Bilkent University

Department of Computer Engineering

CS 319 - Object-Oriented Software Project

IQ Puzzler

Final Report

Group Name: Violet

Group Members

Ege Akın

Safa Aşkın

Betim Doğan

Alp Ertürk

Elif Gülşah Kaşdoğan

Table of Contents

1. Implementation Process

After the process of first iteration, we started to the implementation process. For implementation of our project we used Façade design pattern to reduce coupling and MVC pattern for decomposition. We also used IntelliJ for implementation. Each of our group members connected to GitHub depository named IQ-Puzzler-Pro-Violet, which was created by Alp in the beginning of the course. The biggest convenience it brings us is that IntelliJ has an access to our classes in the project and it can be connected to GitHub directly. It allowed us to change and edit our classes in a synchronized way.

The first step of implementation process of IQ Puzzler Pro was distributing the tasks among group members. The distribution of tasks has been done with considering the three stages of software architecture. Despite the fact that we divide our tasks, we also took part in other tasks due to provide best efficiency in limited time. We distribute the tasks among each other with considering who is familiar and easily handle with which part and difficulty of the tasks. Since we use singleton design patterns, the division of tasks provides us convenience.

- Safa mainly worked on implementing the game engine of IQ Puzzler. Safa also added the game over logic, score calculating and level passing logic to our game. Furthermore, Safa has redesigned the game UI by using royalty free UI sets and photos.
- Alp worked on gameplay which includes user interactions like, putting pieces in correct places. Checking pieces positions due to game rules. Alp also added to additional feature, first of them is freezing ice blocks which make our game more challenging and second feature is that, appearing gifts to collect and gain extra score.
- Ege worked in level and piece mapping, user interactions and gameplay logic such as rotating pieces, board methods such as putting pieces to right places, checking piece boundaries, game class methods, the user class, main controller handler that deals with pieces and other components.
- Gülşah worked on leaderboard and database, some parts of Game class which connects User and Game classes and some parts of the user actions and events.

- Betim worked on determining the shapes and colors of the pieces and specification of the levels. She draw the images of board and pieces in suitable pixels and also, she worked on some diagrams and some methods.

While we were working on our parts, we were always communicating among each other and support each other while we had problem in our own parts. In addition to our meetings three times a week, we use GitHub for sharing our codes, documents and opinions when we were not able to meet.

After the second iteration, we have almost decided about and done with the logic and the design of the game. Therefore, we started to implementation with adding more functional requirements that we've decided and we could satisfy our expectations from the game, and we finished our project at the time given to us.

1.1. System Requirements

As we mentioned in our previous reports we aim our system to not to require high resolution graphics or additional environment except JRE. This game can be played properly in any computer and operating systems which require JavaFX libraries and Java Development kits without freezes or delays. We said we will use MySQL for database however we decided to use SQLite. SQLite was sufficient for our game and it was easier to integrate it to the game. SQLite requires a JDBC connector, JDBC library should be added to project directory for game to work properly.

1.2. User's Install Guide

- To see the source code and reports by using the provided link. https://github.com/AlpErturk/IQ-Puzzler-Pro-Violet
- You can copy the project by bash or command line. Also, you can download the ZIP file of the project. The game will work properly in all JRE installed platforms.
- JavaFX and SQLite JDBC connector libraries should be present in running environment. If not, you can include them easily by downloading from our github page.
- In our second iteration reports which are available on our GitHub page given above , there is more information about our implementation.

2. Changes & Improvements

In this section we explained the design changes since we made from the start. Most of the requirements are satisfied by the end of the implementation but we decided not to implement some of the parts we have mentioned.

2.1. Design Changes & Improvements

In the first iteration of our design report, our classes were not organized as our final iteration. Because after our first iteration was finished, we decided to some changes in our second iteration to provide convenience and time savings. We added one class named GamePlayController. This class initializes board matrix, one matrix is used for holding x positions and one used for holding y positions. Availability of board entries will be determined by this class.

