CMPE 230
Systems Programming
Project 3 - Card Match Qt-Project

05/22/2023

Işıl Su Karakuzu **2020400144**
Atakan Yaşar **2020400099**

Table of Contents

1. Introduction

The objective of this project is to create a single player “Card Match” game using Qt and C++. This game aims to turn over all pairs of matching cards in 50 tries.

Game has a 5x6 grid. In this grid, there are 30 words as a result of having pairs of each 15 different words. At the beginning of the game, all cards are shuffled and turned down. The player is expected to click on two cards in two different places. Regardless of the correctness or inaccuracy of the selection made, the number of remaining tries is reduced. If the two cards chosen by the player match, those cards remain face-up, become unclickable and the player earns 1 point. If the selected cards do not match, the cards are closed after 0.75 seconds. The player can use his/her remaining tries until he/she wins or loses the game. The player can see his/her score and remaining rights from the top of the screen. When the “New Game” button is pressed, the score and tries are reset and the cards are shuffled again. The player cannot double-click a card in the same spot, select three cards in the same try, or rematch cards that have already been paired.

If all cards are not matched in 50 tries, the game is lost, and if 15 card pairs are matched, the game is won. When the game is over, a window opens depending on the winning and losing status. The player can start a new game by closing this window and clicking the "New Game" button. After losing, all cards are revealed for the player to see.

In the process of distributing the cards, 1 concept is randomly selected from 3 different concepts, 15 words are selected from approximately 100 words in this concept and placed on the grid as two of each unique word.

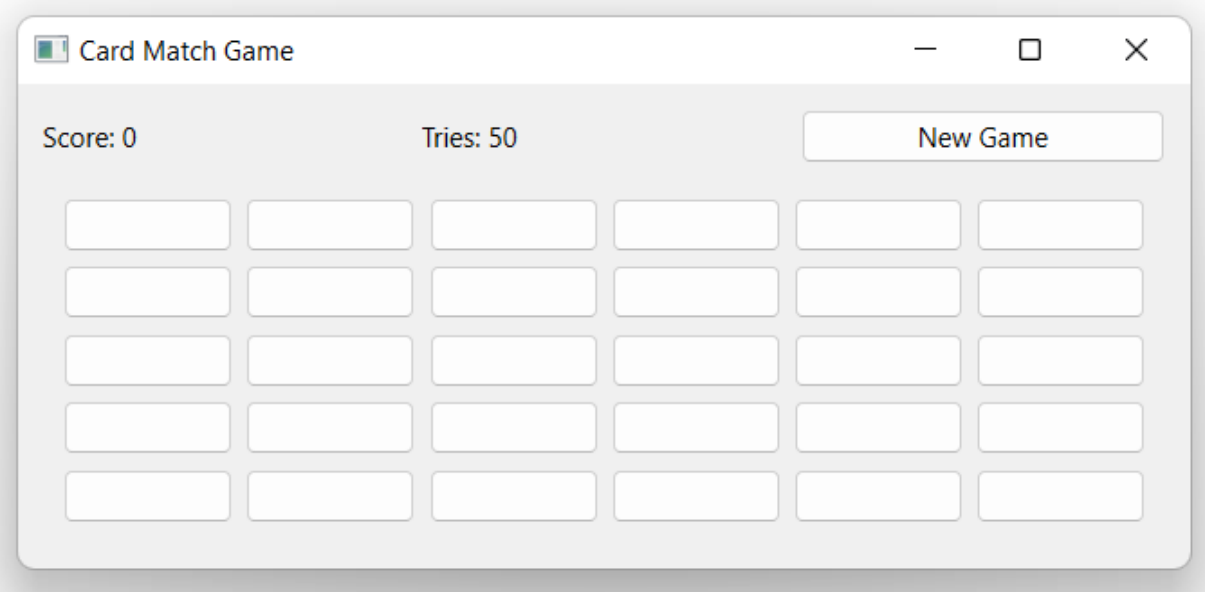
The solution contains 4 header, 5 source, 3 resource, and 1 *.pro file which will be explained in detail in later chapters.

