CS 319 - Object-Oriented Software Engineering

Final Report

Q-Bitz

Group 1F

Burak Yaşar
Görkem Yılmaz
Hikmet Demir
Mert Duman
Mert Alp Taytak

Supervisor: Eray Tüzün

Contents

1. Introduction

The main purpose of the first iteration was to provide the basic functionality of the game. The project is implemented using Java FX library as stated in the design report.. The first task is to get familiar with Java FX library. Also we decided to use some extra libraries such as Java FXyz which is 3D Visualization and Component Library for Java FX and Java FXtras.

In the implementation, game opens with its main menu. Until the demo we are planning to show basic functionalities of our game that includes such screen transitions as Main Menu to Game Screen, Main Menu to How to Play etc. Also, now we can turn the three dimensional cube around all three axes. We can generate random patterns and fill the board using Cube faces, at the end of the game it announce either win or lose based on whether the original pattern is match with the pattern which is built by the player. As it is stated in the previous reports, there are two different types of games which are singleplayer game and multiplayer game and for now, we have implemented the singleplayer mode for the demo. In this type, a single player finishes the given pattern with sample cube and game finishes.

Multiplayer option can be chosen but its implementation and the server and the database will be handled in the second iteration.

After the implementation of the opening screen is finished, we make it sure that all buttons work and open the necessary scenes and therefore we wrote the .fxml files needed for that purpose. After that, the game board is created and grids have been drawn for 3x3, 4x4 and 5x5 games. While the GameUIController class controls the game, panes for the cube faces and sample pattern had been inserted into the game board.

Remaining classes had been created and necessary buttons had been created but the functionality of them will be implemented during the time between the demo and the next iteration.

Each of our group members use IntelliJ IDEA as the development environment along with the Github. Since IntelliJ provides easier connection to the Git and it suggest preselected methods and useful practices and shortcuts we use this IntelliJ. Moreover, Jetbrains which is the developer company of Intellij is also supply educational licenses for students to use ultimate version and because of the ultimate version has more beneficial shortcuts and automatic sentence filling, it was good for us to use it.

2. Design Changes

Not all of the functionalities of the game are provided and implemented since this is the first iteration. However some design changes are made as the following:

2.1 Added Model Classes

-PatternPack: this class is added to handle pattern packs and creation of cubes and patterns
-Cuboid: to rotate the cube this class called it uses JavaFxyz library properties to provide 3D effects
-Player: this class is added to model a user.
-ResourceLoader: this class is added to load images and patterns from the resource files.

3. Lessons Learnt

As we talked and determined that each member of our group will work in collaboration of every stage of the development, we tried to distribute the workload fairly. We both met in person and online using Google Drive or Whatsapp when we could not meet together in person. Although there were cases that we try to divide the workload to the two subgroups as two of us stay in dormitory and three of us stay in outside of the campus, at least one person checks and contributes the other job which is assigned to other subgroup.
We realized that how hard arranging a meeting is. Since we all have different schedule it is hard to set meeting times where all members of the team are available.
We observed that proper diagrams made the task easy for the implementation and the development. These UML diagrams also helped in communication and abstract modeling, it increases understandability.

4. User’s Guide

4.1 System Requirements & Installation

Installation:
Q-Bitz will be played without installation because it runs with an executable jar file.
System Requirements:
Q-Bitz implemented in Java. Graphics will be implemented by using JavaFX libraries.
Minimum System Requirements
- Any platform that supports Java is supported
- Java 8 should be downloaded
- 1.2 Ghz Dual Core processor
- 256 MB RAM or higher
- Minimum of 1400 x 900 screen resolution
- 5 MB hard-disk space

