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

Teaching Assistant: Gülden Olgun

Table Of Contents

1. Introduction

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

In the implementation, game opens with its main menu. Until the demo we were 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, until the demo we were only implementing single player game modes. But now, there is also multiplayer part of the game works and a special multiplayer game mode is implemented which is Two vs Two game mode. Implementing the multiplayer part of the game was the biggest challenge for us because it requires at least a basic level of knowledge on network systems, socket programming and client-host approach. Also making our Q-Bitz game an executable jar file was also challenging. In the process of doing it, we learned a lot about filing and computer directories.

After the implementation of the multiplayer parts are 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 different game modes were implemented such as From Memory, Maximum Patterns, Against Time, Different Cubes, Painting Puzzle and Two vs Two modes. While the GameUIController class controls the game, panes for the cube faces and sample pattern had been inserted into the game board.

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 Parts

-Client: This class is an example of socket programming, it creates a host and clients in order to allow playing the multiplayer game modes.

-ClientHandler: This class extends the Thread class and handles the game status by using the enumerated game states. Basically the purpose of this class is to handle situations that happens through a server.

-MouseControl: Since playing a game with only a keyboard is not very efficient, implementation of a mouse control class was necessary. This class allows player to play the game with a mouse. It controls the cube rotations and cube face insertions to the board.

-Server: This class is implemented to make multiplayer game modes work. This class handles the multiplayer game options and players (which are a host and client(s)).

-SmartBox: This class allows players to rotate and use a 3D cube in the game. When the cube is rotated, front side of the cube is automatically selected with the usage of this class. So player do not have to drag from cube to board, he/she can simply double click on the board to insert a cube face.

2.2 Removed Parts

Some css parts and unused game modes removed from the project. Also, we decided not use JavaFxtras library rather we control the cube with the mouse. Board class in model subsystem was in our class but we realize that it is not necessary therefore, it was removed.

3. Work Allocation

Mert Duman:

- All reports (analysis design final for both implementation 1 and 2)
- Multiplayer mode implementation
- Multiplayer related classes such as GameClient, ClientHandler, HostServer
- GUI of the game (FXMLs)
- Controller classes implementation
- Implementation of the levels
- 2vs2 Mode implementation
- From Memory Mode implementation
- Pattern Matching mode implementation

Mert Alp Taytak:

- All reports (analysis design final for both implementation 1 and 2)
- Game Model classes such as Game,Pattern, PatternPack, Cube
- Painting mode implementation
- GUI of the game (FXMLs)
- Controller classes implementation
- Pattern Matching mode implementation
- Building Jar file

Görkem Yılmaz:

- All reports (analysis design final for both implementation 1 and 2)
- Game Pictures creating patterns
- GUI of the game (FXMLs)
- Controller classes implementation
- From Memory Mode implementation
- PatternMatching mode implementation
- Demo1 slides

Hikmet Demir:

- All reports (analysis design final for both implementation 1 and 2)
- Against Time mode implementation
- General Structure
- Game Model classes such as
- GUI of the game (FXMLs)
- Painting mode implementation
- Demo1 Youtube video
- Controller classes implementation

Burak Yaşar:

- All reports (analysis design final for both implementation 1 and 2)
- 3D Cube Implementation, SmartBox,Cube classes
- Against Time mode implementation
- Game Pictures creating patterns
- Background music of the game
- GUI of the game (FXMLs)
- Controller classes implementation

4. 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.

5. User's Guide

5.1 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 QBitz.jar file is enough to play the game.

5.2 Build Instructions

First of all your system must provide

You should install IntelliJ Idea from the website https://www.jetbrains.com/idea/download, then you have to download our project Qbitz from our github website (which involves the source code) https://github.com/byburakyasar/cs319fellowship. Open IntelliJ and you can see a bar on the top of IntelliJ from this bar select File>>Open then select to the path of file which involves our source code that you download from Github. Click Ok, then IntelliJ loads our project. After the project is opened in the IntelliJ IDE you should build the project like the following. You can click File>>Project Structure>>Libraries then click the + icon, select Java

then navigate to cs319fellowship/lib select fxyz3d-0.4.0.jar. You can select Build>>Build Project, After that you can go through the folder src and under this folder you can see Main.java move your cursor on the Main.java file. Simply press right click on Main.java then select Run ‘Main.main()’ from options, it will run the code and open our game and or instead of this you can simply double click the jar file of our game.