2. Program Interface

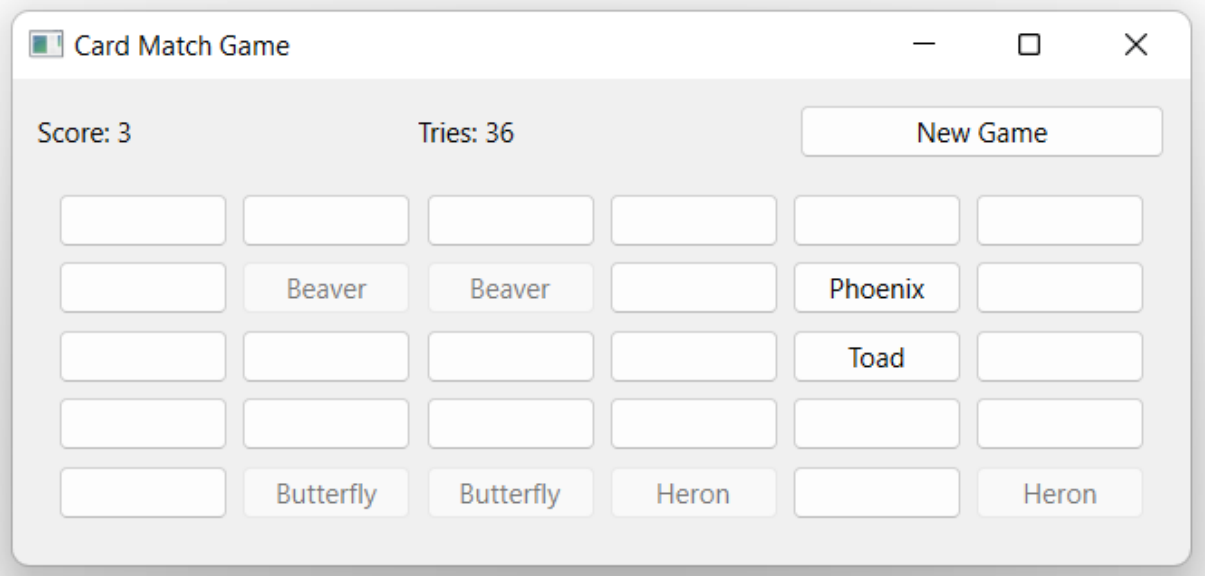
To run the program, you need to be sure that Qt is installed in your computer. Open Qt Creator, click open project and select the cmpe230-qt.pro file. After that, the project will be opened, wait for Qt Creator to index the project. Finally, click the run button or ctrl+r to run the project. After running, the game window opens and you can play the game. You can close the window to quit the program whenever you want.

3. Program Execution

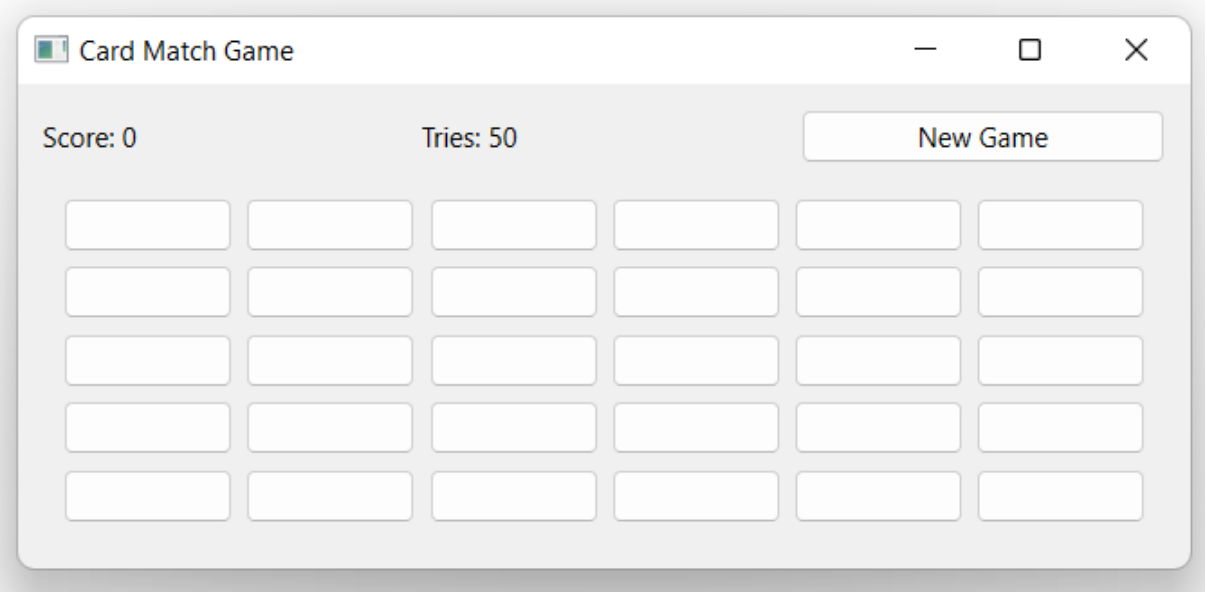
Game looks like this at the beginning.