Recommended System Requirements

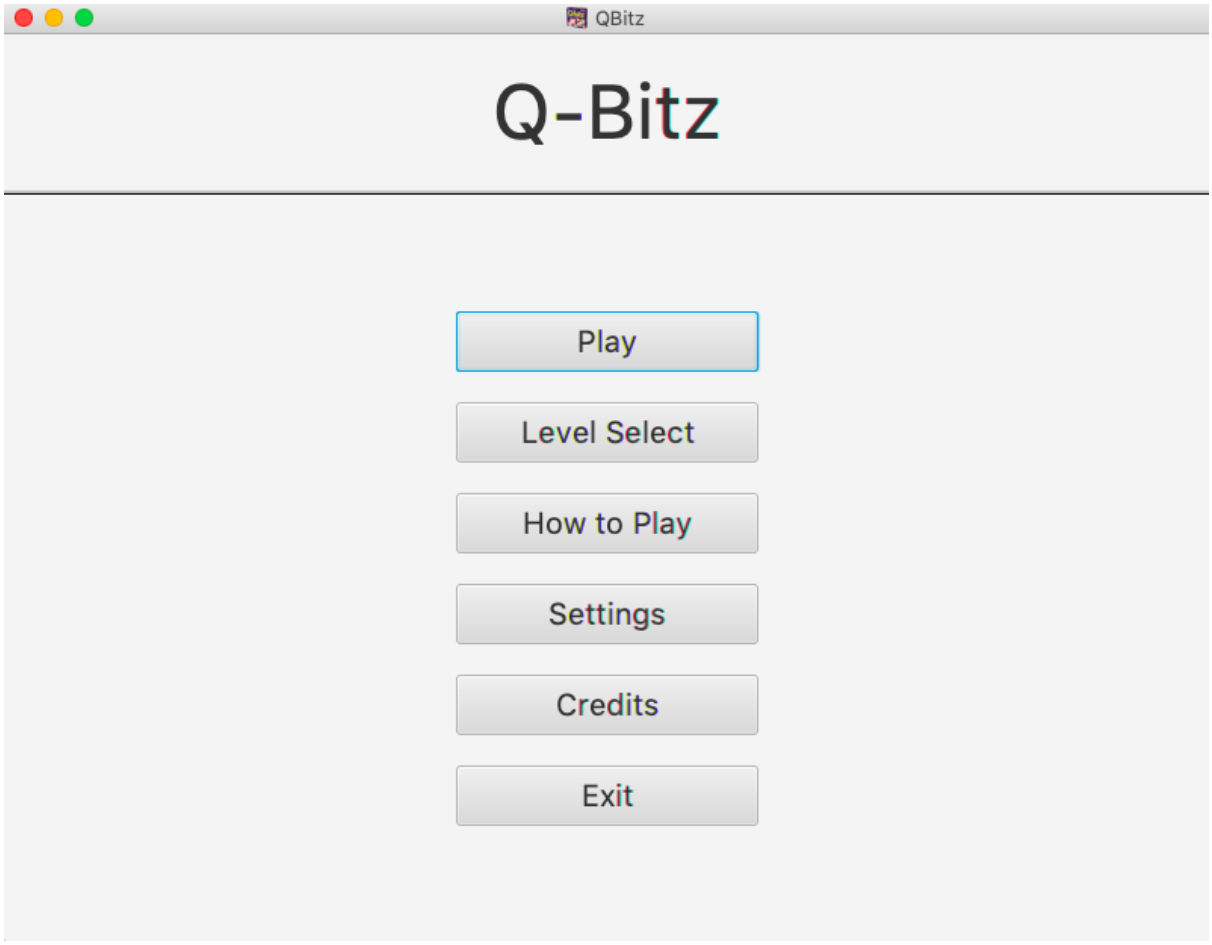
- 2.0 Ghz Quad Core Processor
- 1 GB RAM or higher
- Minimum of 1920x1080 resolution
- 20 MB hard-disk space

