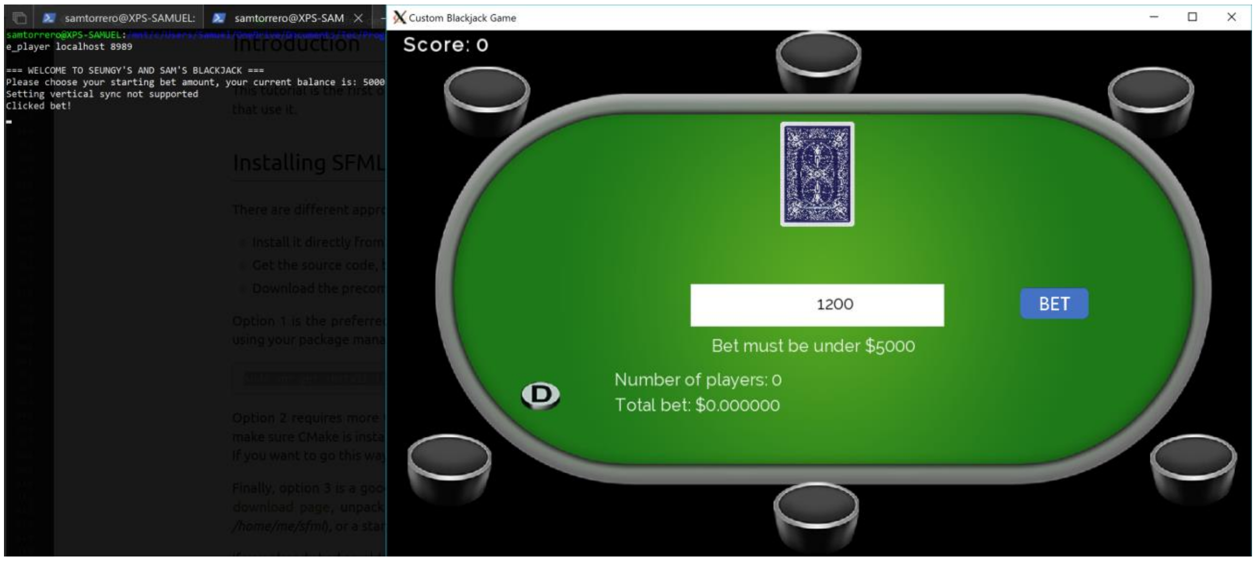
After running that line, a new window showing the game interface will appear:



With that the game is ready to be played. The interface will show the action buttons at the right, the score at top left; and the number of players and total bet in the bottom left area of the table. Cards and inputs are shown in the center.

GAME PLAY

To play the game start by typing the bet button and making a bet:

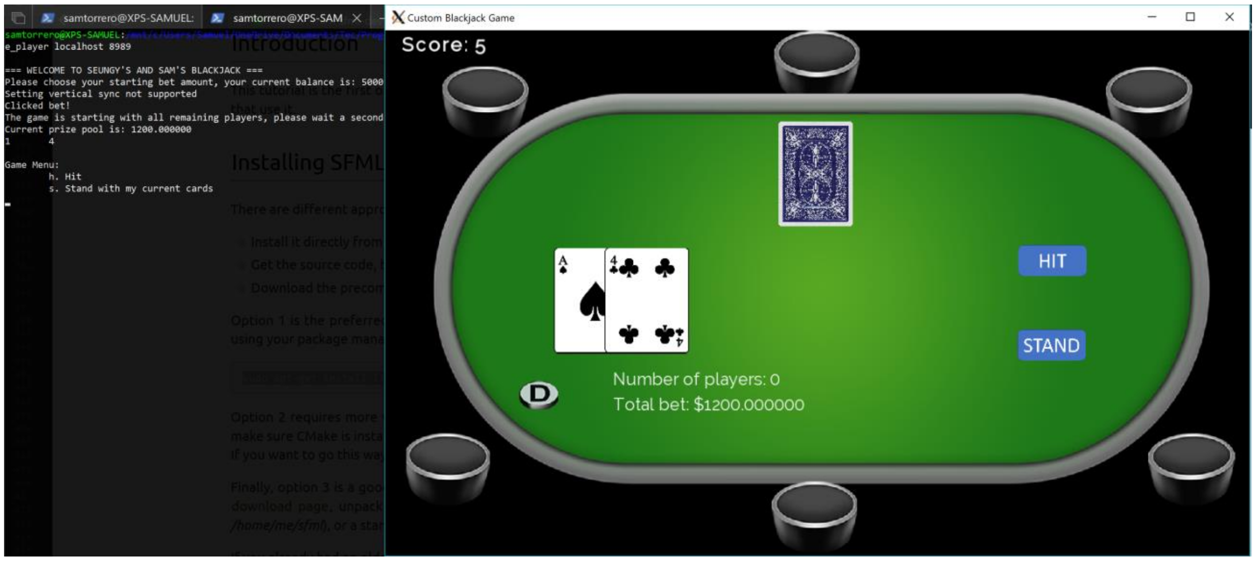


```
samtorrero@XPS-SAMUEL:
e_player localhost 8989

=== WELCOME TO SEUNGY'S AND SAM'S BLACKJACK ===
Please choose your starting bet amount, your current balance is: 5000
Setting vertical sync not supported
Clicked bet!
```