5.3 How to Play

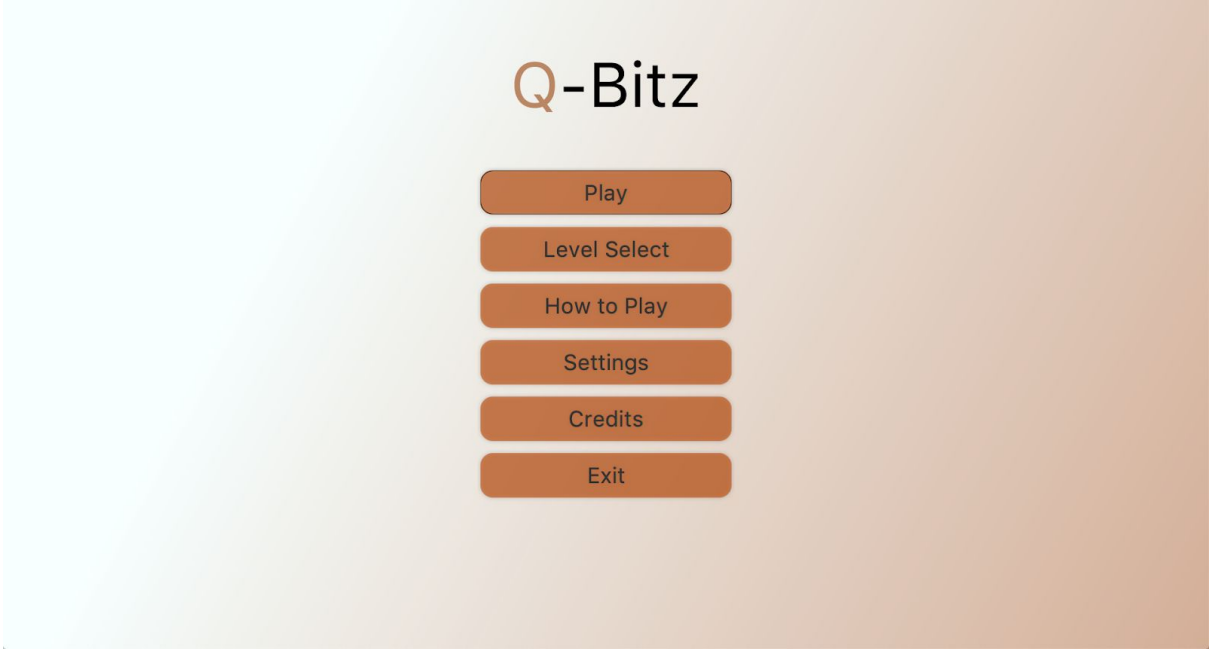

Figure 1: Main Menu

Main Menu contains Play, Level Select, How to Play, Settings, Credits and Exit buttons.

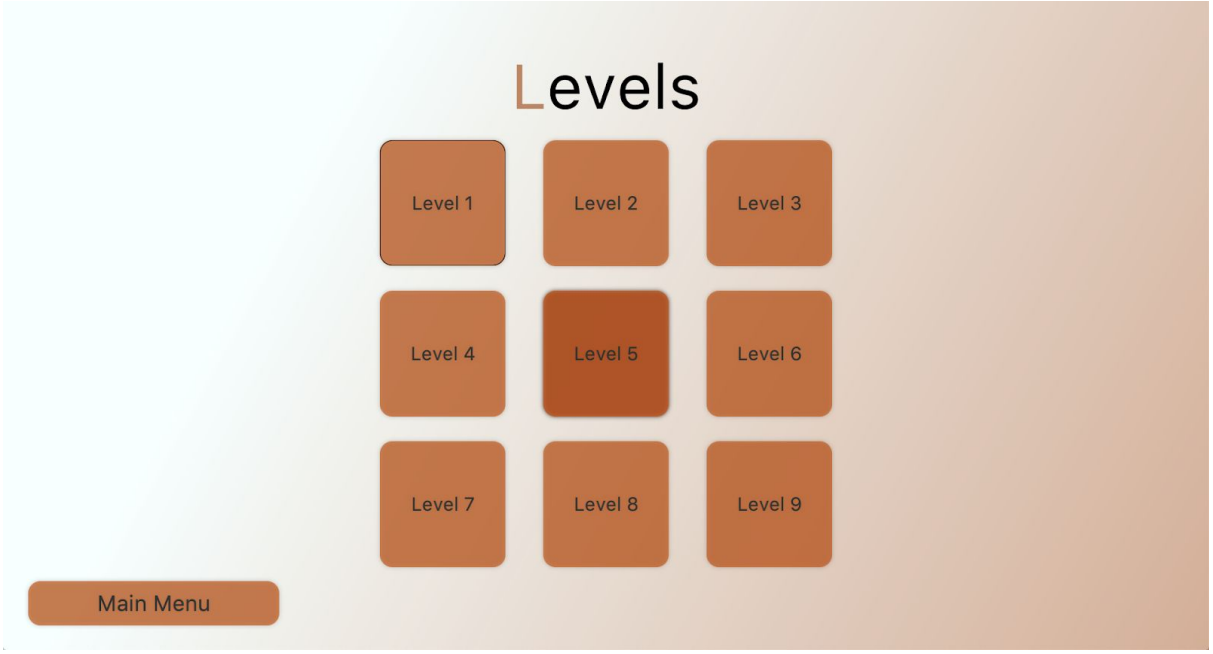

Figure 2: Levels

Player can select different levels and as the level number increases, difficulty of the game also increases.