Installation

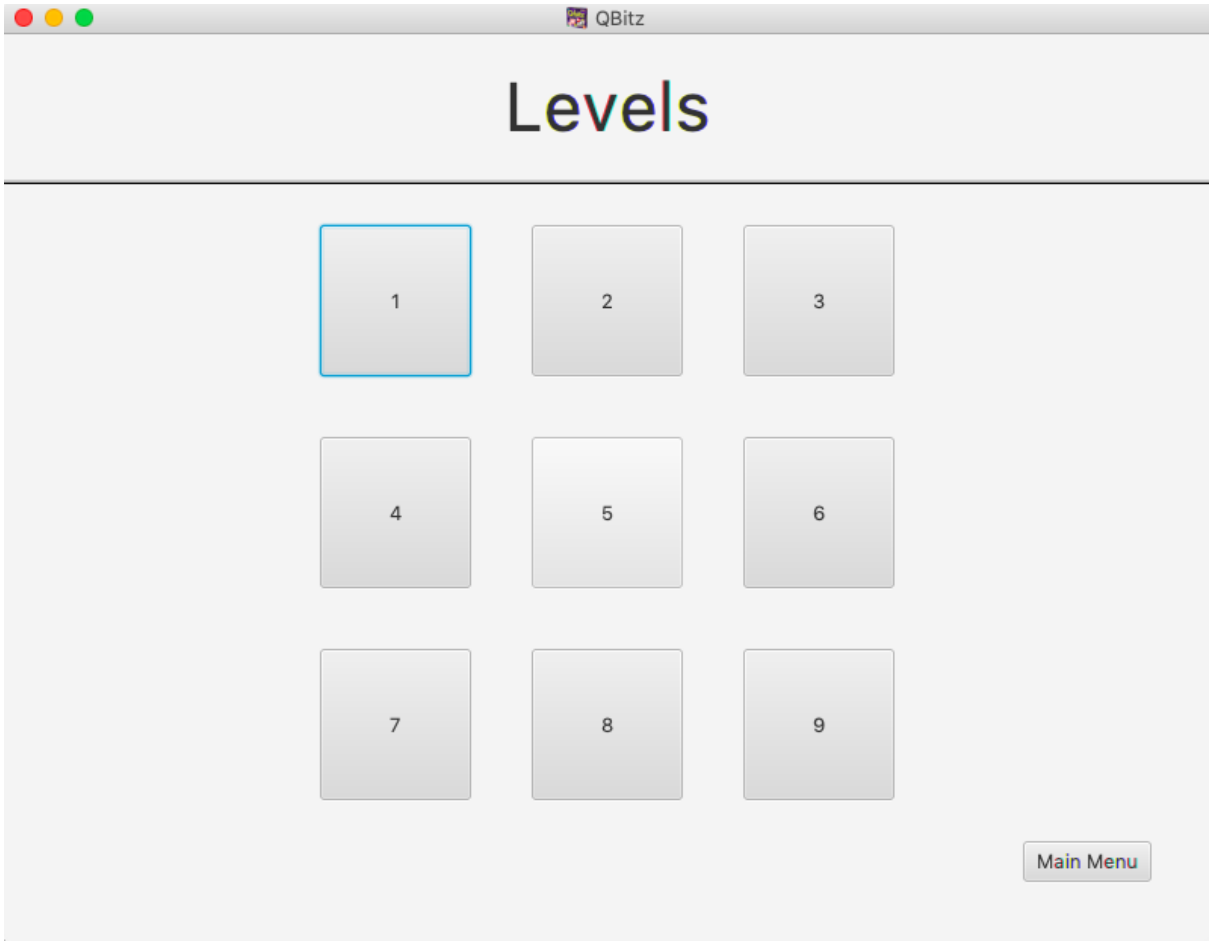
Our program does not require any installation, just executing the StarsLeague.jar file is enough to play the game.