We used the Facade Design Pattern while we made a class which was a interface for database and connects our code to database and this provided us to deal with database clearly. We changed our design into MVC after the first iteration. Before the first iteration, we had a big controller class which was for gameplay logic, but we figured out that this class got very big and complex, so that we could not be able to debug it. So we decided to separate this class to Model, View and Controller. After doing this we solved a big problem in level passing so passing to MVC design pattern was a good choice for us.

After, we used fixed size of components in our first implementation; we noticed that this design could result in some problems in computers which have different monitor size. To cope with this problem, we changed the component sizes accordingly; program now determines the size of board and pieces by looking at the current PC's screen size. We also provide the rotation functionality for pieces.

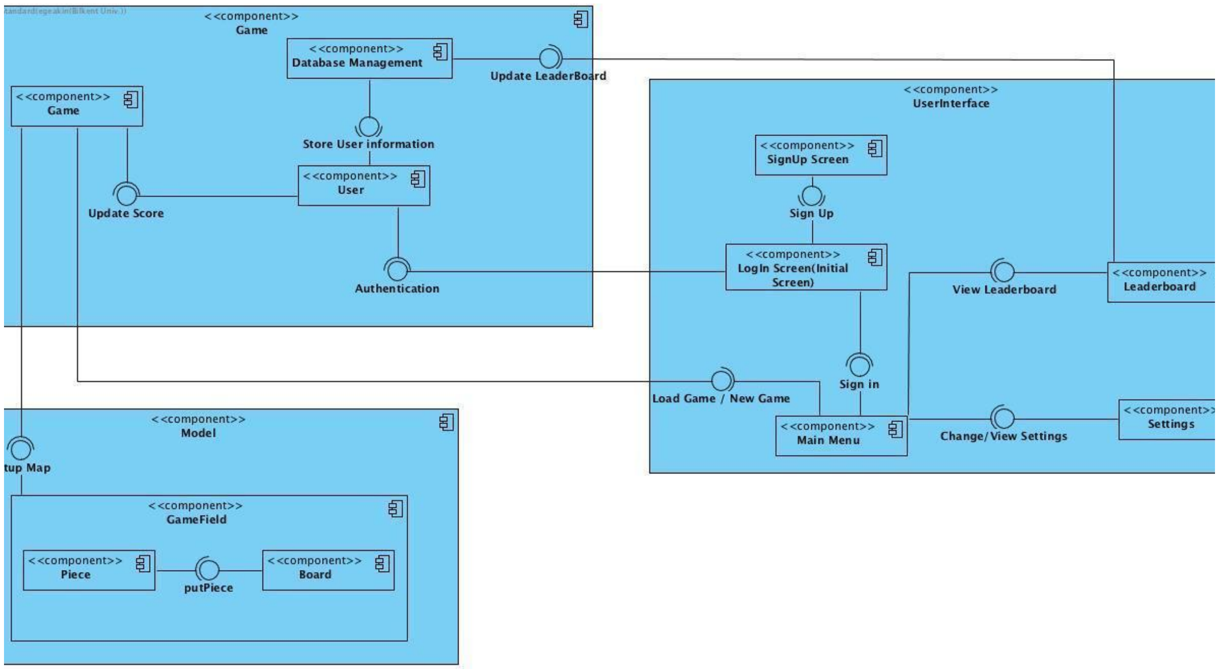
Although we were planning to use file system and database, we realized that the way we use it might be problematic, so we reconsidered the conditions and decided to use only database. Database is sufficient for our game.

For making our code easy to read and easy to debug, we had reviewed each other's code and we changed some classes' internal design. On the other hand, due to fact that we follow object oriented design principles, these changes did not influence other components that much.

To ease future improvements and avoid maintenance problems, we replaced some of the methods into proper classes and fixed some object oriented problems.

