Project 3: Card Match Game

Cmpe 230 : Systems Programming, Spring 2023

Aslı Gök -2020400189
Nazlıcan Aka -2020400027
23.05.2023

1- Introduction

In this project, we implemented a game known as “Card Match” or “Pairs” using QT. Main target for the game is that the player tries to match all the cards with the same name in a given number of tries. If the player can match all the cards in the given number of tries, she will win the game. Otherwise, when the number of tries ends, the game is closed. However, in order to help the player, as developers of the game, we add some useful features to the game.

2- Design

Rules of the game:

You have 50 tries to match 30 cards. In other words, you have 15 different cards and you need to match them all in 50 tries in order to win. There is no limitation of time. The only limitation is the number of tries.

You can check your remaining tries from the “Number of Tries Remaining” label. Tries are started 50 and decreased each true or false match.

There is a “Hint” button which shows you one of the unmatched cards. But there is only one “Hint” you can use. After clicking the button once, the button will be disabled, so do not give “Hint” anymore.

You can quit the game anytime you want clicking the “x” button at the right top of the page. Additionally, you can make the game full screen.

There is a “New Game” button next to the “Hint” button. When you click the “New Game” button, the game starts from the beginning. In other words, you lose your past matches, and 15 cards are mixed again.

When you match the 30 cards in the 50 tries, you will win the game. When you win the game, the game is not closed by yourself. Therefore, you can start a “New Game” or close the game page.

You can keep track of your scores from the “score” label, scores increase when you find a match. As a result of 30 cards, the maximum score is 15.

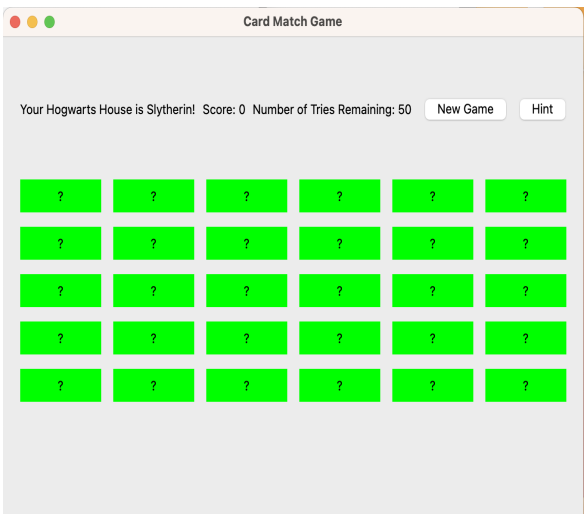
If your 50 tries are over, the game is closed automatically.

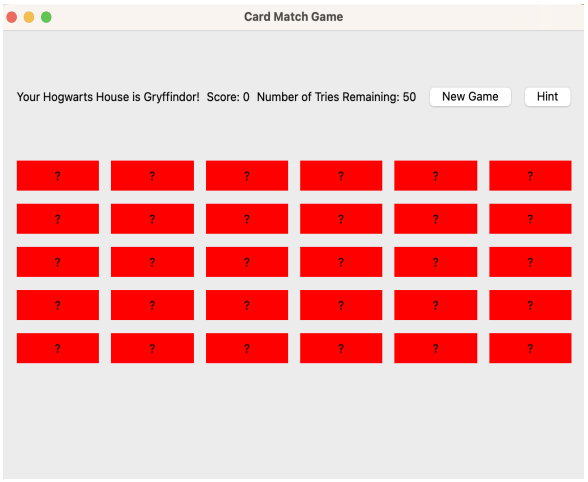
Concept of the game:

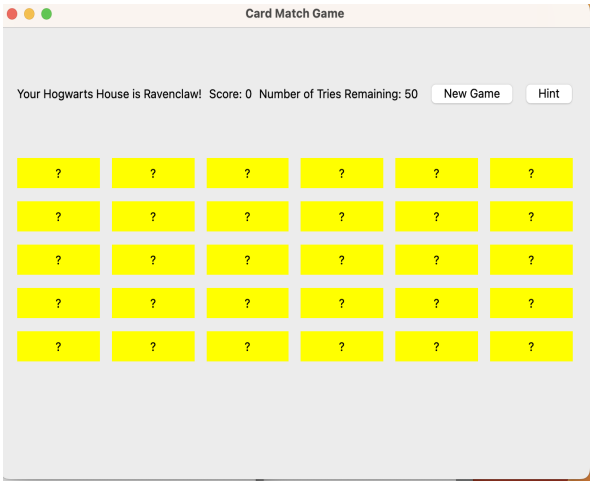
In order to make the game more fun and exciting, we used the “Harry Potter” Concept.