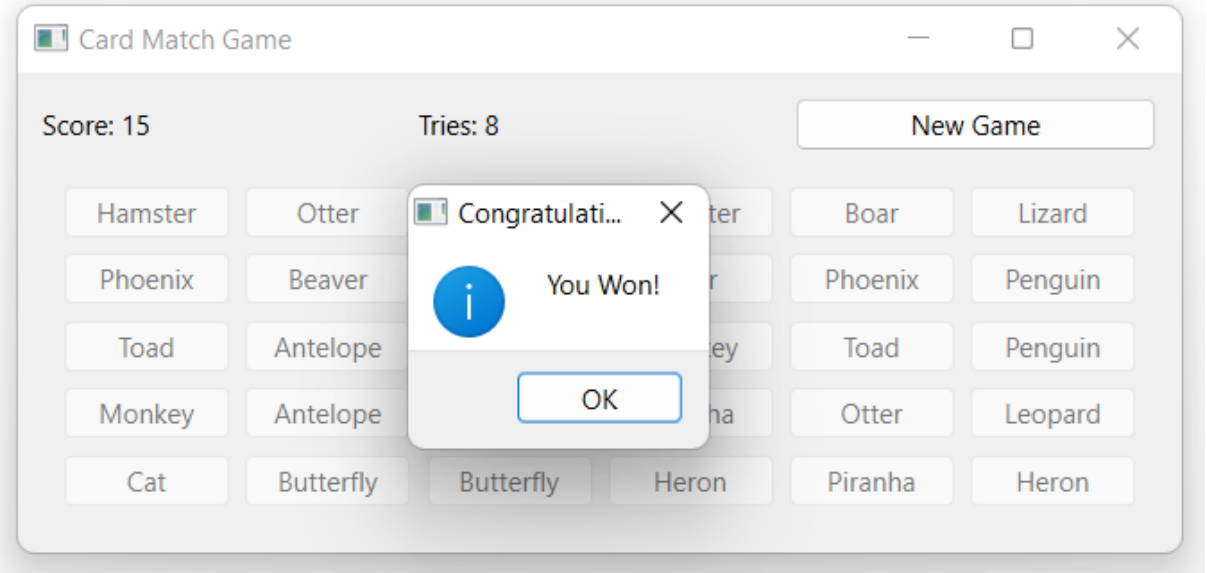
After playing the game for a while you can see the correct matches you have made dimly. Also you can see your current choices brightly for a while. You can check your score and tries all the time.