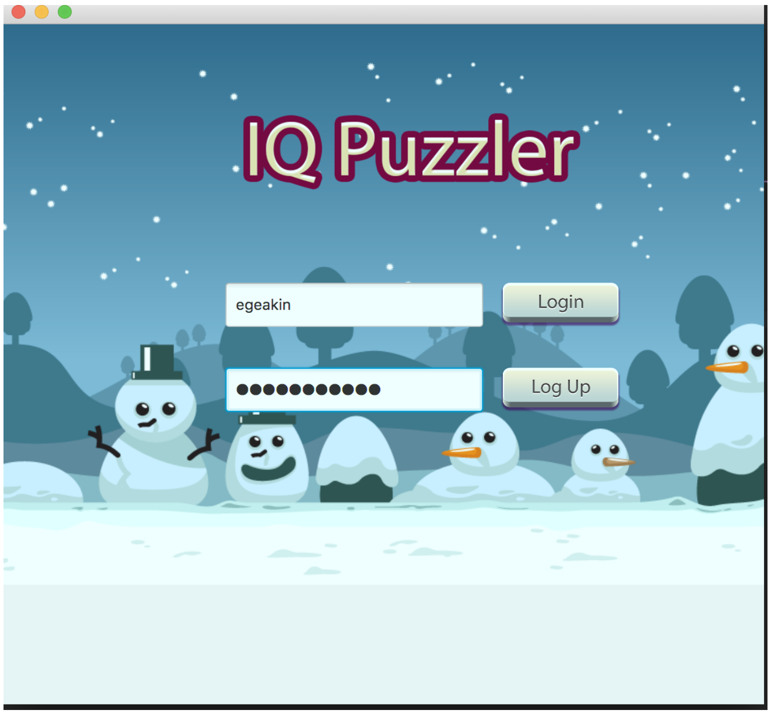
In order to make it more attractive, we added an extra scoring feature to our game. From a random place on board, a gift appears on random empty position on board in given periods, when user clicks these gift positions he gains extra score. For an another additional feature, during any level random empty places on board freezes and in order to put piece on these freezed places user should click hammer button to select hammer and click on ice blocks to break them. Then user will be able to put pieces.

Our design is improves as below:

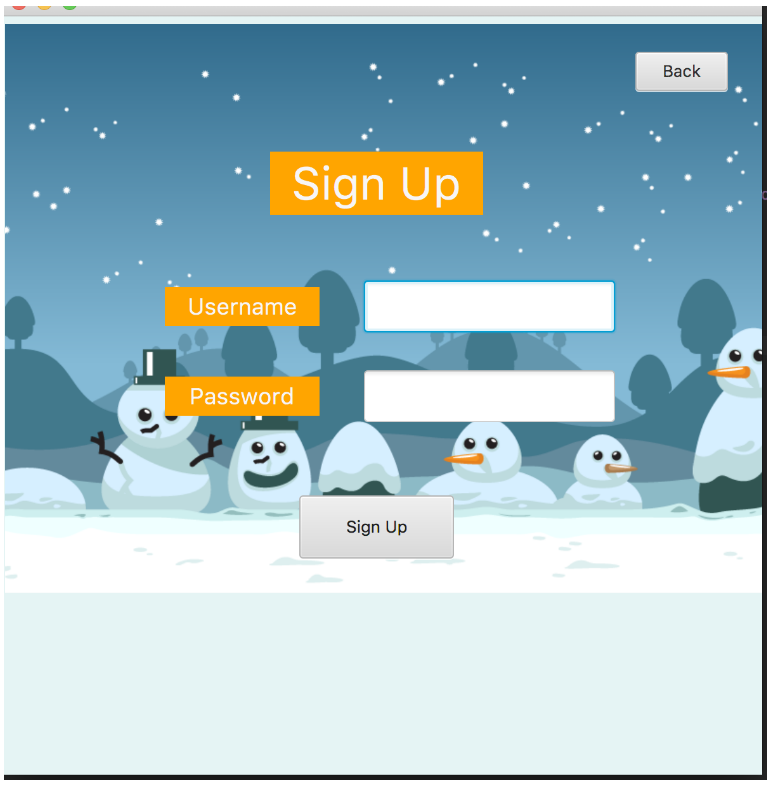


2.2. User Interfaces

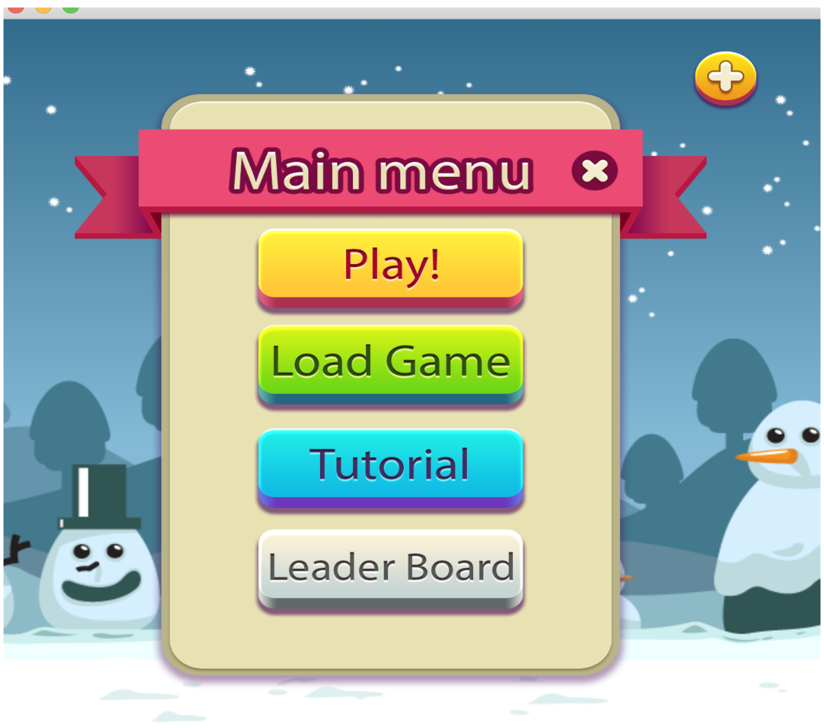
- **Log-in Screen:**



