Hao Huynh

Professor Mark Lehr

CIS 17B – 48941

10 December 2015

Individual Project: Poker Game (GUI).

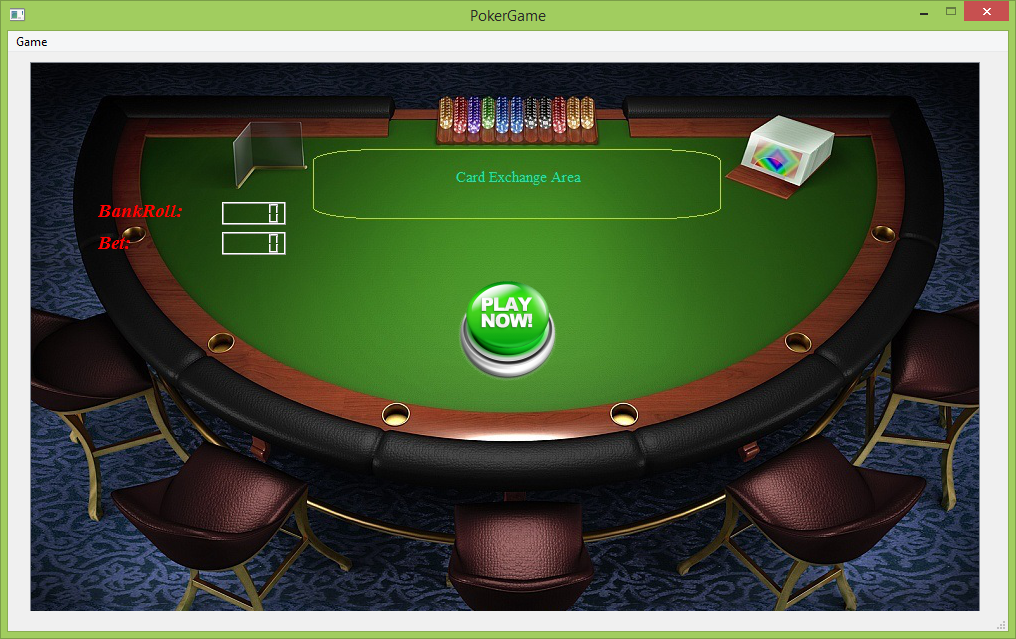
1. Introduction:

This Poker Game is a simple form of the standard gambling Poker card games. The purpose of this game is to find a winning hand, and the player has to play one-on-one against the dealer. The player is given five random cards at the beginning. Then, that player can choose to exchange any number of cards from the current given hand. After that, all cards on both hands of the player and the dealer will be scored by Poker rules being explained in the description. Basically, the player may win the game if the final five cards are in one of the below winning conditions:

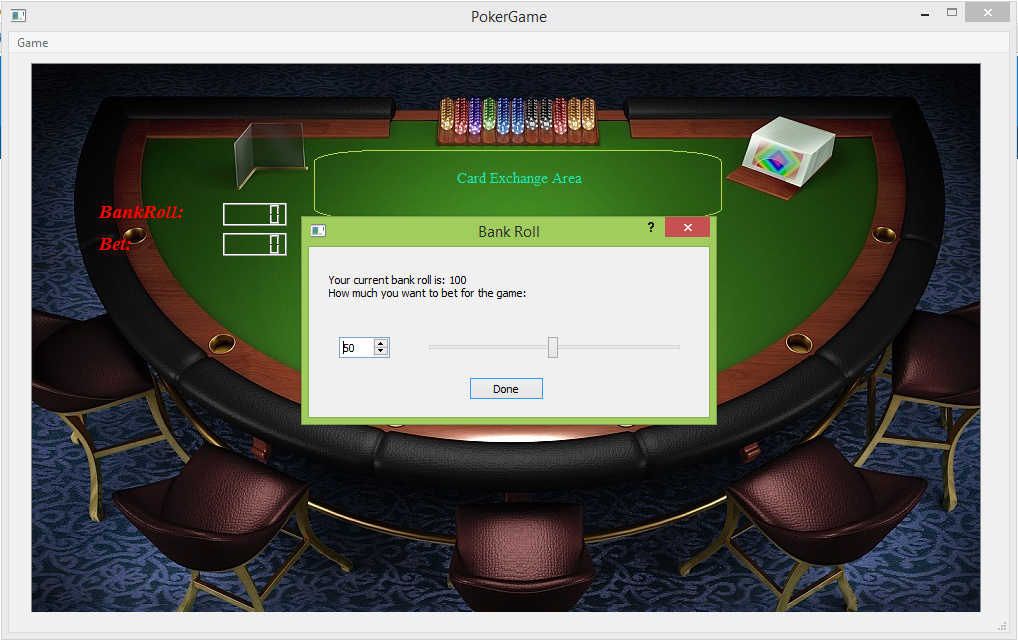
|  |  |  |
| --- | --- | --- |
| Poker Rank | Base Score | Comments |
| Royal Flush | 10000000 pts | A Straight Flush with an ACE |
| Straight Flush | 9000000 pts | A combination of  Flush and Straight |
| Four of the Same Kind | 8000000 pts | by rank |
| Full House | 7000000 pts | There are three cards in a rank and the others in another rank |
| Flush | 6000000 pts | All five cards are in the same suit |
| Straight | 5000000 pts | All five cards increase continuously by rank |
| Three of the Same Kind | 4000000 pts | by rank |
| Two Pairs | 3000000 pts |
| A Pair | 2000000 pts |