Figure 3: How to Play General Info

Figure 4: How to Play Pattern Matching

Figure 5: How to Play Multiplayer

Figure 6: How to Play From Memory

Figure 7: How to Play Against Time

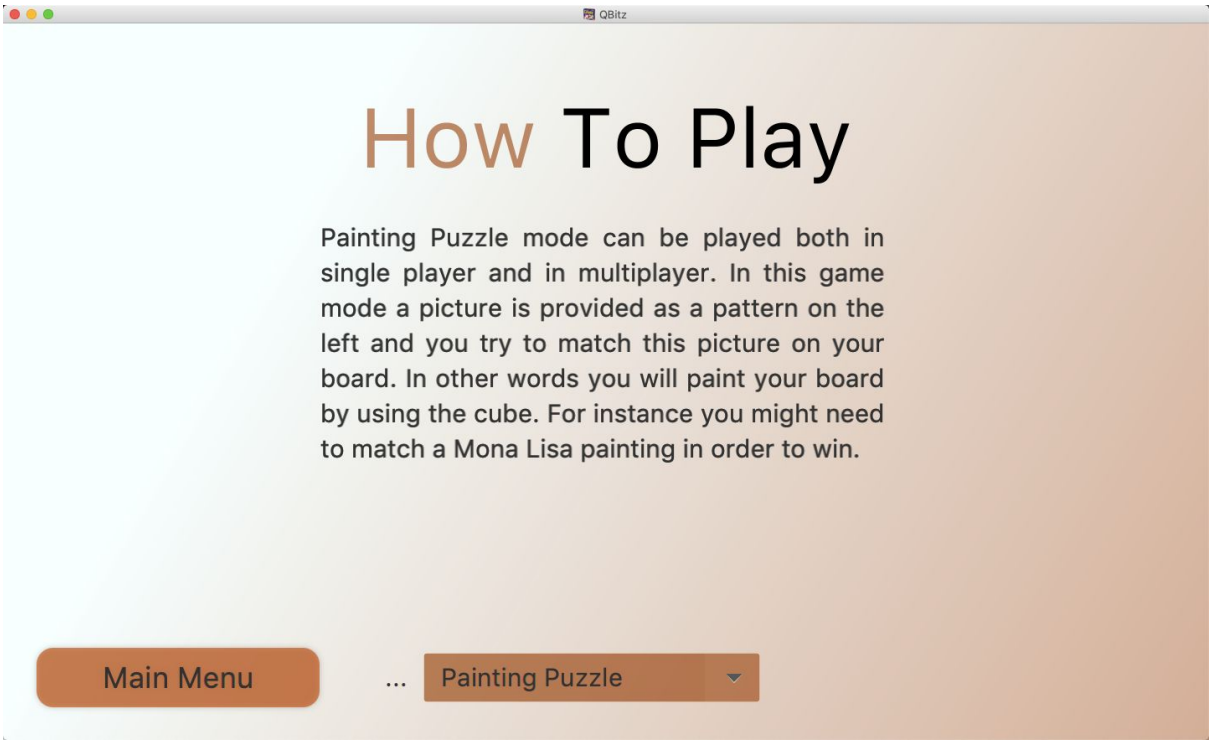

Figure 8: How to Play Painting Puzzle

Figure 9: How to Play Two vs Two

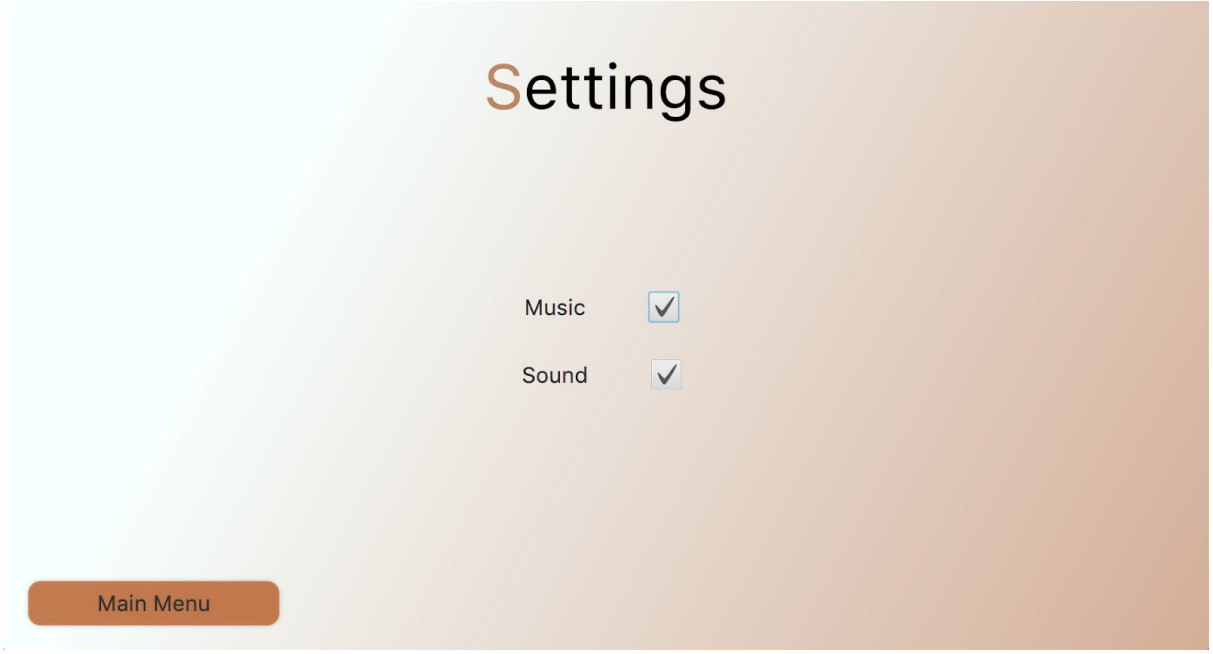

Figure 10: Settings

Player can choose if the music or/and sound effects will be on during the game from the settings page.

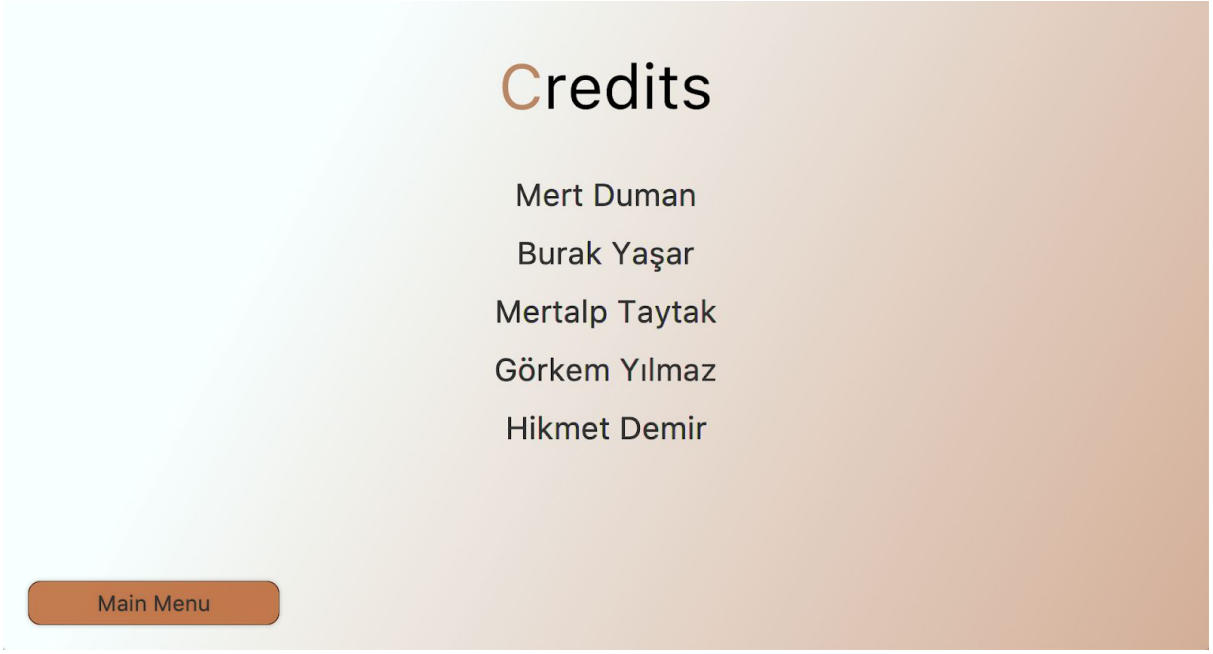

Figure 11: Credits

Credits page shows the developers of the game.

Figure 12: Game Options

Game options page is important to create a game and start. Player can select the difficulty and each level of difficulty represent the grid size of the pattern. Number of player can also be chosen for the difference of the single player and multiplayer games. Player can select to play with a 3D cube or a 2D cube and lastly, player selects the game mode in order to start a game in that mode. Players can also select the pattern that you will play. While the start button starts the game, main menu button sends player to the main menu.

Figure 13: Pick username

Each player selects a username in order to start the game.

