Bilkent University

Department of Computer Engineering

CS319-OBJECT ORIENTED SOFTWARE ENGINEERING

Q-Bitz: Q-Bitz

Iteration 2 - Final Report

Sait Aktürk, Zafer Tan Çankırı, Berkin İnan, Halil Şahiner, Abdullah Talayhan

Supervisor: Eray Tüzün

1. Introduction **3**

2. Design Changes **3**

3. Lessons Learnt **4**

3.1 Technical 4

3.2 Teamwork 4

4. User's Guide **5**

4.1 System Requirements and Dependencies 5

4.2 How to Execute 5

4.2.1 Build from Source 5

4.2.2 Execute using JAR file 5

5. Game Screens **6**

5.3.1 Main Menu 6

5.3.2 Room List and Create Room Screens 6

5.3.3 Level Selection Menu 7

5.3.4 Race Mode Game Screen **8**

5.3.5 Image Recreation Mode Game Screen **9**

5.3.6 Memory Mode Game Screen **10**

5.3.7 Game Screen After Solution **11**

6. Work Allocation **11**

7. References **12**

1. Introduction

We have started the development process after the design report. Since the game has both singleplayer and multiplayer modes, we wanted to work o both of the modes at the same time. All functionalities for the Multiplayer Mode is implemented, we also have a remote server so that we do not need a local machine acting as a server for the multiplayer game.For single player mode, we have implemented all game modes consisting of race mode, image recreation and memory, they can be played on several board sizes such as 3x3, 4x4 and 5x5.

The 3D mechanics are implemented using JavaFX[1] without using third party 3D engines. Rest of the user interface components are also implemented using JavaFX and JavaFXML.

2. Design Changes

Although there are not much changes in the classes, there are significant changes in the game mechanics like the following:

- We changed the cube control mechanics from keyboard instead of mouse in order to make the the controls faster
- Instead of moving the cube from focused area to the board, we put a preview of the selected cube face on the board before placing it.
- The user is able to rotate the placed cube face on the board without the need of removing it, this functionality was necessary because there is no need to select a face again only for the sake of changing the rotation of it.

We had several changes for Client Server mechanics as well:

- There was a missing connection between the ServerSocket and the main server class, this connection was necessary because only the main server class can access the database using the database connector.
- We added a SceneController to the Client, the reason of this change is that, for each message that comes from the server, we need to notify different scenes in the client. This class controls the information for deciding which scene to send the message from the server. Since this is the only additional class, we haven't included the updated class diagram.

3. Lessons Learnt

3.1 Technical

- Since the 3D functionalities of JavaFX is fairly limited, the actual game mechanics took more time than it was expected. JavaFX does not include Quaternion rotations in 3D objects, it uses Euler angles resulting in a famous computational problem called Gimball Lock[2]. We have learned that it would be better to examine the functionalities of the system that we are using. We could have started the implementation of game mechanics earlier if we knew the capabilities of JavaFX better.

- We learned about splitting Client threads from javafx threads in order to resolve the error related to illegal threads.

3.2 Teamwork

- We learned the importance of using github, it enhanced the workflow of the project monumentally. Everyone in the team were able to use different development environments and still we were able to merge and pull the codebase efficiently.

- We learned the importance of splitting the work between team members, this was useful for implementing the project faster because different team members built different parts of the project. Still different parts were easy to integrate because of the project design phase.

- Other than splitting the work, teamwork was useful for solving difficult problems. For figuring out the rotational mechanics and pattern checking mechanics, multiple people worked on the problem and proposed different solutions.

4. User’s Guide

4.1 System Requirements and Dependencies

- Java 8 in order to support JavaFX 3D properties.
- “org.json” Library. The link for this library can be found in the instructions

4.2 How to Execute

4.2.1 Build from Source

- **Instructions for Building the Game:**
- The game requires one additional library which is "org.json".
  - * It can be gathered from our repository or from https://mvnrepository.com/artifact/org.json/json
  -
- The game should be pulled from our repository's master branch. https://github.com/InanBerkin/CS319-Project
-
- Then the JSON library should be imported to the project in the desired IDE.
-
- Then it should be compiled and runned.

