Bilkent University

Department of Computer Engineering

CS-319 Project

Quadrillion

Final Report

Group No: 1H
Group Name: COGENE

Osman Orhan Uysal

Samet Özcan

Mehmet Alper Karadağ

Ziya Erkoç

Talha Murathan Göktaş

1. Introduction

We have chiefly completed the functionalities that we promised in highest user-friendly way that we could achieve. Since our skills are limited in UI/UX we could not design extremely elegant and appealing User Interface. Nevertheless, our core game functionalities are up and running which we prioritized more. Below are the information about the completed and uncompleted features

We had completed the following functionalities during the first iteration:

1. Game can now be both played in single player or ranked mode.
2. User can compose a new level and send it to the database.
3. Users can select levels (either user-created or predefined).
4. User can see leaderboard of the level they are playing.
5. User can buy hints. Currently, no real money transaction occurs but fake money.

As of second iteration we completed:

1. Hint Feature
2. UI/UX Improvement
3. Back board

Below are the ones that we have yet to complete along with the explanations of why they are left unimplemented:

1. Pieces are not structured well within the game area
   We actually implemented a version where each time 4 pieces are shown to the user and through up-down arrows (like a scroll-bar), the player could travel between 4-piece groups. Yet, this way seemed problematic because players would not see the all pieces together.
2. Hint Purchase
   Hint purchase mechanism is working in implementation-wise without real-money involved. We wanted to make it live and use real money but our Payment System Provider *iyzico* warned us that we could not do that because we are not a company, we do not have a tax-plate and they do not support in-game purchasing. We realized that all other similar payment systems require E-Commerce site and tax-plate. Hence, we left it in the experimental mode with no real-money involved. As a result, payments can be made only through the fake credit-cards that are defined by the *iyzico*.

2. Work Allocation

- Orhan Uysal
  - Created the skeleton code out of UML diagrams and implemented some basic functionalities.

    - Implemented the feature of showing hint
    - Came up with an algorithm to find out if a board has a solution.
- Mehmet Alper Karadağ
    - Prepared Sequence Diagrams
    - Prepared State Diagrams
    - Implemented the following features of Ground: rotation, flipping, moving, dragging.
    - Initialized Payment System
- Talha Murathan Göktaş
    - Prepared Activity Diagram
    - Created Demo videos
    - Contributed to UI
- Samet Özcan
    - Prepared Use Case Diagram and its textual represantation
    - Contributed to UI
- Ziya Erkoç
    - Prepared Class Diagrams and Subsystem Decomposition.
    - Prepared State Diagrams
    - Prepared initial GUI elements
    - Implemented the following features of Piece: rotation, flipping, moving, dragging and the business logic of Piece placing.
    - Maintained Database interactions and Payment System.

3. Lessons Learnt

Having UML diagram before implementing relieved our burden a lot. Since we made sure that our logic works in UML, we just figuratively converted UML diagram into code. We also realized the power of Design Patterns and harnessed it a lot in our projects. One of our confess is we gave so much responsibility to Core Game subsystem and especially to LevelManager class. Hence, our members who worked on that part did a lot of job. We could have divided the system more evenly. Also, we tried to position UI elements (i.e. buttons, text fields, labels) on our own instead of making use of JavaFX SceneBuilder. We could develop an architecture that would allow us to use SceneBuilder.

4. User's Guide

a. System requirements

Quadrillon requires JVM (Java Virtual Machine) to run on a machine. Also, it requires internet connection to login and purchase hint operations. Since the data of levels and leaderboard records are kept in local database, computer to run the project must have a local database.

b. How to Build using Intellij Idea

- Open project with Intellij Idea IDE.

- Select Project Structure under File menu (Ctrl + Alt + Shift + S).
- Select Artifacts from the menu on the left.
- Select JAR as Type.
- Make sure that the manifest file and main class (ui.Screen) is specified correctly
- Click Apply and OK.
- Select Build Artifacts under Build at toolbar.
- Build the jar.
- Congratulations you have successfully built the project.

c. Installation

- Go to the link below to see the project and reports
  https://github.com/Rgtemze/Quadrillion-CS-319
- Download and open the quadrillion.jar file
- Congratulations you have installed Quadrillion.