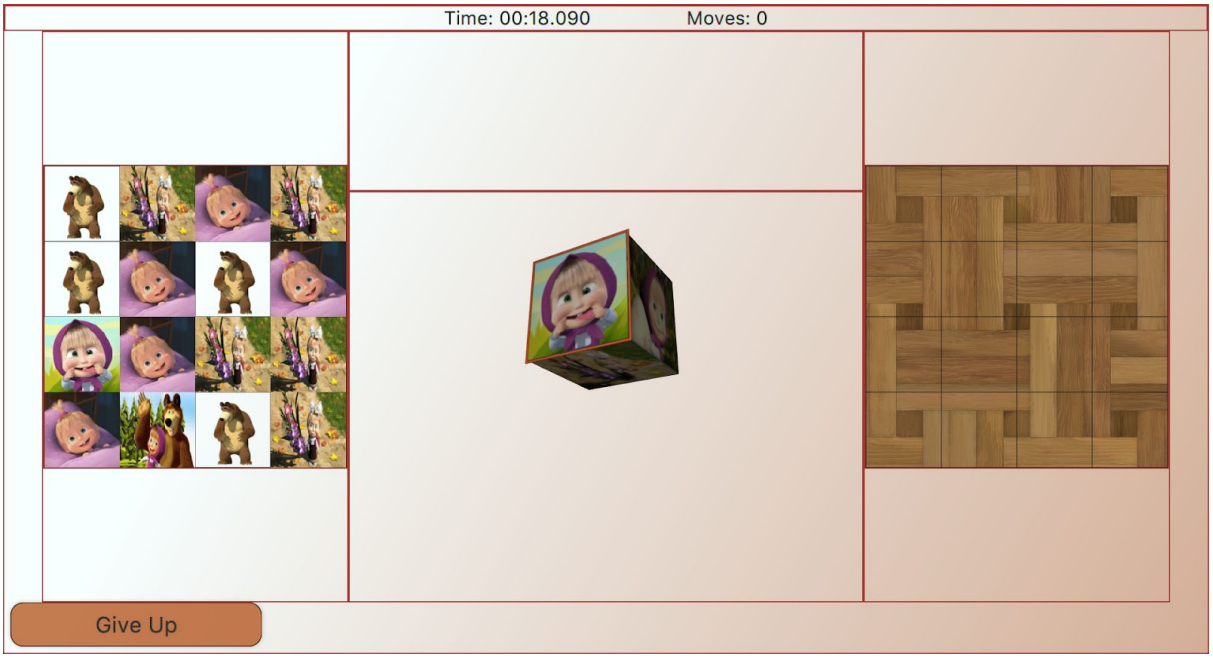

Figure 14: Pattern Matching

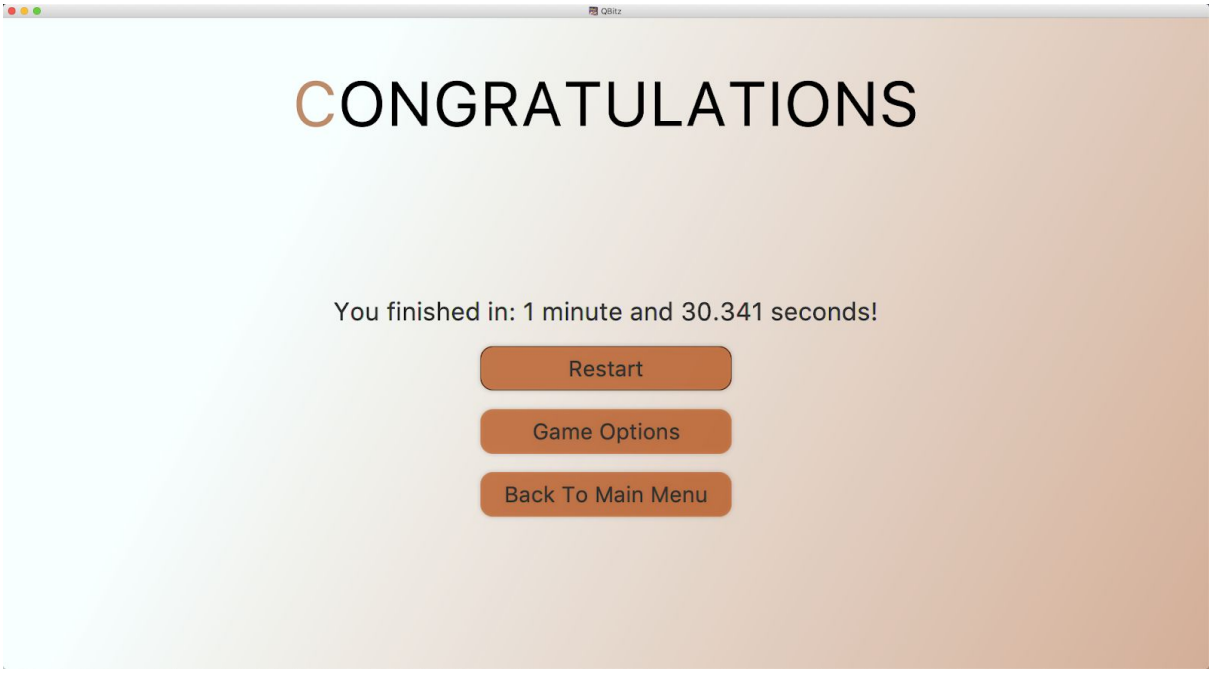

Figure 15: Congratulations

When the game mode is finished with success, congratulations page opens. It shows much time it did take to finish the game and there are three buttons which are Restart, Game Options and Back to Main Menu. Restart button start a new game with the same game options as before. Player selects the Game Options button in order to create a new game with different difficulty or different game mode. Back to Main Menu button sends player back to the main menu.