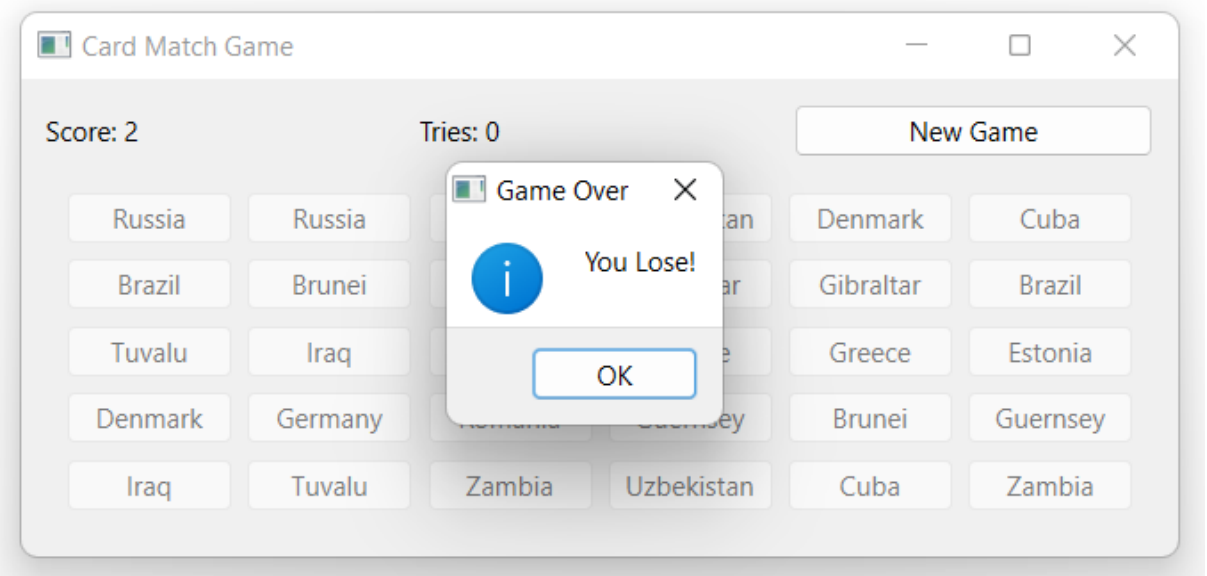
After clicking the “New Game” button, a new game is created.



When you match all the cards, you win.



If you can't match all the cards using the tries given to you, that is, when you finish your remaining tries, you lose. After you lose, all cards are opened for you to check.



4. Input and Output

Program only reads card texts from resource files and picks randomly which text pairs to be used. The program also detects clicks on buttons. Clicks on buttons trigger some events.

When the program runs, there will be a visual window as an output. There are labels and a push button at the top. There are also push buttons for cards closed at the beginning. A closed button means that text content of the card is hidden. At some time of the game, cards will be opened and that means the text content of the card is visible.

5. Program Structure

The project consists of this files:

- cmpe230-p3.pro
- main.cpp
- Shuffle.h
- MainGameWindow.h
- GridLayout.h
- CardButton.h

5.1. cmpe230-p3.pro

This ".pro" file contains commands that are required for building the project with qmake. Header, source and resource files are added here. Qt Creator

recognizes this like a project description. This file allows Qt Creator to build and run the project.

5.2. main.cpp

main.cpp is the first file to run. It calls MainGameWindow widget and runs it.

5.3. Shuffle Function

There are three files where the shuffle function reads possible card's text contents. One file consists of animal names, one file country names and the last one fruit names.

Firstly, the shuffle function randomly picks a file to read. Then, it chooses 15 objects to be placed as card contents. Finally, it shuffles all texts as a pair into the grid.

5.4. MainGameWindow Class

It creates a mainLayout which has one layout at the top and one widget at the bottom. The top layout includes score and remaining time labels, also a push button to start a new game. The bottom widget contains a grid widget including GridLayout class.

New game button can detect clicks. The click event is connected to a restart function. When the new game button is clicked, everything restarts. Restart function rebuilds GridLayout class and resets thescore and remaining time labels in the top layout.

Win/lose game state is handled here. Winning the game pops up a window saying "You won!". Likewise, Losing the game pops up a window saying "You lose!". Besides, when losing the game, all cards will be opened.

5.5. GridLayout Class

The class keeps the game score, remaining tries, first clicked button, second clicked button and a timer.

In the constructor of the class, a QGridLayout is set as the layout of this GridLayout widget. Then, each cell of the grid is filled with empty CardButton objects. After that, the build function sets values of every feature and objects inside. Finally, the connection of the timeout signal of the timer and match function are linked.

The build function sets the score and the remaining tries to their initial value. First and second clicked buttons are set as null. Card texts are created by the Shuffle function. Then, created card texts are placed into CardButtons inside of cells.