4.2 How to Play

This is the main screen of the game. There exists Play, Level Select, How to Play, Settings,

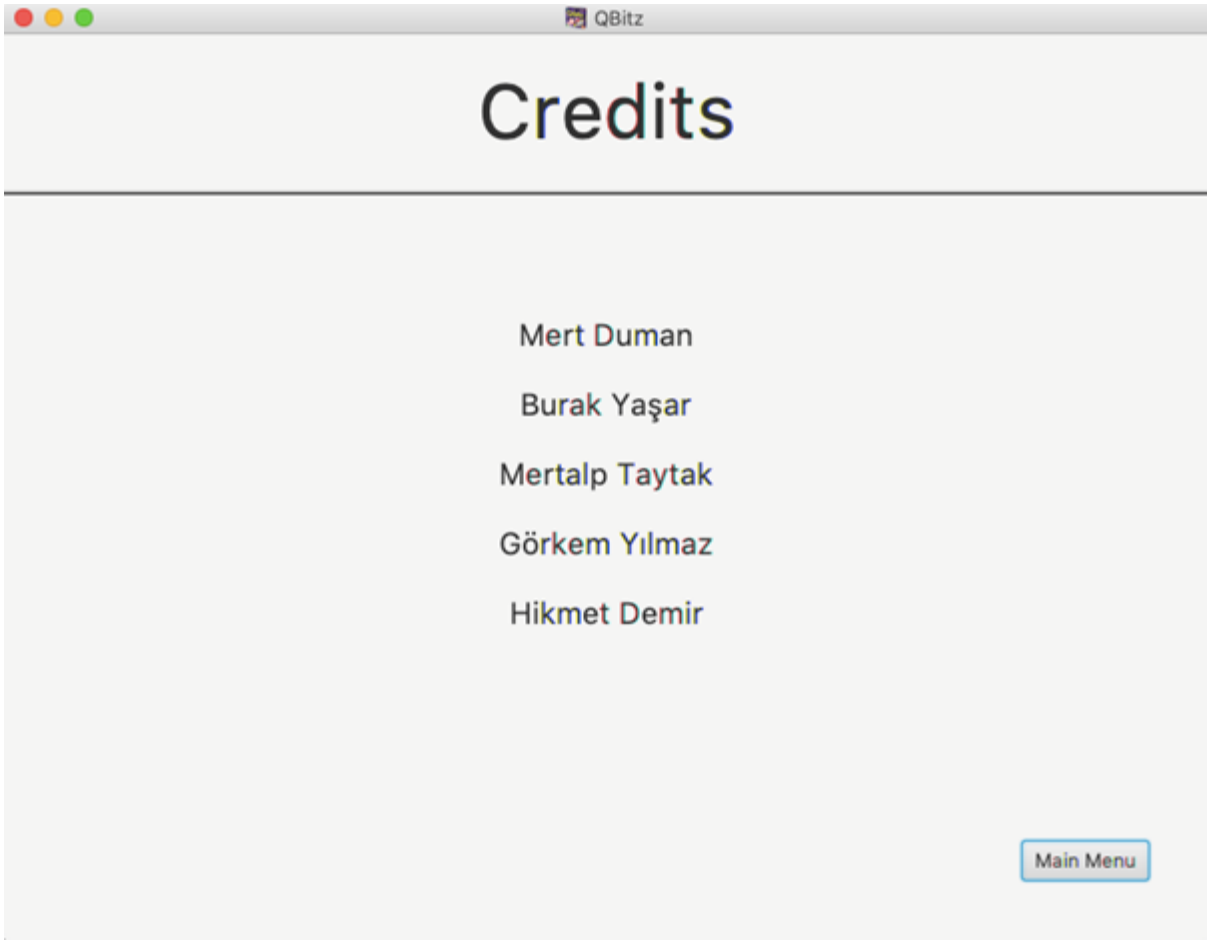


Credits and Exit buttons. Exit button closes the application. Player clicks on the Play button to start the game.