Actions and outputs will be shown on the terminal for confirmation. Hit enter to accept the bet.

After hitting enter the two initial cards are delivered to the player. He gets to choose between hitting to get another card or standing to move to the next phase with the cards he has:

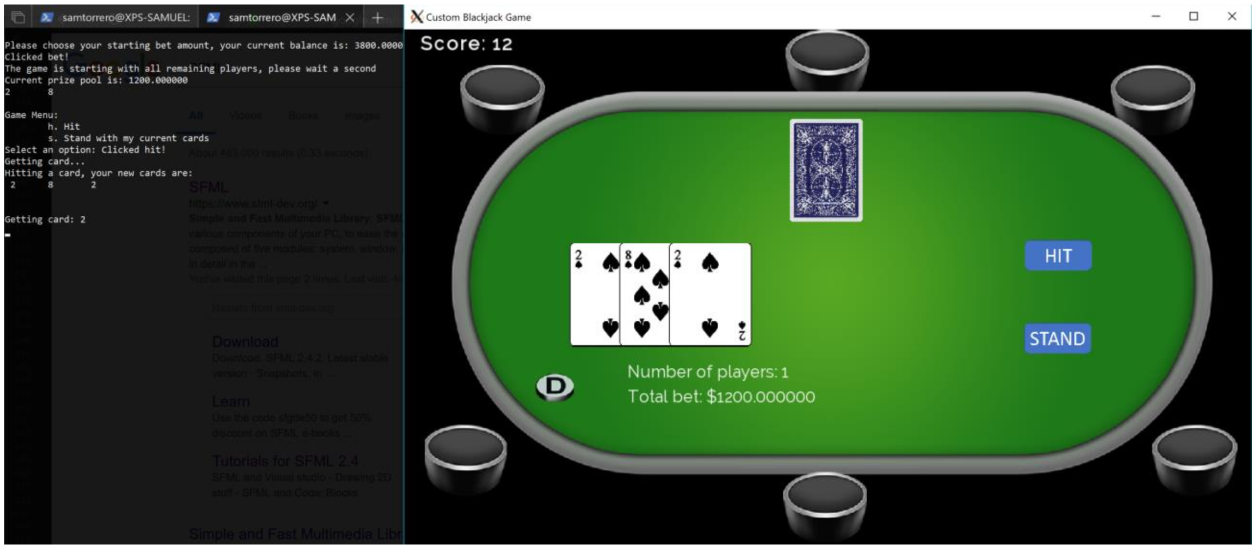


```
samtorrero@XPS-SAMUEL:
e_player localhost 8989

=== WELCOME TO SEUNGY'S AND SAM'S BLACKJACK ===
Please choose your starting bet amount, your current balance is: 5000
Setting vertical sync not supported
Clicked bet!
The game is starting with all remaining players, please wait a second
Current prize pool is: 1200.000000
1       4

Game Menu:
        h. Hit
        s. Stand with my current cards
```

If the player hits, a card is delivered:

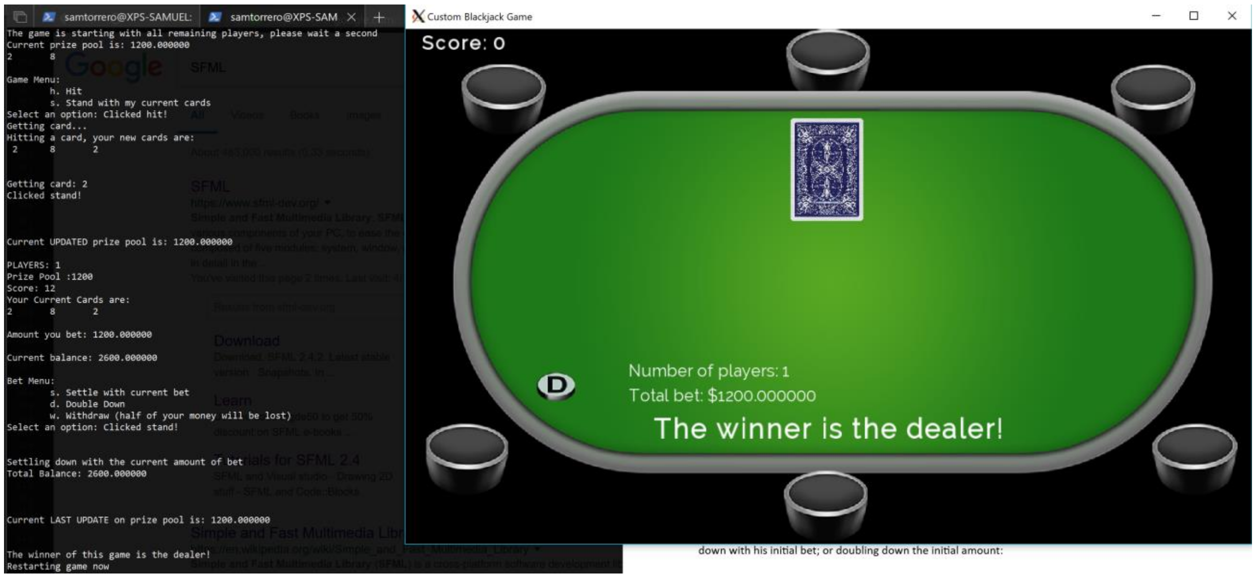


```
Please choose your starting bet amount, your current balance is: 3800.0000
Clicked bet!
The game is starting with all remaining players, please wait a second
Current prize pool is: 1200.000000
2       8

Game Menu:
        h. Hit
        s. Stand with my current cards
Select an option: Clicked hit!
Getting card...
Hitting a card, your new cards are:
 2      8       2


Getting card: 2
```

When the players choose to stand the game moves to the next phase, where the player gets to choose between withdrawing from the game, which will give him back only the half of this initial bet; standing or settling down with his initial bet; or doubling down the initial amount:



```
The game is starting with all remaining players, please wait a second
Current prize pool is: 1200.000000
2       8

Game Menu:
        h. Hit
        s. Stand with my current cards
Select an option: Clicked hit!
Getting card...
Hitting a card, your new cards are:
 2      8       2


Getting card: 2
Clicked stand!



Current UPDATED prize pool is: 1200.000000

PLAYERS: 1
Prize Pool :1200
Score: 12
Your Current Cards are:
2       8       2

Amount you bet: 1200.000000

Current balance: 2600.000000

Bet Menu:
        s. Settle with current bet
        d. Double Down
        w. Withdraw (half of your money will be lost)
Select an option: Clicked stand!


Settling down with the current amount of bet
Total Balance: 2600.000000



Current LAST UPDATE on prize pool is: 1200.000000


The winner of this game is the dealer!
Restarting game now
```

Finally, there is a message showing the winner. Wins whoever has the higher score. If the dealer has a score over 21 then the player with the highest score wins. If both, the dealer and the player have scores over 21, then by default the player wins. If, in that previous case, there is a player that has a score which is less than 21, then he will be the winner.

The game restarts after that and the process repeats until the game window is closed or the game server is stopped. After the game window is closed, the server waits for all clients to be closed and then the server itself needs to be closed.