When the click handler for the grid is called, it checks if the first clicked button exists. If there is no first clicked button, it defines the clicked button as the first clicked button. If there is a first clicked button, it checks if the second clicked button has not been defined yet. If not defined, it defines the clicked button as the second clicked button. If the second clicked button is assigned, the timer starts to run. Since the first clicked button and the second clicked button are defined until time is out, there will be no permission for a third click.

```
clickHandlerForGrid(button) {
    if first_clicked is null:
        first_clicked = button
    else if second_clicked is null:
        second_clicked = button
        start the timer
}
```

When the timer runs out, the matching handler is called. The matching handler checks if the content of the first clicked button and the second clicked button matches. If there is a matching, both buttons will be deactivated and score is increased. If there is no matching, buttons will be unclicked. After the checks, the first and second clicked buttons are defined as null and the number of tries is decreased. Then, score and remaining tries labels are updated. At the end, the handler checks whether the current state is an ending state. If there are no remaining tries left or all cards are matched, the game ends.

```
matchingHandler() {
    if first_clicked matches second_clicked:
        deactivate both buttons
        increase score
    else:
       unclick both buttons
    set buttons as null
    decrease the number of tries
    update score and remaining tries labels
    check game is finished
}
```

5.6. CardButton Class

CardButton class extends from QPushButton. There are active and clicked booleans to track the status of the card. The button is not clickable when it is not active or it is clicked. Also, two cards are in clicked status, the button is not clickable.

Click event onto the button connected to a click handler function. When a button is clicked, the click handler function is called. It will not do anything if the button is not clickable. If the button is clickable, the button will be marked as clicked and the text content of the card will be opened. Then, the click handler for the grid is called.

Unclick function marks clicked boolean as unclicked and conceals the text content of the card.

Deactivate function marks active boolean as inactive and the button does not accept any click event then.

6. Examples

Lets play a game with 10 tries and 2x3 grid. This example is made for understanding the game, the real game will include 50 tries and a 5x6 grid. You can check the Program Execution chapter for detailed pictures. Watch the video to see an example of the game.

Example Video of the Game

7. Improvements and Extensions

To give an example of the strong points of this project, it will not be difficult to add new features since the project was developed in accordance with OOP principles. In addition to this feature, the design of this project in accordance with clean code principles makes the code readable. This design style also makes it easier to test the code for bugs and allows us to get a more reliable and bug-free result.

As for its weak features, the project could have been more appealing to the eye. The design and beautification part of the job was a bit weak. In addition, the game does not contain much functionality, timer, and multiplayer features can be added to the game. In addition, the resources provided for cards can be enlarged, more concepts

can be added, picture cards can be added instead of just text. Features such as changing the grid size, increasing or decreasing the total tries can be added to the game. It can be integrated into levels from easy to difficult by creating a level system.

8. Difficulties Encountered

Qt was difficult to download and install. Qt Creator was paid to use, so we considered opting for other IDEs. But this time, we couldn't set the necessary configuration files to run the program. After hesitations about which helper to configure with, we decided to run our code with CMakeLists.txt at first. While we were running our code, we had trouble in the continuation because it is a difficult system to handle. We saw that the reason for some of our bugs was that we wrote CMakeLists.txt wrong, we had a hard time with its hard-to-understand structure. In addition, we tried to change our program to be configured with qmake, since the person who gave the project wanted configuration with qmake. While doing this, we saw that CMakeLists.txt and qmake caused the files to be built differently, and also caused significant conflicts in the program's operation. We had a hard time finding a good IDE that supports qmake and satisfies our needs.

After all these troubles, we decided to drop everything and use Qt Creator. We got a student license for Qt Creator because it's paid to use. When we couldn't run our project in Qt Creator, we had to rewrite the project in Qt Creator with a new *.pro file. This took a lot of time and effort and was the hardest part of the project.

Another challenge for us was that Qt is a new technology for us. We caused a lot of errors and bugs while establishing connections, writing functions and connecting them, and we spent a lot of effort to fix them. Another point is that we have not done an OOP project with C++, so we had a hard time understanding the requirements of this language.

9. Conclusion

In conclusion, we successfully developed a “Card Match” game with interactive GUI using Qt and C++.

This project taught us how to design a project in accordance with C++ and OOP principles and Qt infrastructure.

Although we encountered difficulties, we came through them by researching. Progress in a systematic way helped us a lot. Doing teamwork reduced the workload and helped the project move faster by adding different perspectives.

10. Appendices

The source code is provided as an extra ZIP file because of its size.