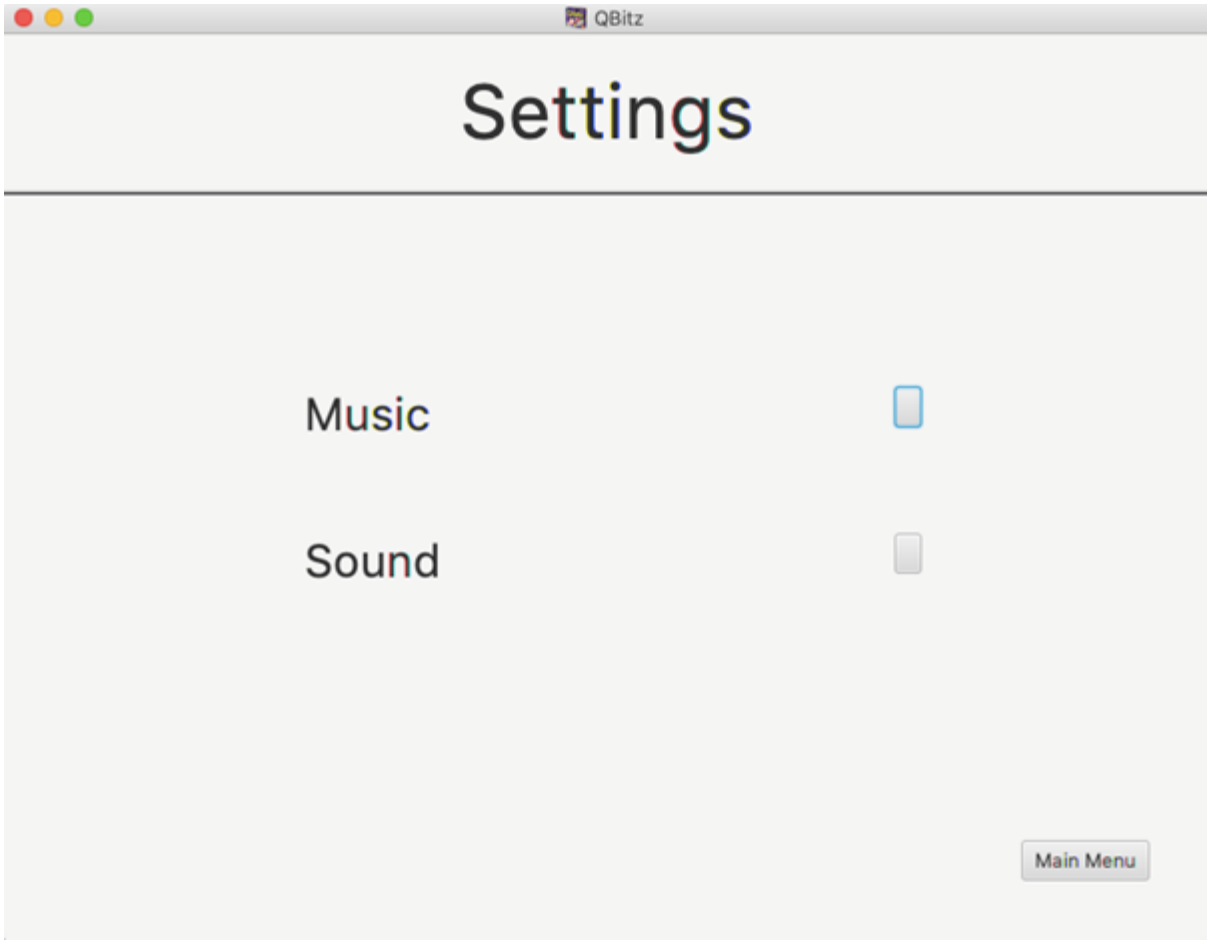


Level Select button opens a screen where the player can select which level he/she wants to play. Levels represent different difficulties.