4.2.2 Execute using JAR file

5. Game Screens

5.3.1 Main Menu

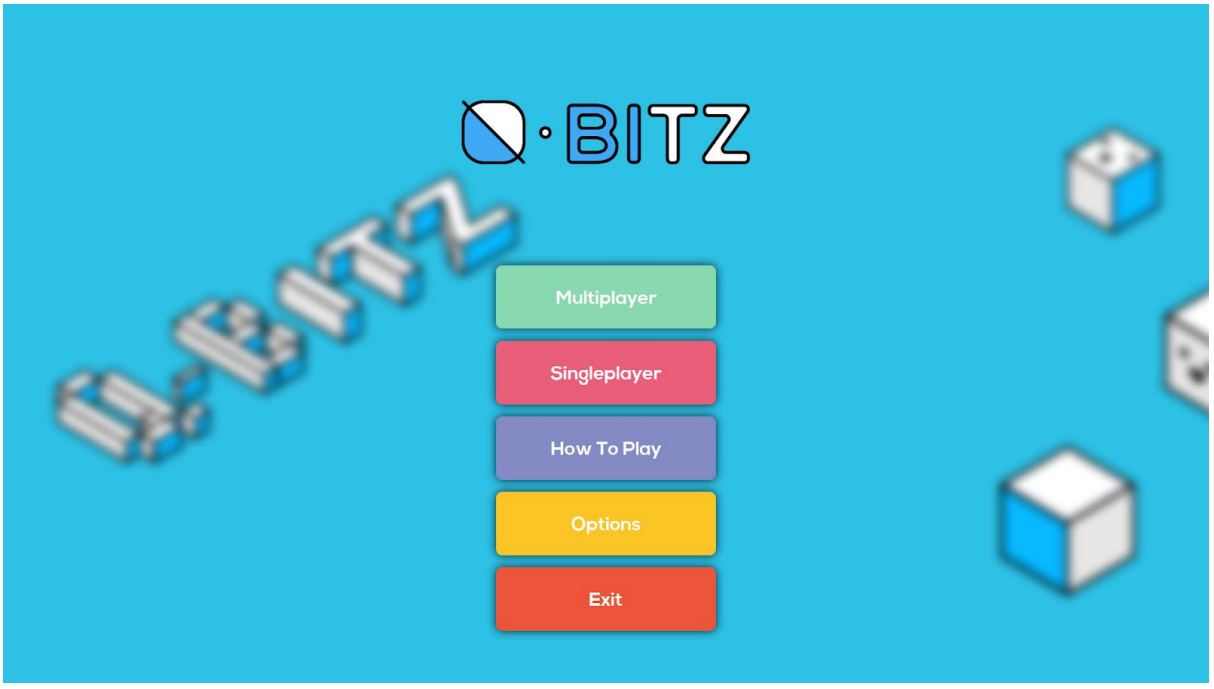

Figure 1. Main Menu

5.3.2 Room List and Create Room Screens

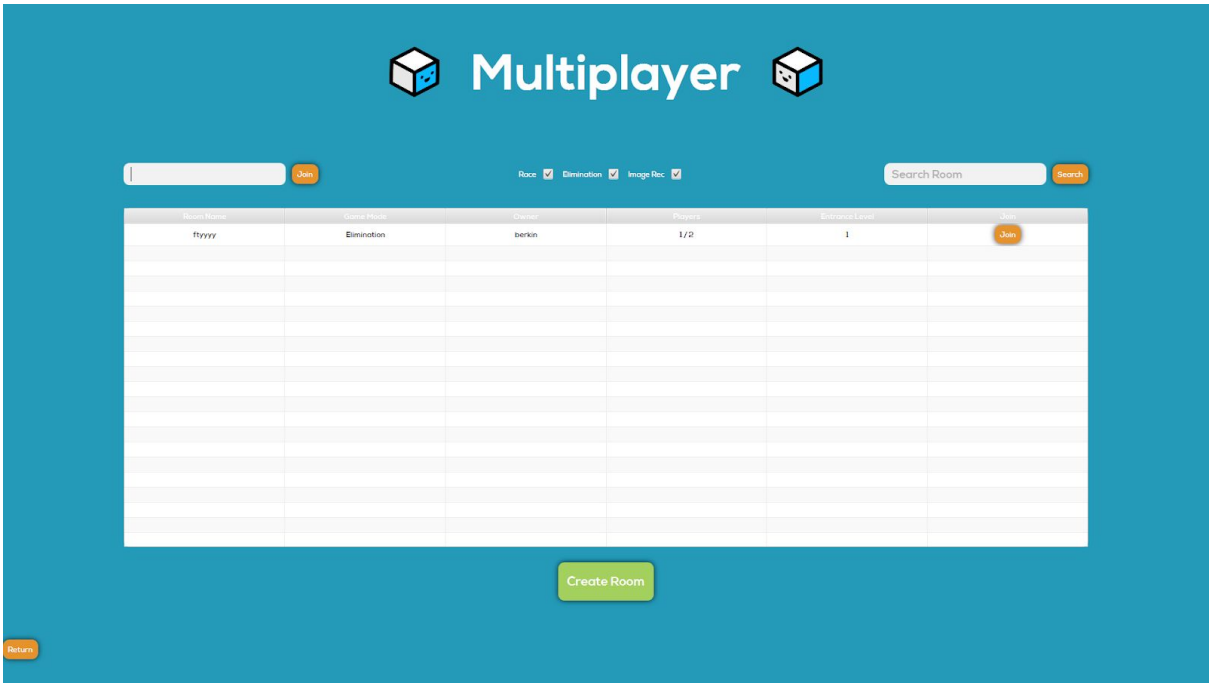

Figure 3: List of Rooms

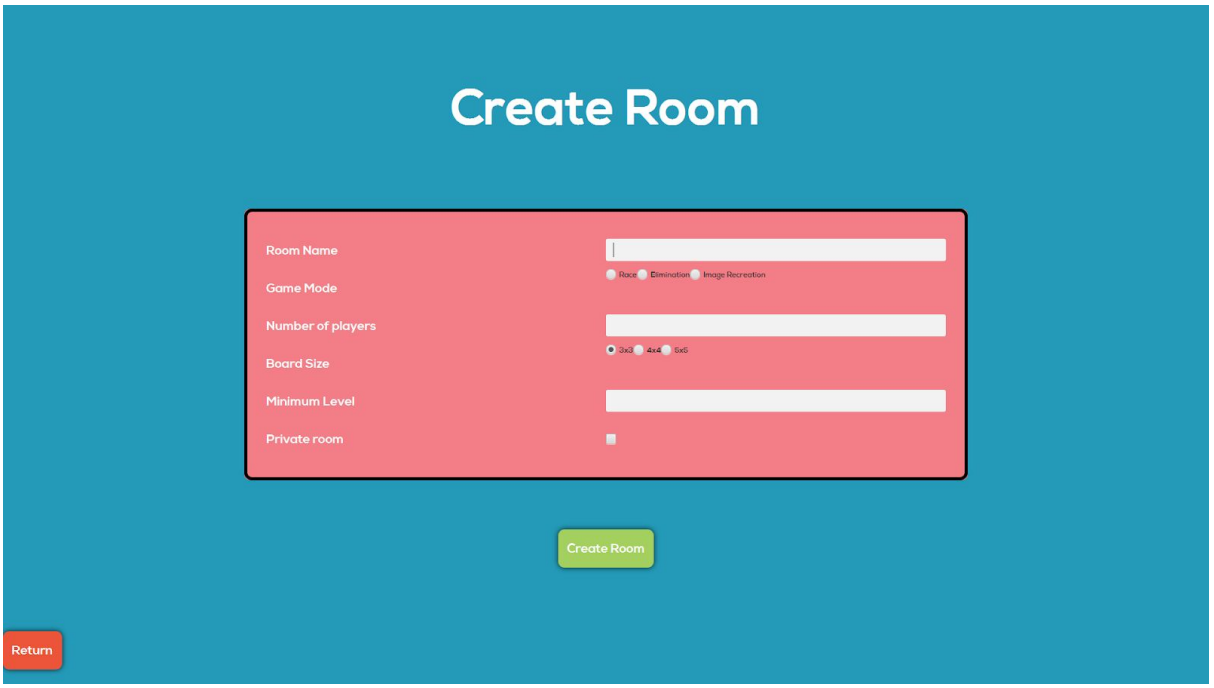

Figure 4: Login Menu

5.3.3 Level Selection Menu

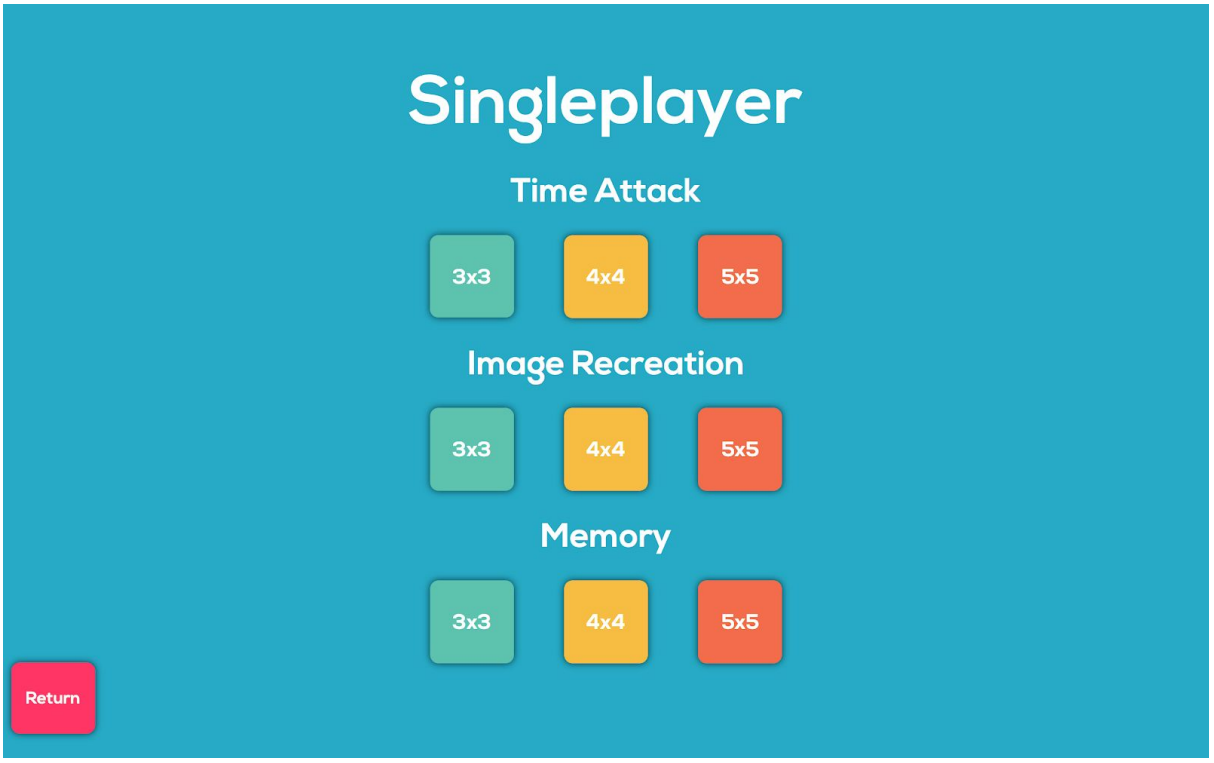

Figure 5: Level Selection Menu

5.3.4 Race Mode Game Screen

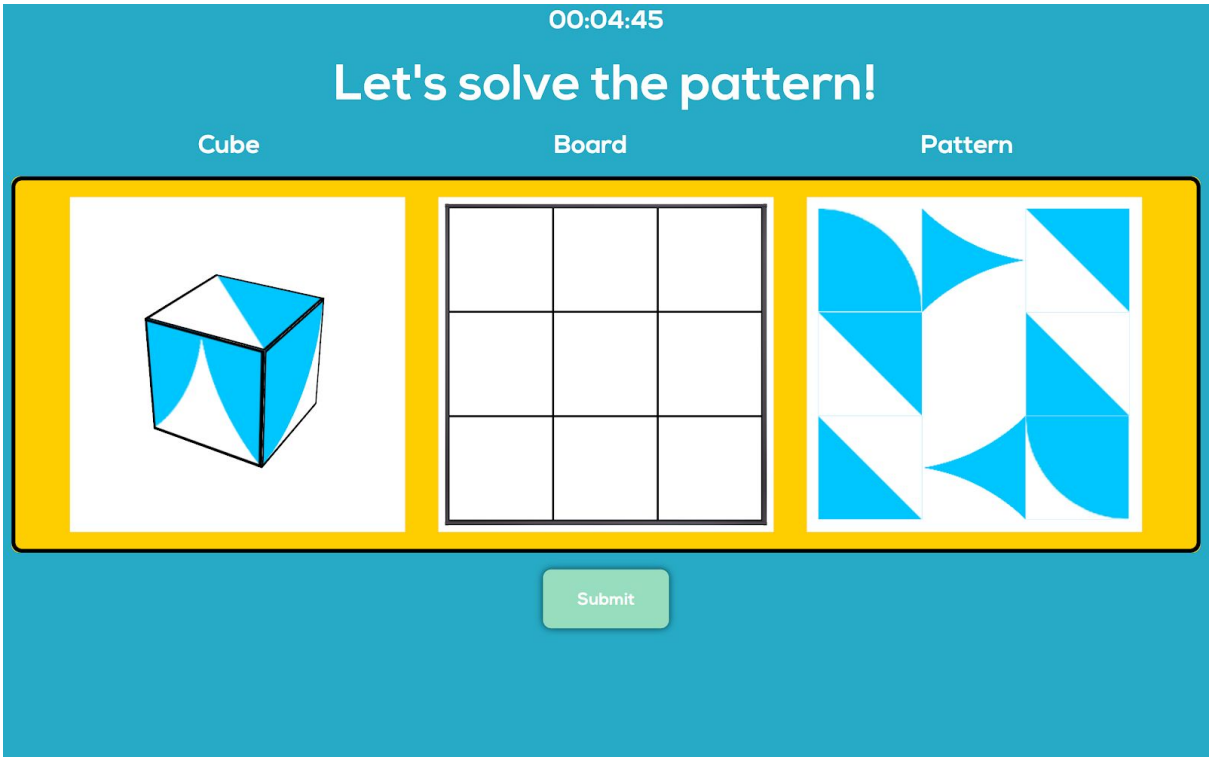

Figure 6: Game Screen for Race Mode