I decided to implement this project because I could take advantage of the Poker game in console mode. Moreover, the algorithms behind the screen were not complicated, and this game didn’t require many graphics affects and UI events. Lastly, I could demonstrate the beauty of the Object Oriented Programming.

1. Summary:
   1. The program statistics:
      1. The number lines of code (LOC) are about 2245 lines.
      2. There are 28 main variables.
      3. There are about 59 constants.
      4. There is 8 constructors and 8 destructors.
      5. There are about 75 methods.
   2. Miscellaneous: This program took me approximately 72 hours for coding and fixing errors. Moreover, I had learned about the Qt OOP design.
2. Description:
   1. The program problems:
      1. Mange 52 cards and apply Poker rules for a game.
      2. Create a graphics user interface for the game.
      3. Connecting UI components as well as the TCP and MySQL clients.
   2. The program solutions:
      1. Create the Poker Card Table class.
      2. Take advantage of Qt libraries such as: QMainWindow, QDialog, QGraphicsScene, QImage, QMovie, QGraphicsPixmapItem and other minor sub libraries.
      3. Using Model – View – Controller design pattern.
   3. Sample Input/Output:
      1. The Poker welcome scene with a background music:



* + 1. After clicking Start Now! the player can choose a bet roll before playing the Poker:



* + 1. The player gets the first five Poker cards:



* + 1. The player chooses any number of the first five ones for replacing by moving the cards and clicking on the animated button near the top right corner:



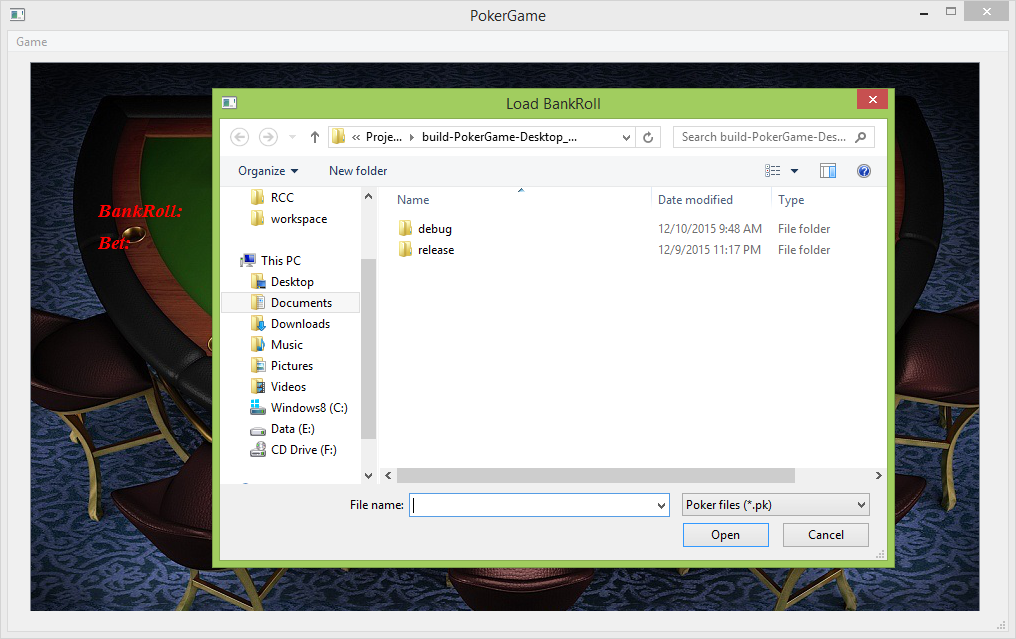
* + 1. The final result of Poker:



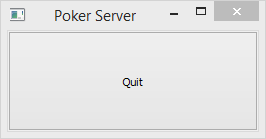
* + 1. The player continues on a new game:



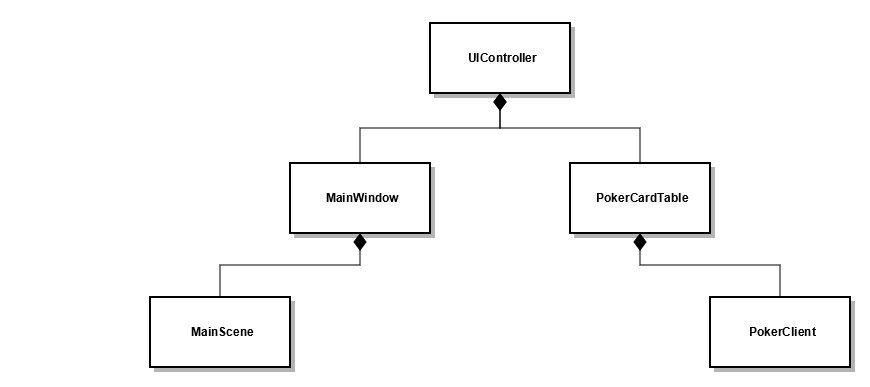
* + 1. The player choose the Load menu option:



* + 1. The PokerTCPServer that handle MySQL database business:



* 1. Flowchart and Pseudo code:
     1. Class design:



* + 1. [Replace card flowchart](FlowCharts/replacecards.jpg).
    2. PokerCardTable::isRoyalFlush():

If the highest card rank is an Ace, and the final five cards make a Straight Flush, then return true, else return false.

* + 1. PokerCardTable::isStraightFlush():

If the final five cards make a Flush as well as a Straight, then return true, else return false.

* + 1. PokerCardTable::isFlush():

If the lowest card suit equals to the highest card suit, then return true, else return false.

* + 1. PokerCardTable::isStraight():

In the final five cards, if one of the below three cases happens:

* {Ten, Jack, Queen, King, Ace}
* {Two, Three, Four, Five, Ace}
* Five cards are increased continuously

Then, return true, else return false.

* + 1. PokerCardTable::isFourOfAKind():

If the first four cards or the last four cards of the final five poker cards are in the same rank, then return true, else return false.

* + 1. PokerCardTable::isFullHouse():

In the final five poker cars, if the first three cards are in the same rank, and the last two cards are in another same rank; or if the last three cards are in the same rank, and the first two cards are in another same rank; then return true, else return false.

* + 1. PokerCardTable::isThreeOfAKind():

In the final five poker cards, if the first, the middle, or the last three cards are in the same rank, then return true, else return false.

* + 1. PokerCardTable::isTwoPairs():

In the final five poker cards, if there are two pairs in one of the below three cases:

* 2 + 2 + 1
* 2 + 1 + 2
* 1 + 2 + 2

Then, return true, else return false.

* + 1. PokerCardTable::isPair():

In the final five poker cards, if there is a pair in one of the below four cases:

* 2 + 1 + 1 + 1
* 1 + 2 + 1 + 1
* 1 + 1 + 2 + 1
* 1 + 1 + 1 + 2

Then, return true, else return false.

* + 1. PokerCardTable::calPokerScore():
       1. Sort cards by rank and by suit.
       2. Depends on the 9 winning conditions, calls the equivalent functions to get the score.
    2. PokerCardTable::getHandScore():

The total score of cards in a hand equals to

10^4\*the-first-highest-rank + 10^3\*the-second-highest-rank

+ 10^2\*the-third-highest-rank + 10^1-the-fourth-highest-rank

+ the-lowest-rank

* + 1. PokerCardTable::getRoyalFlushScore():

The total Royal Flush score of cards in a hand equals to the basic Royal Flush score (10000000 pts) + the hand score.

* + 1. PokerCardTable::getStraightFlushScore():

The total Straight Flush score of cards in a hand equals to the basic Straight Flush score (9000000 pts) + the hand score.

* + 1. PokerCardTable::getFlushScore():

The total Flush score of cards in a hand equals to the basic Flush score (6000000 pts) + the hand score.