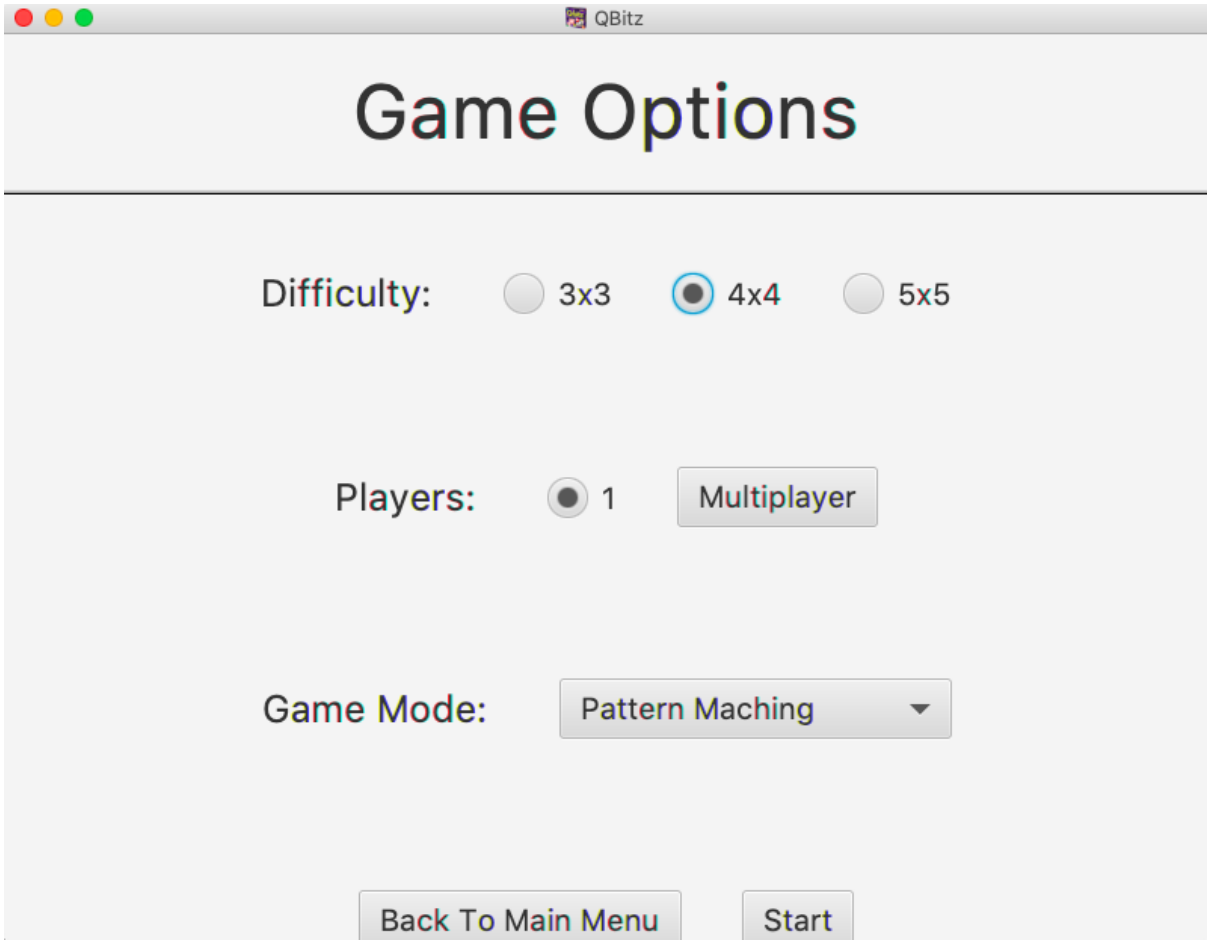
How to Play button shows a text which explains how can a player play the game.