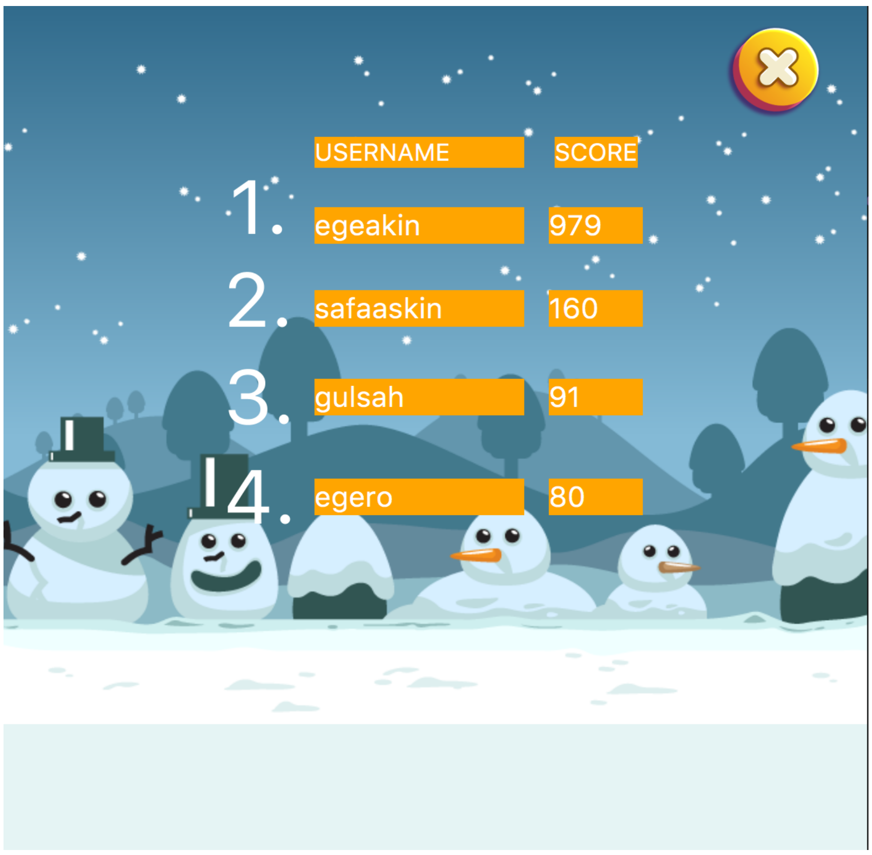
- **Sign Up Screen**



- **Main Menu**

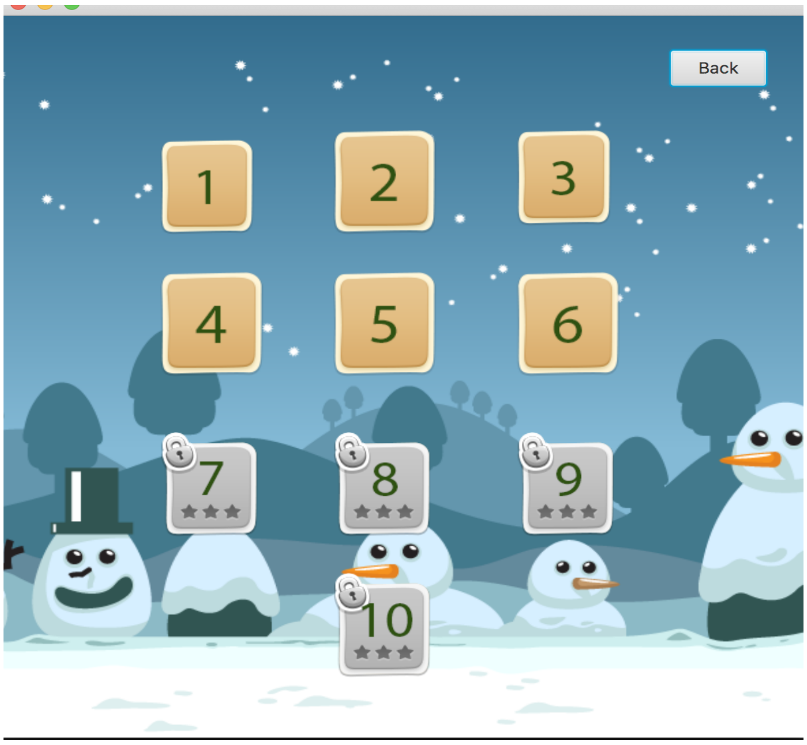


- **Leaderboard Screen**