5.3.5 Image Recreation Mode Game Screen

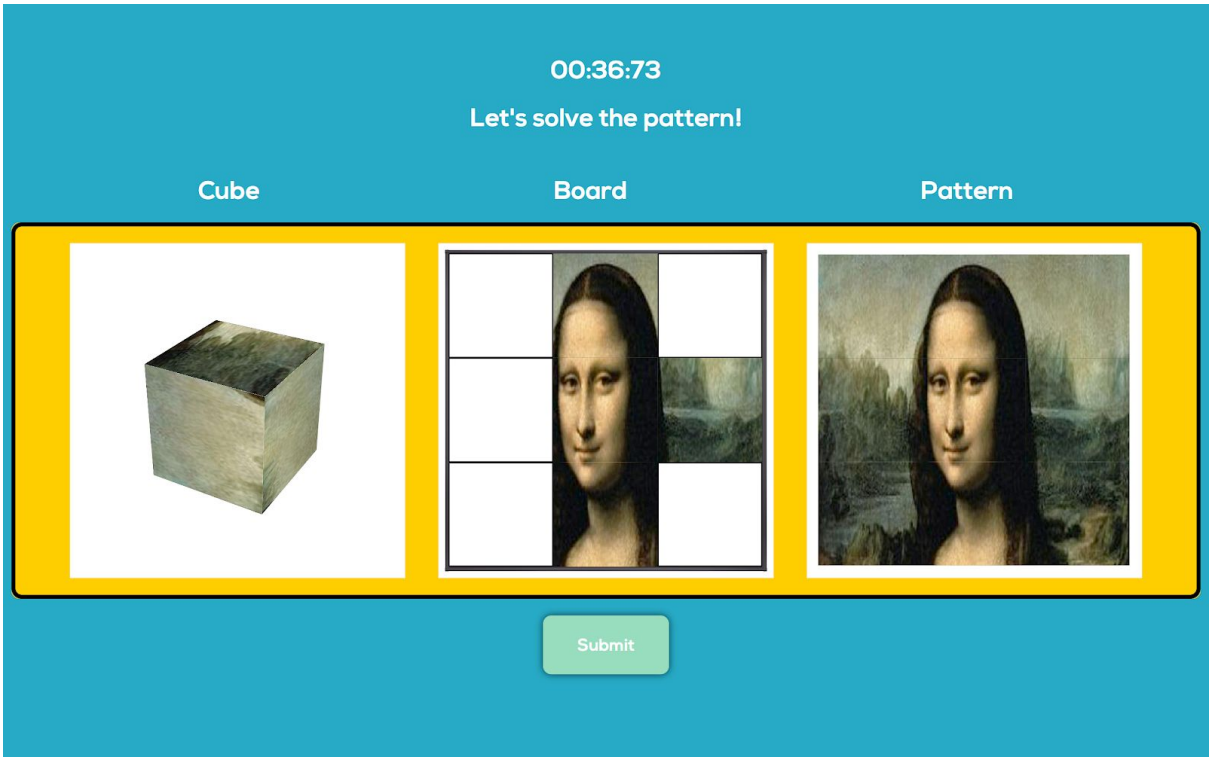

Figure 7: Game Screen for Image Recreation

5.3.6 Memory Mode Game Screen

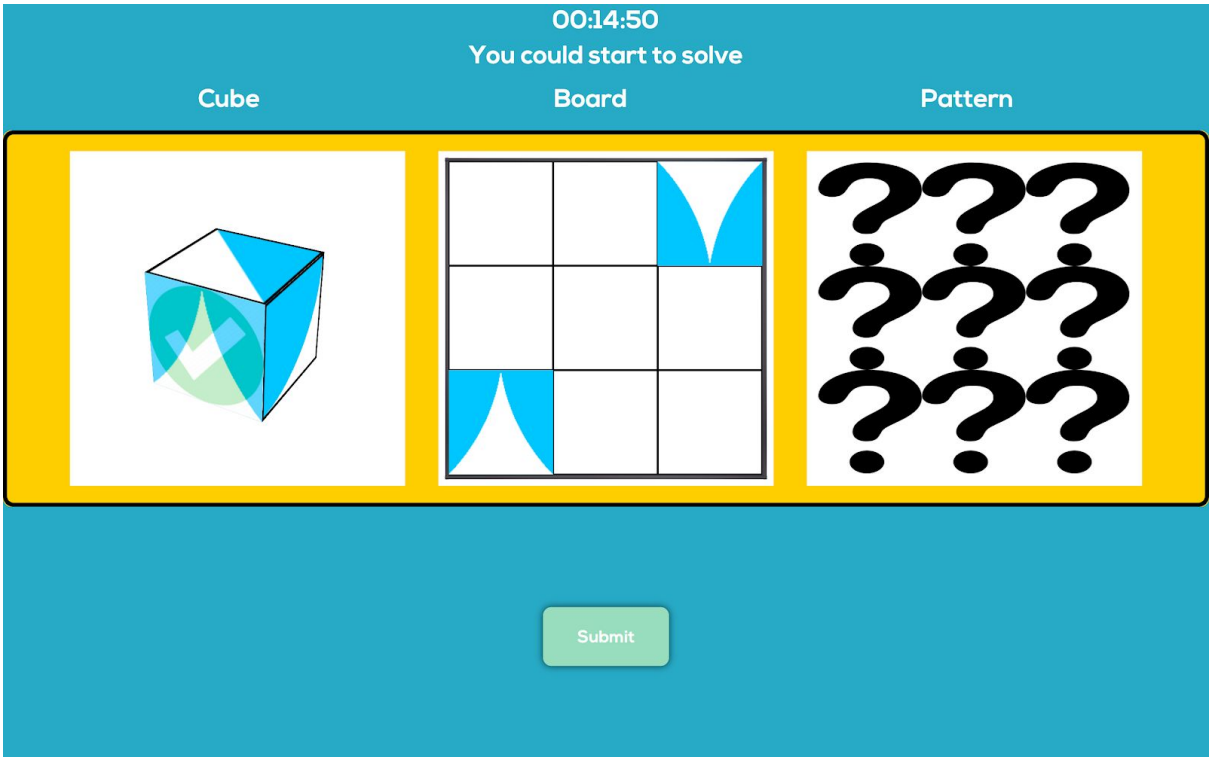

Figure 8: Game Screen for Memory Mode

5.3.7 Game Screen After Solution

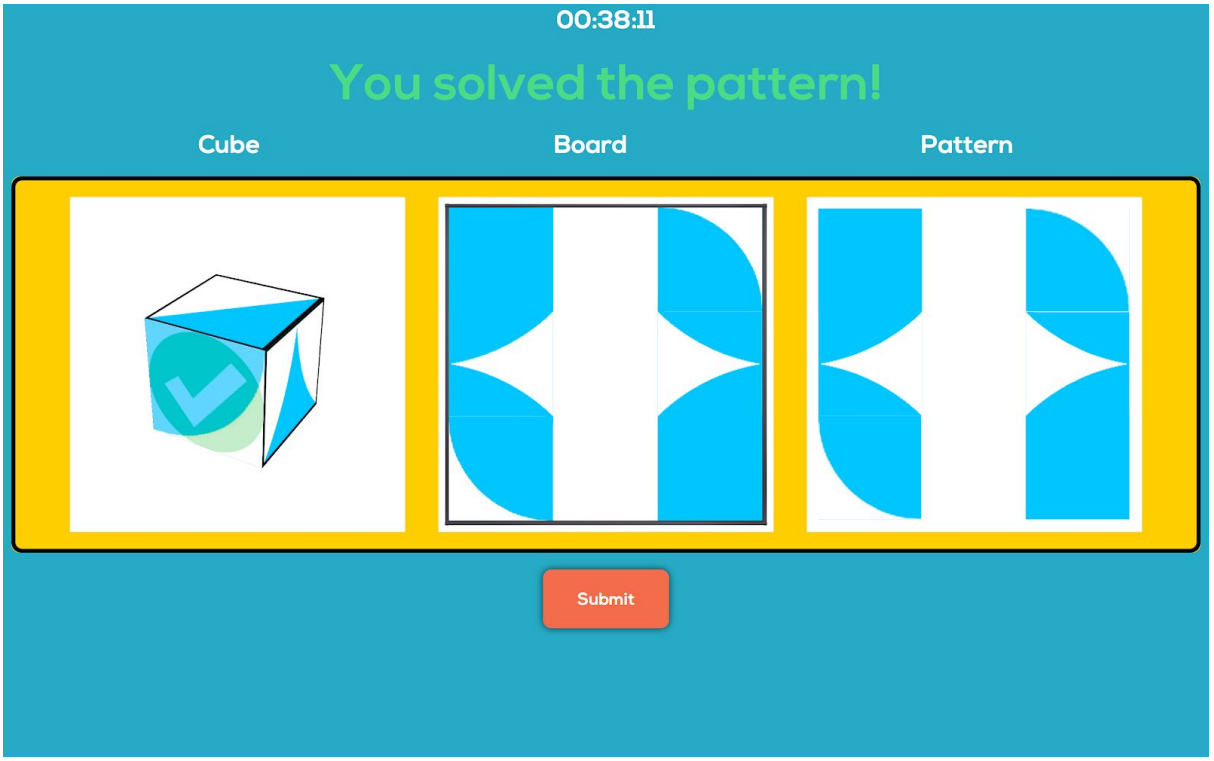

Figure 9: Game Screen after Solution

6. Work Allocation

6.1 Sait Aktürk
He worked on cube rotation and animation, also he implemented image recreation and memory