There are 15 “Harry Potter” characters’ names written on the cards, **"dobby", "hagrid", "mcgonagall", "harry", "hermione", "ron","snape", "dumbledore", "malfoy", "sirius", "bellatrix", "hedwig", "lupin".**

Another fun thing about the game is before starting the game the “Sorting Hat” will decide your on of the house of Hogwarts **"Gryffindor", "Hufflepuff", "Ravenclaw", "Slytherin".** The house is chosen randomly. But if you want some house so much, the hat will understand you. Each house has a different color concept. **"Gryffindor"** is “red”, **"Hufflepuff"** is “cyan”, **"Ravenclaw"** is “yellow” and **"Slytherin"** is “green”**.** Thus, there are 4 different views of the game scene.







3- Implementation

In this project 4 header files and 5 source code files are used. Header files include the instances of QObjects, slots and signals related to objects,source code files include implementations of those.

1)- card.h/ card.cpp

Card object has 3 instances: selectedCard queue that holds the clicked cards, color QString that indicates the color of the card, openText QString that has the character name.

Slots and descriptions:

same_cards(): checks whether the two clicked cards have the same character. It emits same_signal() when they are the same.

show_card(): show the clicked cards' character names, add them to selectedCard queue and turn them back by setting "?" text again.It emits check_cards() signal when two cards are clicked.

2)- grid.h/ grid.cpp

Grid is the main QObject that will help us to visualize the game scene. Grid object has 3 instances: selectedCard queue that holds the clicked cards, my_remaining_list that holds the remaining unmatched cards and hint boolean that will help us to make sure that the "Hint" button is used once in each game.
Slots and descriptions:

check_same(): checks whether the two clicked cards have the same character names obtaining them from selectedCard QQueue with the dequeue method.If the cards are matched it emits gridsame() signal, otherwise gridnot() signal. These signals will be used in main.cpp with connections later.

bad_lost(): this slot ends the game when the given 50 tries are used and display a QMessageBox.

new_game(): this slot starts a new game when the "New Game" button is clicked.

rem_card(): this slot iterates over the cards and checks whether they have been matched or not and appends unmatched ones to the my_remaining_list.

give_hint(): this slot shows a random character name for 25ms and turns it back ,chooses this random card from my_remaining_list in each game once. It emits

gridhint() signal when the “Hint” button is clicked, sets hint boolean true to make sure the “Hint” button is used once.

3)- score.h/ score.cpp

Score object has 2 instances: QLabel label that will display the score and int counter that will change during the game dynamically when a match happens.

increment() slot: This slot increments the counter if a match happens and when the score is 15 displays a QMessageBox that indicates the user won the game.

4)- try.h/ try.cpp

Try object has 2 instances: QLabel label that will display the score and int counter that will change during the game dynamically when two cards are clicked.

finished() slot: This slot decrements the counter if two cards are clicked and when 50 tries are over, it displays a QMessageBox that indicates the user lost the game. It emits lost() signal, later it will be used in the main.cpp with connection.

5)- main.cpp

In the main.cpp, there is main() function and QApplication qApp object that manages the program.Additionally needed labels, buttons, message box and other QObjects are declared.

The design of the central window is done here by adding the labels , spacer items, buttons and the grid.

After declaring the Grid object, we create the game design as 6x5 cards and assign character names randomly in nested for loops.

4- Conclusion

This project can be implemented with many additional functionalities, we chose to use a “Harry Potter” scenario to make the game more entertaining and add a hint button to give a little help to users. As the objective of the project is to make students familiar with the QT framework and C++ programming language, one can consider adding different labels and buttons.