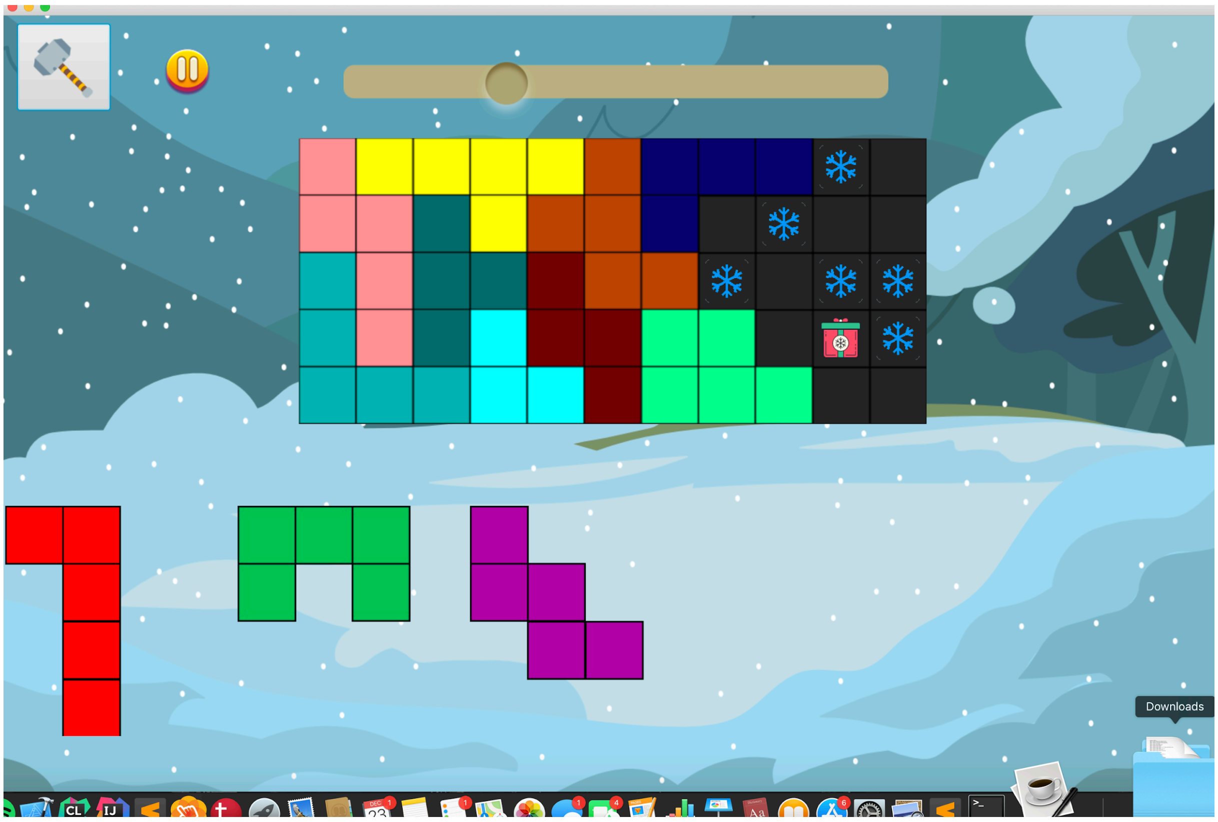


	USERNAME	SCORE
1.	egeakin	979
2.	safaaskin	160
3.	gulsah	91
4.	egero	80