* + 1. PokerCardTable::getStraightScore():

The total Flush score of cards in a hand equals to the basic Straight score (5000000 pts) + the hand score.

* + 1. PokerCardTable::getFourOfAKindScore():

The total Four of a Kind score of cards in a hand equals to its basic score (8000000 pts) + the rank of the middle card.

* + 1. PokerCardTable::getFullHouseScore():

The total Full House score of cards in a hand equals to its basic score (7000000 pts) + the rank of the middle card.

* + 1. PokerCardTable::getThreeOfAKindScore():

The total Three of a Kind score of cards in a hand equals to its basic score (4000000 pts) + the rank of the middle card.

* + 1. PokerCardTable::getTwoPairScore():

The total Two Pair score of cards in a hand equals to its basic score (3000000 pts) + 10^2\*The-First-Highest-Two-Pair + 10\*The-Second-Highest-Two-Pair + Low-Card

* + 1. PokerCardTable::getAPairScore():

The total A Pair score of cards in a hand equals to its basic score (2000000 pts) + 10^3\*A-Pair + 10^2\*First-High-Card + 10\*Second-High-Card + Low-Card

* 1. Variables:

|  |  |  |
| --- | --- | --- |
| Type | Name | Location |
| vector<Card\*> | crCards  (The player’s current cards) | class PokerCardTable |
| vector<Card\*> | crDCrds  (The dealer’s current cards) |
| vector<Card\*> | crSRnks  (The player’s current cards sorted by rank) |
| vector<Card\*> | crSSuits  (The player’s current cards sorted by suit) |
| PokerClient\* | pokClnt  (The Poker client that manages the TCP connections to the Poker server) |
| QMediaPlayer \* | bgMusic  (The media player of the back ground music) | class UIController |
| MainWindow \* | mainWindow  (The main window of Poker Game) |
| BankRoll\* | bankroll  (The BankRoll dialog of Poker Game) |
| PokerCardTable\* | pokTble  The Porker Card Table object |
| Ui::MainWindow \* | ui  (Main UI) | class MainWindow |
| MainScene \* | mainScene  (The fundamental graphics scene of the game) |
| PokerPushButton\* | playBtn  (The play button) |
| PokerPushButton\* | dealBtn  (The replace button) |
| QImage | bgImage  (The Background Image Object) | class MainScene |
| QVector<QGraphicsPixmapItem \*> | crdImgs  (A vetors of all possible card images) |
| QVector<QGraphicsPixmapItem \*> | crCrdIs  (A vetors of all five displayed card images) |
| QVector<QGraphicsPixmapItem \*> | crDCdIs  (A vetors of all five displayed card images belong to dealer) |
| QVector<QPoint> | crdPoss  (The default positions of five porker cards on the scene) |
| QVector<QPoint> | dCdPoss  (The default positions of five dealer's cards on the scene) |
| QGraphicsRectItem\* | exArea  (The exchanged area that used for replacing cards) |

* 1. Concepts:

|  |  |
| --- | --- |
| Concept | Location |
| 1. Widgets | * class BankRoll * class MainWindow |
| 1. Main Windows | * class MainWindow |
| 1. Dialogs | * class BankRoll |
| 1. Event Processing | * class MainWindow * class BankRoll * class UIController |
| 1. Graphics | * class MainScene * class PokerPushButton |
| 1. Containers | * class PokerCardTable * class MainScene |
| 1. File Input / Output | * class PokerHelper |
| 1. Data Base | * class PokerSocket |
| 1. Networking | * class PokerClient * class PokerTCPServer * class PokerSocket |
| 1. Regular Expression | * class PokerHelper |
| 1. Multimedia | * class UIController * class PokerPushButton |
| 1. Memory allocation | * main function. * class UIController * class MainWindow * class MainScene |
| 1. Memory deallocation | * class PokerCardTable |
| 1. Data structure | * Card in class PokerCardTable |
| 1. enum data type | * class PokerCardTable * class PokerHelper |
| 1. Functions with structures | * class PokerCardTable * class MainScene |
| 1. Arrays with structures |

1. Program (Header File List):
   1. “bankroll.h”
   2. “mainscene.h”
   3. “mainwindow.h”
   4. “PokerCardTable.h”
   5. “pokerclient.h”
   6. “pokerhelper.h”
   7. “pokerpushbutton.h”
   8. “uicontroller.h”
   9. “pokerserver.h”
   10. “pokersocket.h”