Figure 16: Give Up

When the player wants to give up, he/she presses on the Give Up button and when that happens, You Gave Up page appears. In this page, the same three buttons in the Congratulations page also appears.

Figure 17: Multiplayer Player1 Aspect

Figure 18: Multiplayer Player2 Aspect

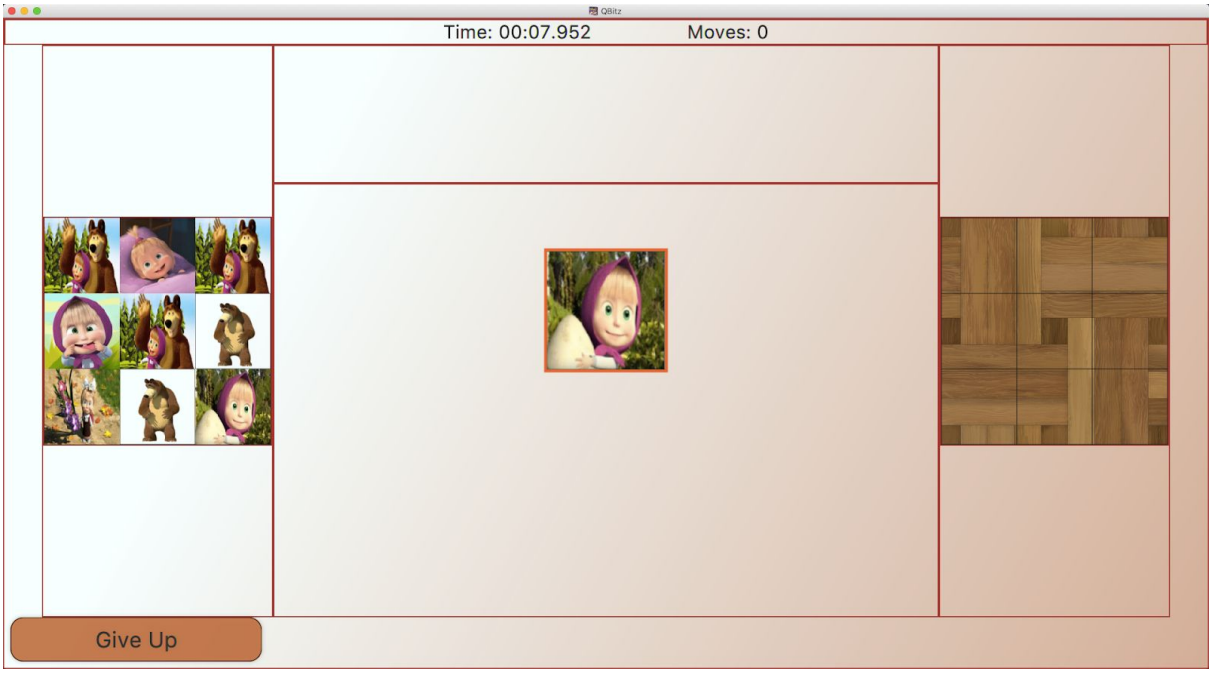

Figure 19: Against Time Mode

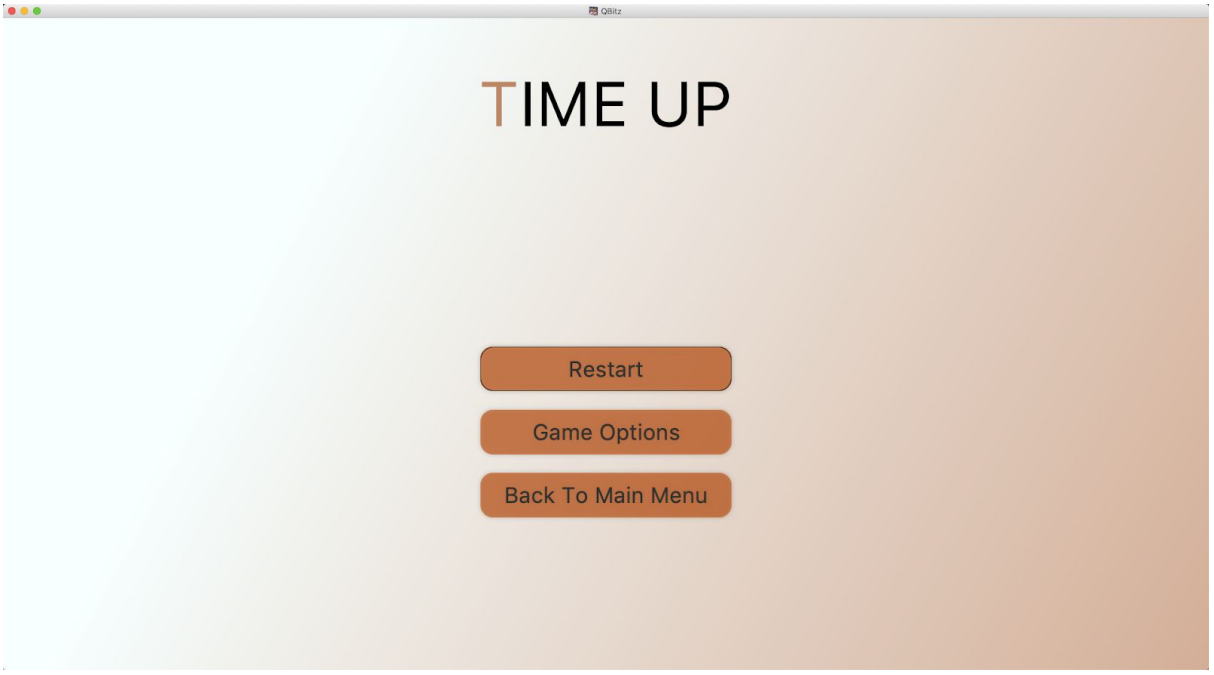

Figure 20: Time Up

In the Against Time mode, when the player cannot finish the pattern in the given time limit, he/she loses the game and Time Up page opens. In this page, the same three buttons in the Congratulations page also appears.