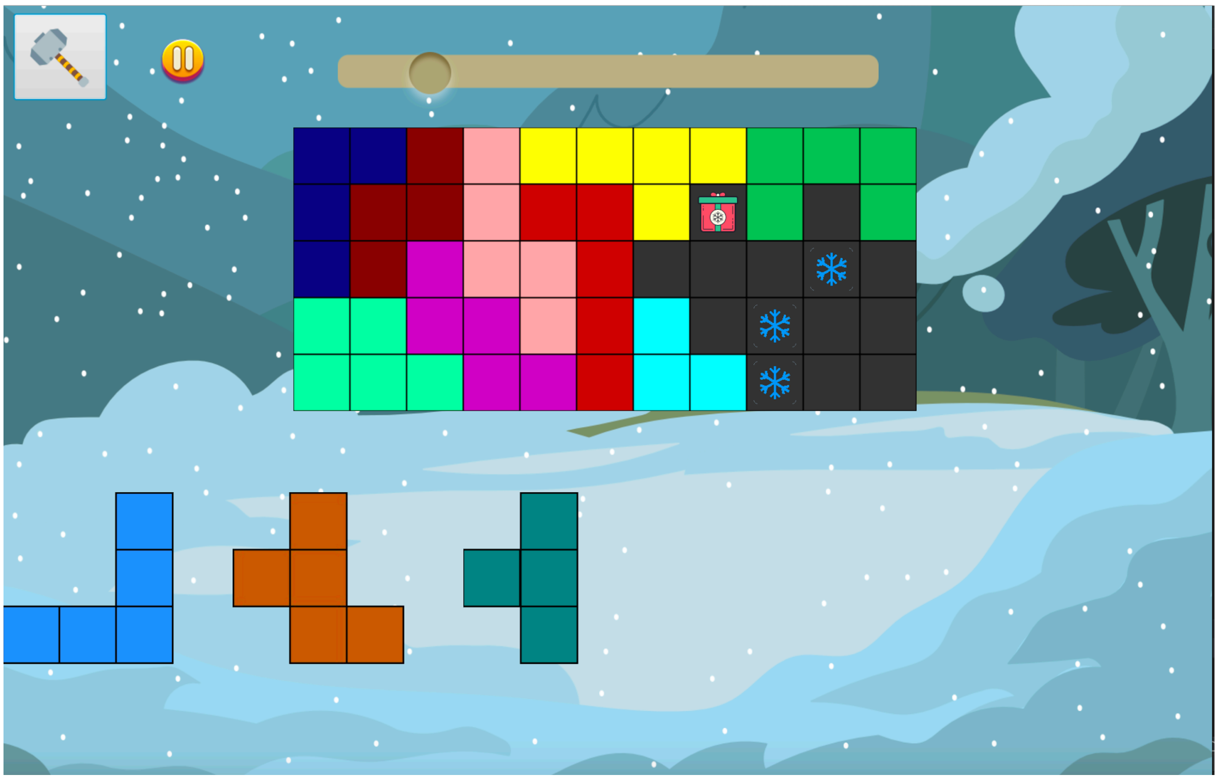
- **Load Game Screen**



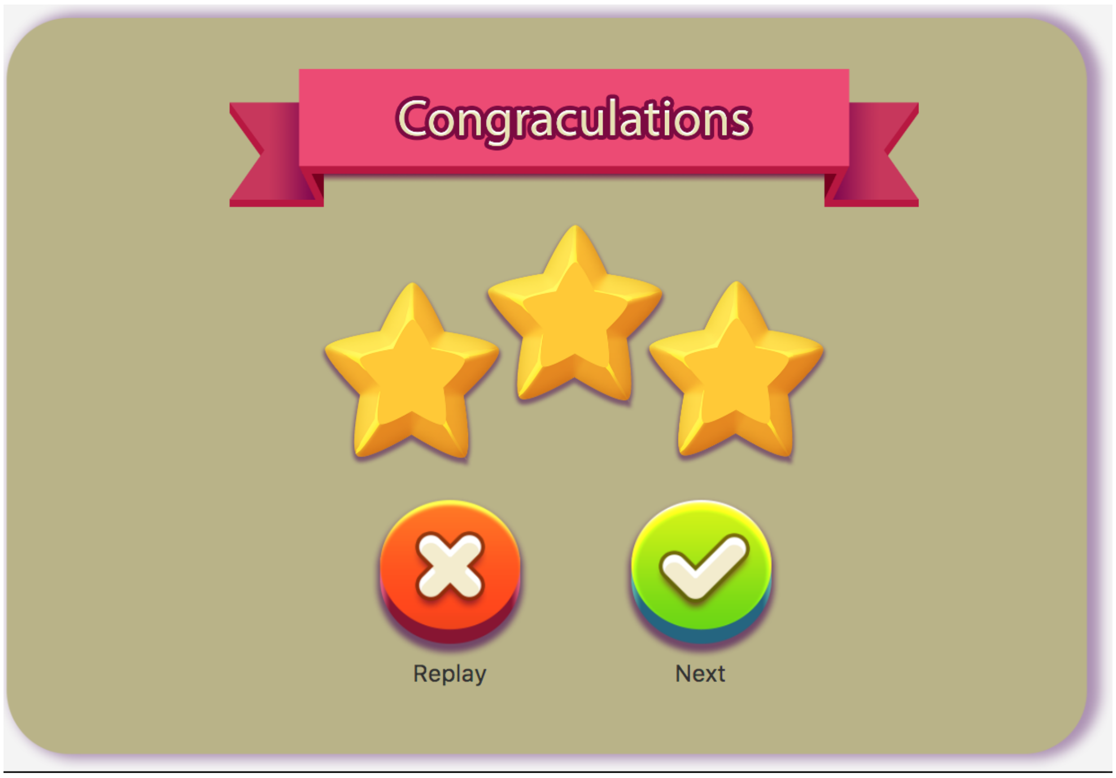
- **Gameplay Screen(Gift and Freeze Extension)**