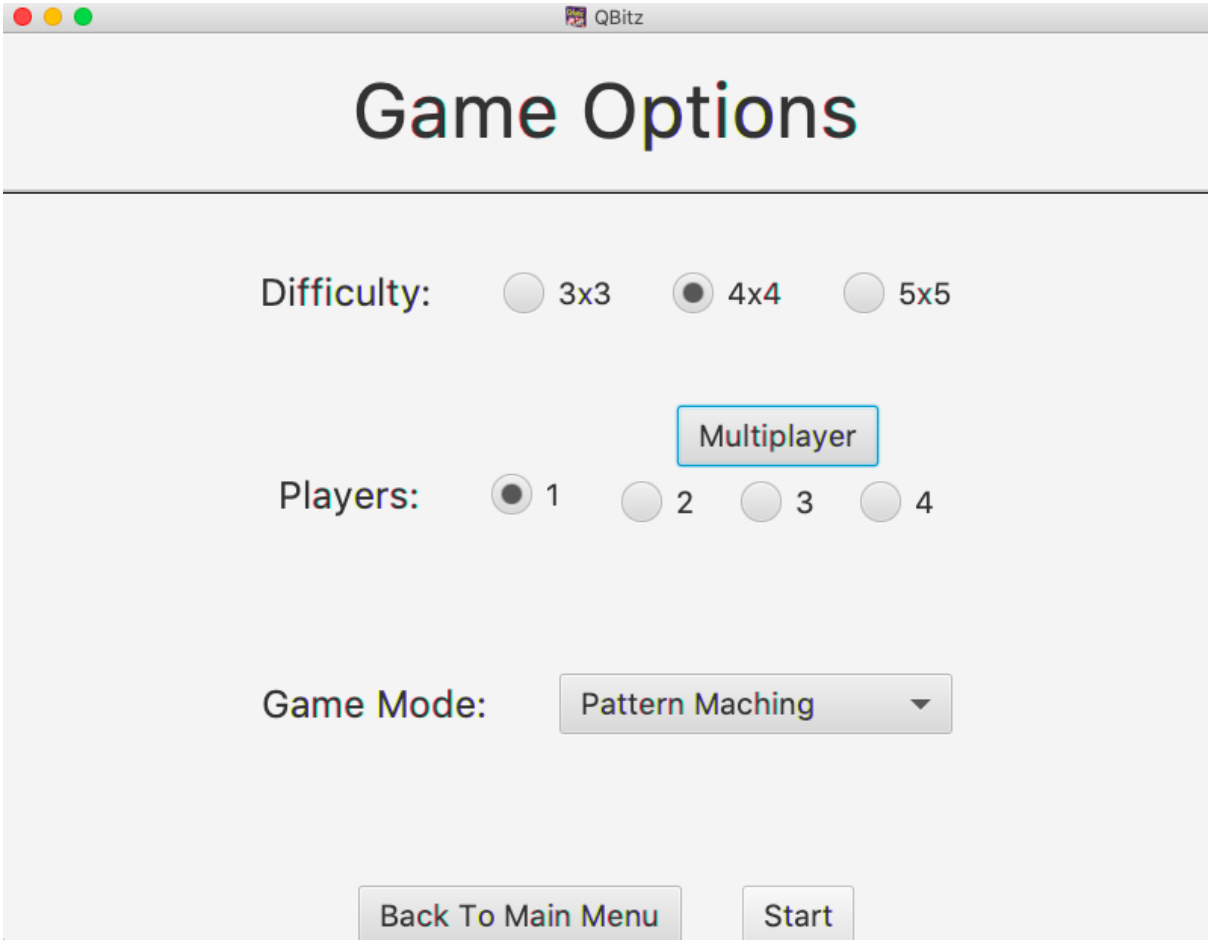
Credits scene shows the name of the each developer.