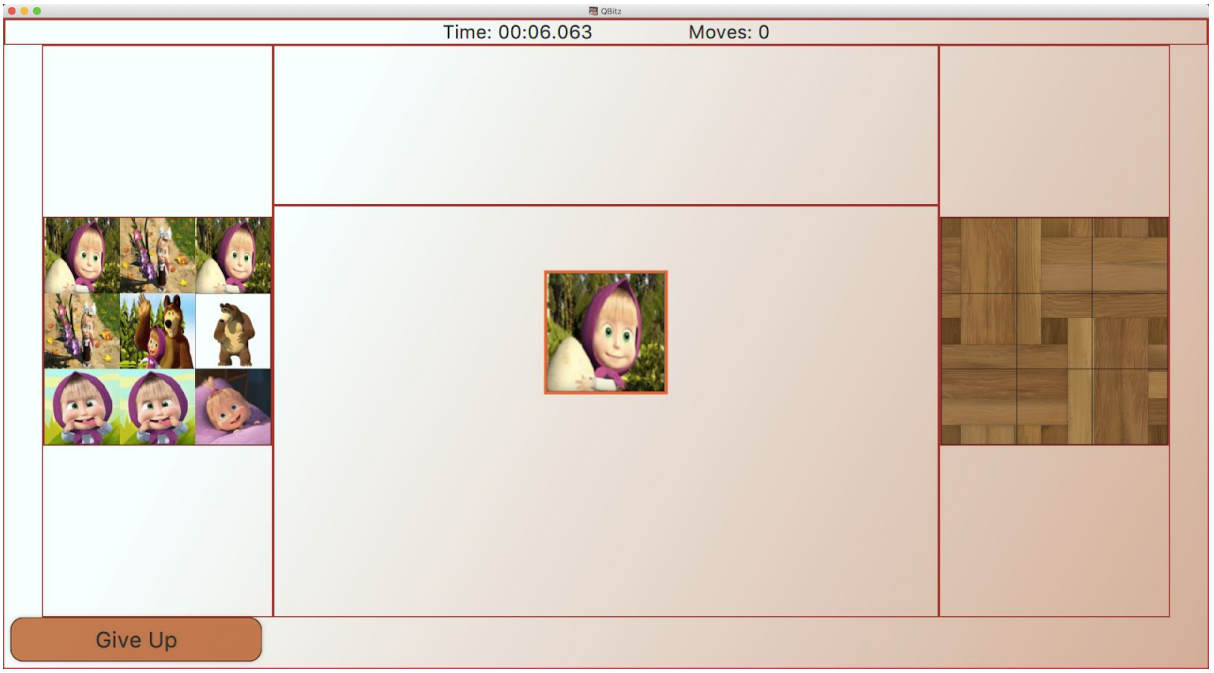

Figure 21: From Memory Mode Pattern Not Disappear Yet

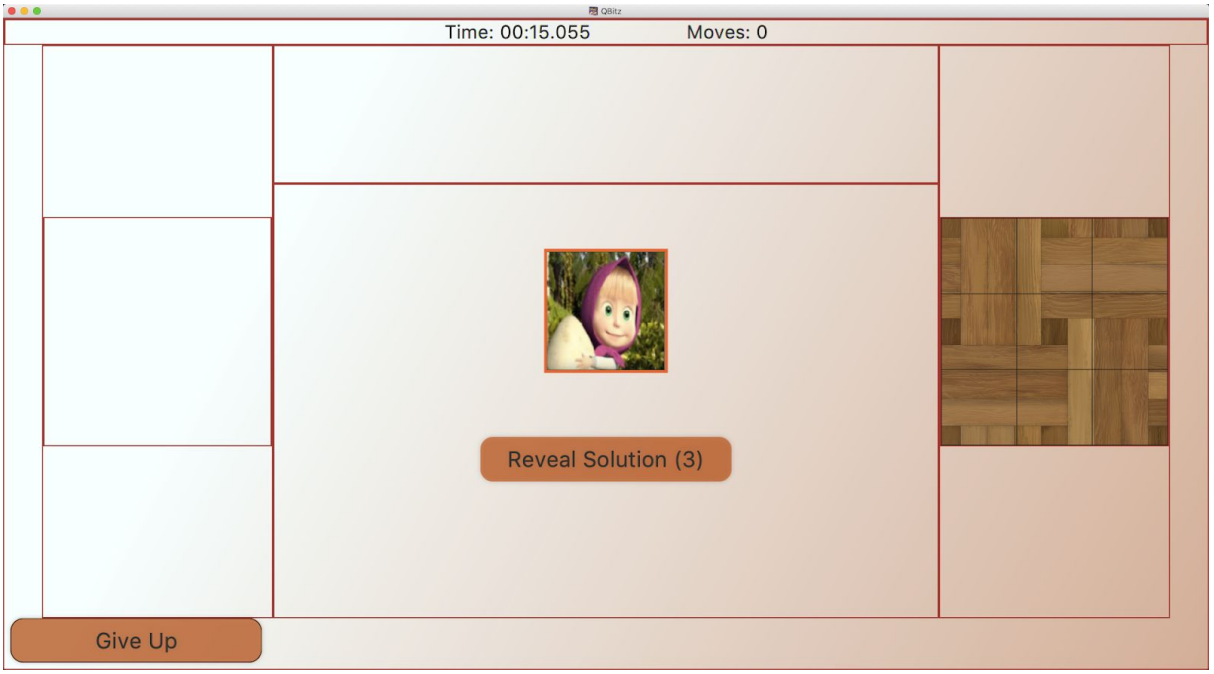

Figure 22: From Memory Pattern Disappear

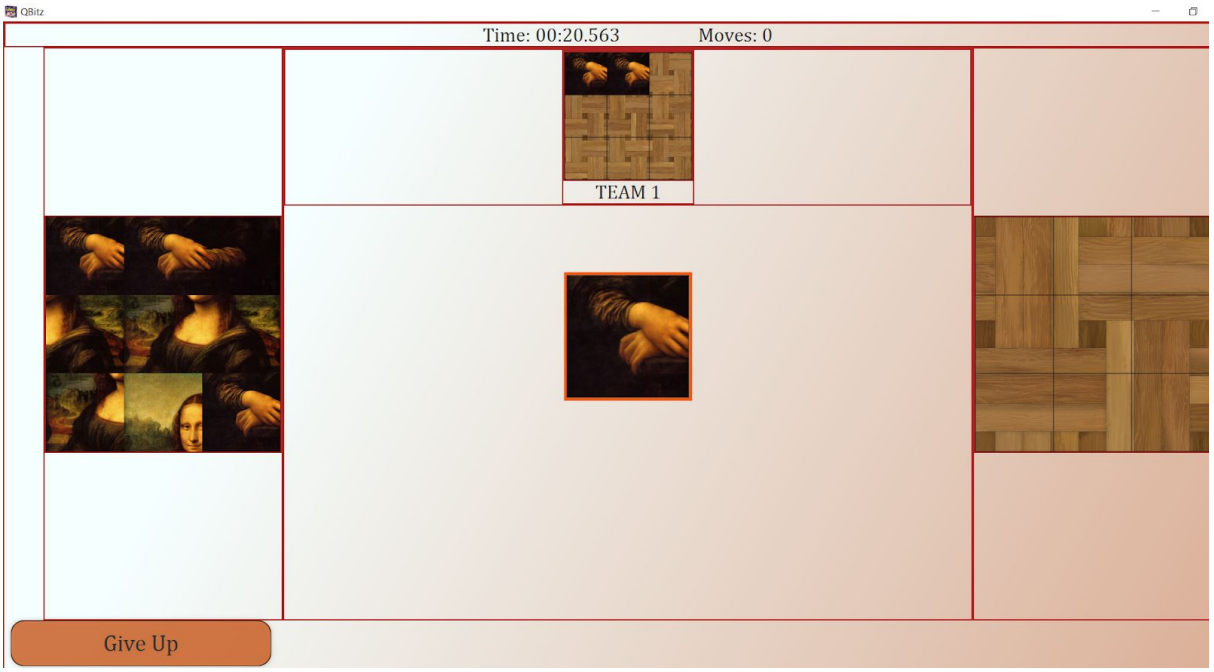

Figure 23: Two VS Two Mode Aspect From One Team

Figure 24: Two VS Two Mode Aspect From Both Teams

6. References

[1] “Java FX documentation, https://docs.oracle.com/javase/8/javafx/api/

[2] “Java FXyz documentation, https://github.com/FXyz/FXyz