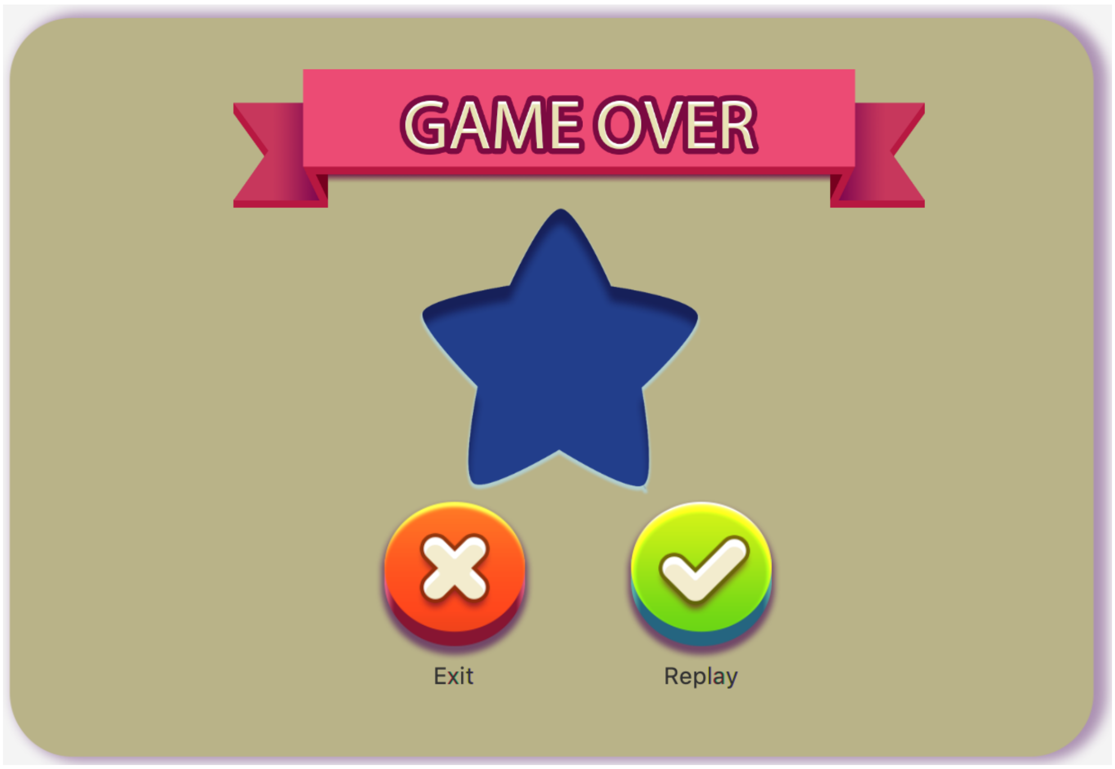
- **GamePlay Screen(Another Level)**



- **Congratulations Screen after level successfully passed**



- **Game Over Screen when time is finished.**



3. User’s Manual

After the manual installation whose instructions given above, the game guide is given below.

a. How to Start

In the main menu whose screenshot is given below, users have different options available via buttons. The first screen of our game is the main menu. “Play!” button starts a new game and the “Load Game” button allow players to continue their previously saved game.

b. How to Play

After pressing 2D

Players can access the information about how to play the game from by pressing the button “Tutorial”.

c. How to Proceed

The goal of the player is to place geometric pieces of different colors and sizes to board and fill it. Pieces will be given such that there will be one unique solution for each level. Therefore, the player can complete the game if he places the pieces completely in their correct positions. Level is successfully finished if the board is completely filled with the given pieces. Players will obtain a score for each successful level. Time and count of moves are not important to finish the game but it will be important for score.

4. What is Missing?

Although we have mostly fulfilled our functional requirements, we have changed and abandoned some of our design goals. On the other hand, there were also some additional features as we mentioned about.

We were planning to implement a 3D version of the game however we decided that it will be better to implement a nice 2D game rather than a regular 3D game. We thought it will provide a better experience to our users so we spent time for a better 2D game. We mentioned about the settings in our previous reports however we decided to not to implement it because it was unnecessary for our game.

5. Conclusion

In general, we have tried to do our best with applying the “Principles of Object-Oriented Design” and we think we have done a good job in the direction of our expectations. We have learned how analysis and design processes work beside implementation. While we were implementing our project, we had benefited from the advantages of analysis and design. It provides us a more planned and better designed implementation process. Each of us was able to carry out the tasks given by helping each other. As a result, we did our best to fulfill the requirements of our project and learn from it.