mode, pattern generator, pattern checker, some part of the game screen of the game generally speaking image handling related work. He also worked on experimental mode that the user could play the game with hands through to artificial intelligence.

6.2 Zafer Tan Çankırı
His main task was handling and development of the server of the game. He worked on the database structure of the game, and also he developed the connection between the different clients over the server. He also worked on the cube rotation, 3D camera settings and animations of the game. He also worked on an experimental mode that the user could play the game with Arduino powered wireless gloves which are developed to control the game.

6.3 Berkin İnan
He worked on the designs, illustrations and implementations of the all the game screens. He connected the client side to the server. He also maintained the GitHub repository of the project for all the team members. He worked on implementing multiplayer modes. He handled the JavaFX problems of the project.

6.4 Halil Şahiner
His main task is the implementation and integrating the game logic to the application. He design and insert the pattern and board to the game instances of the game. He also worked on the game modes for the application which is mainly multiplayer and singleplayer game modes. He initially created the pattern generator for the game; however, Sait Akturk was the one who created the further implementation with his guidance.

6.5 Abdullah Talayhan
His main task is the implementation of Game settings, local file management and assisting the client-server communication. He also worked on several controller classes and design decisions about the game mechanics. He is maintaining the server and database located on his remote server located in France.

7. References

[1] https://www.oracle.com/technetwork/java/javafx/overview/index.html

[2] http://www.okanagan.ieee.ca/wp-content/uploads/2016/02/GimbalLockPresentationJan2016.pdf