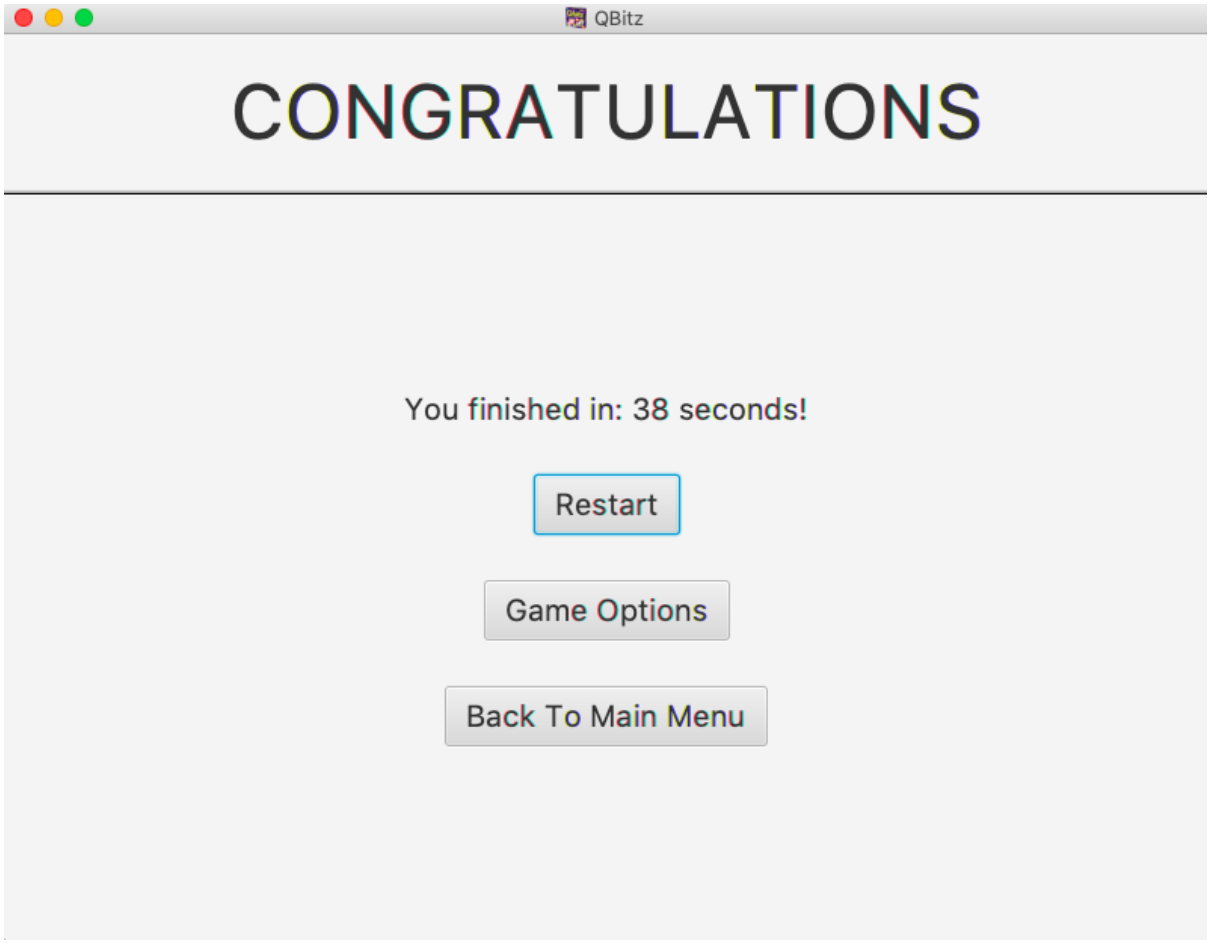
Setting has two options which are music and sound. Sound option activates the sound effects of the game and music option activates the background music of the game.



Game options has difficulty options, number of player selections(singleplayer or multiplayer) and game mode selections.



Once the multiplayer is selected, there occurs there buttons which contain the number of players that can be chosen for the multiplayer games.



Once the game is finished, a window shows the finish time. Also player can choose restart to play the same game as previos, he/she can change the game options by clicking the game options button and he/she can go back to the main menu.

5. References

https://docs.oracle.com/javase/8/javafx/api/

https://github.com/FXyz/FXyz

https://github.com/JFXtras/jfxtras

Bruegge, B., & Dutoit, A. H. (2014). Object-oriented software engineering: using UML, patterns, and Java. Harlow, Essex: Pearson.