d. How to use

- First user has to log in to his account if he has no account he has to a create new account.
- In main menu user can select to play a casual game, ranked game or he can compose a new level.
- In casual game user can use hints select which level to play.
- In ranked game levels are given to user randomly and user cannot use hints. Also, his records after finishing the game are recorded into that level's leaderboard.
- In compose level user can create his own level by moving, rotating and flipping the 4 grounds.

e. How to play

- The purpose is to fill all empty spots with pieces.
- User should drag and drop the pieces in order to complete the game.
- Pieces cannot be dropped even one of the circles overlap with an unavailable spot.
- Pieces can be flipped with right mouse button and rotated with middle mouse button.
- User can use hints if he gets stuck.
- User can buy hints if he has no hint.
- If user can fill all the spots he can click submit button to complete the game.

f. How to compose level

- The purpose is to create new challanges for players by discovering new board combinations among quadrillion possibilities.
- User can drag and drop grounds.
- Grounds can be flipped with right mouse button and rotated with middle mouse button.

- User can click the check button after finishing composing level, if the level is valid, namely it is connected and has valid solution.

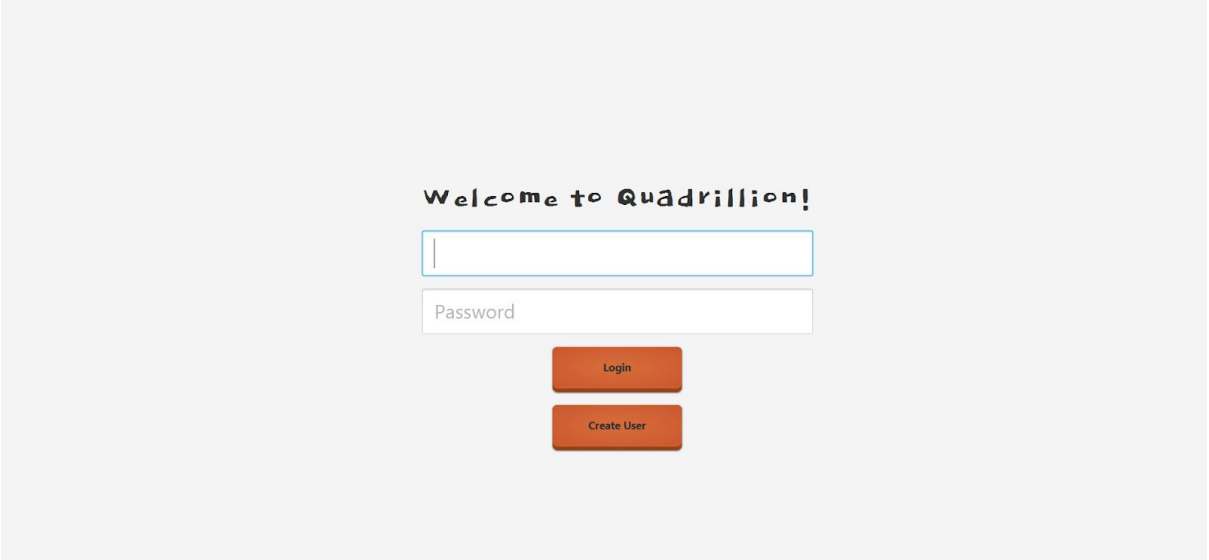

Figure 1: Login Screen

Figure 2: MainMenu Screen

Figure 3: Leaderboard Pop-up

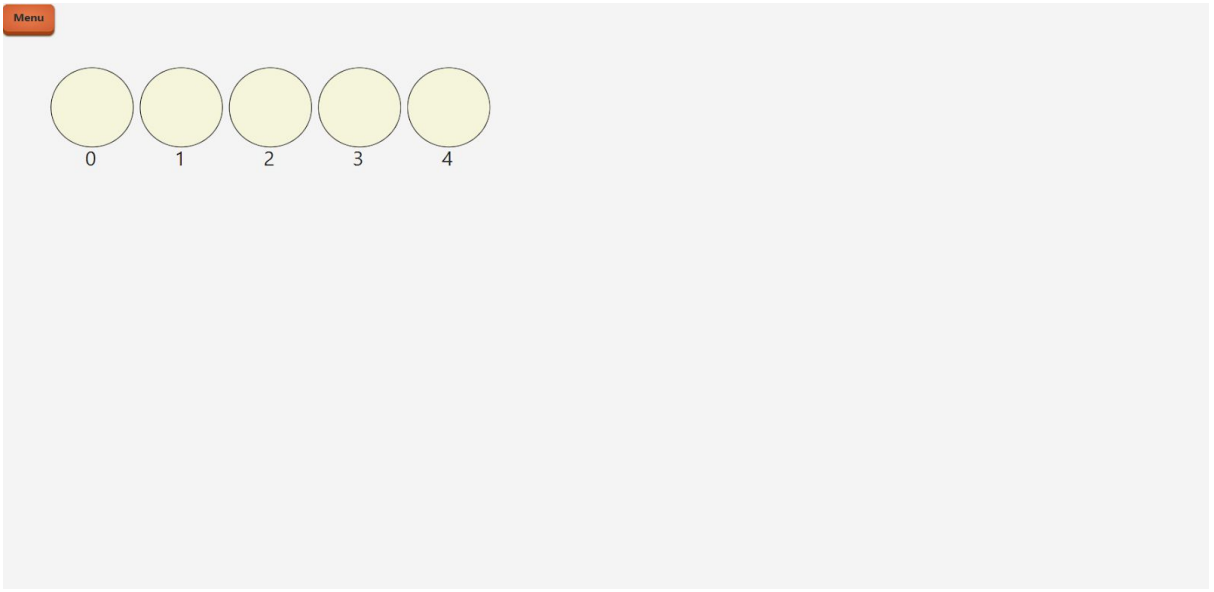

Figure 4: SelectLevel Screen

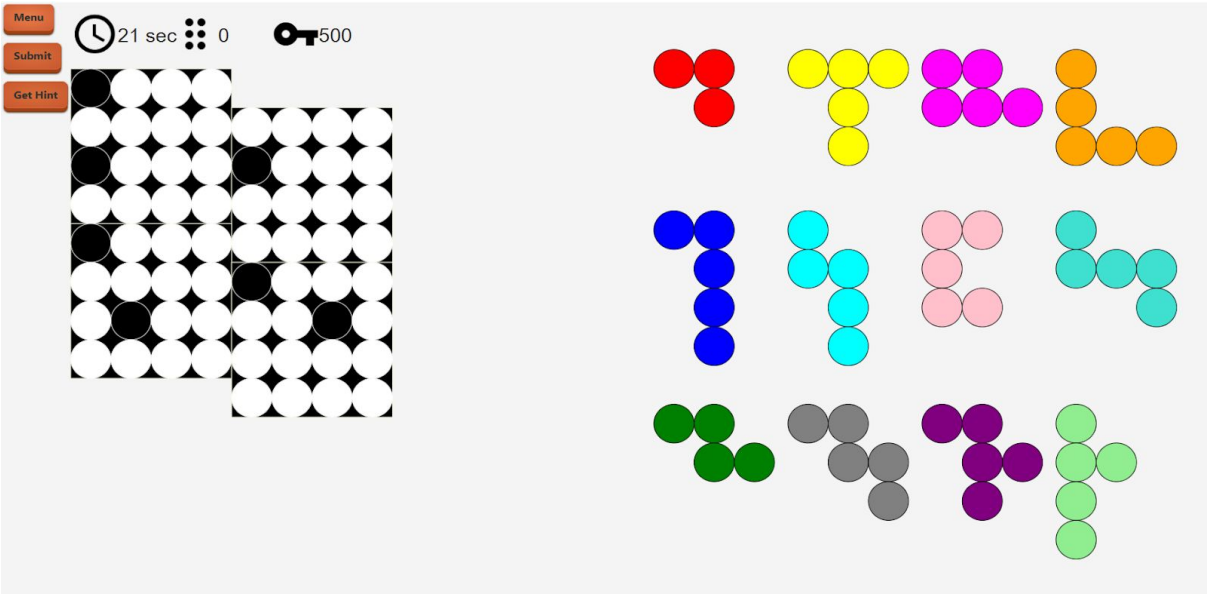

Figure 5: PlayGame Screen

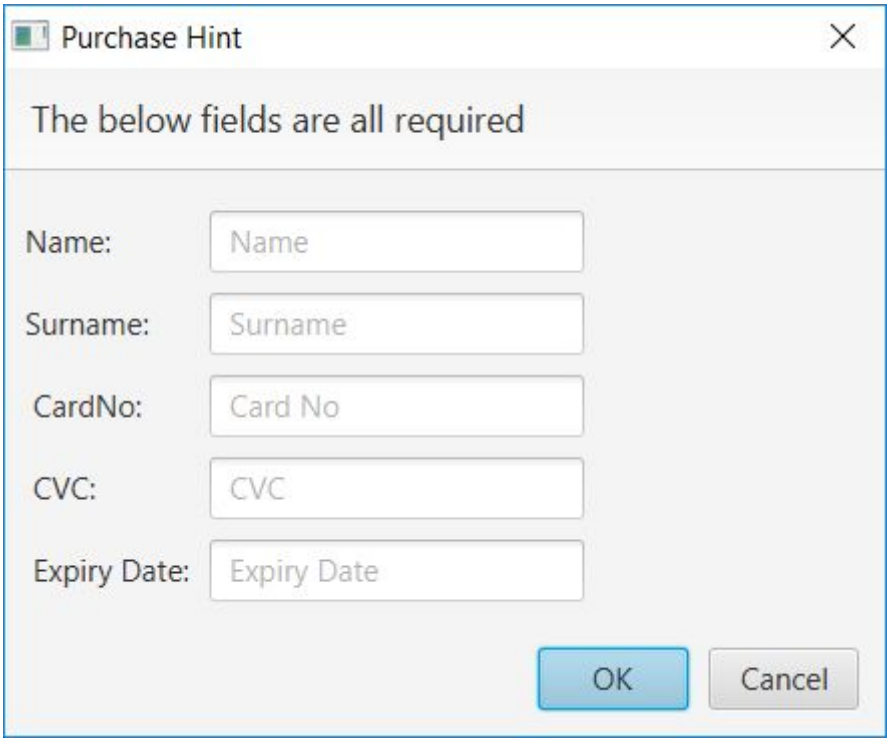

Figure 6: Hint Purchase Popup

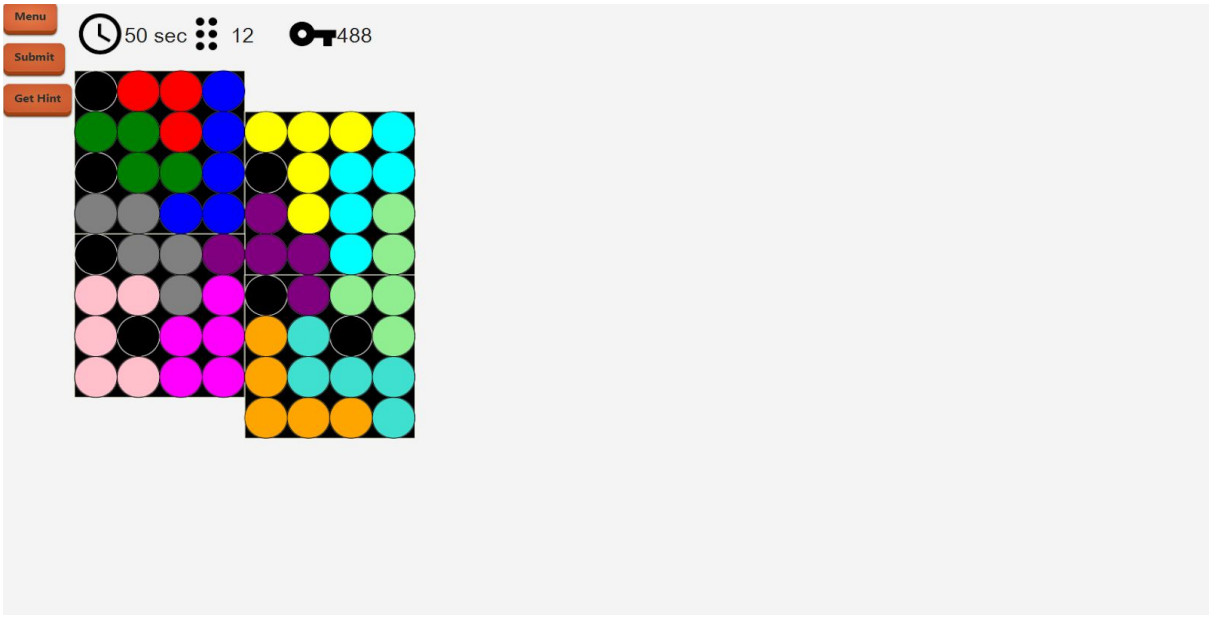

Figure 7: An example of Completed Level

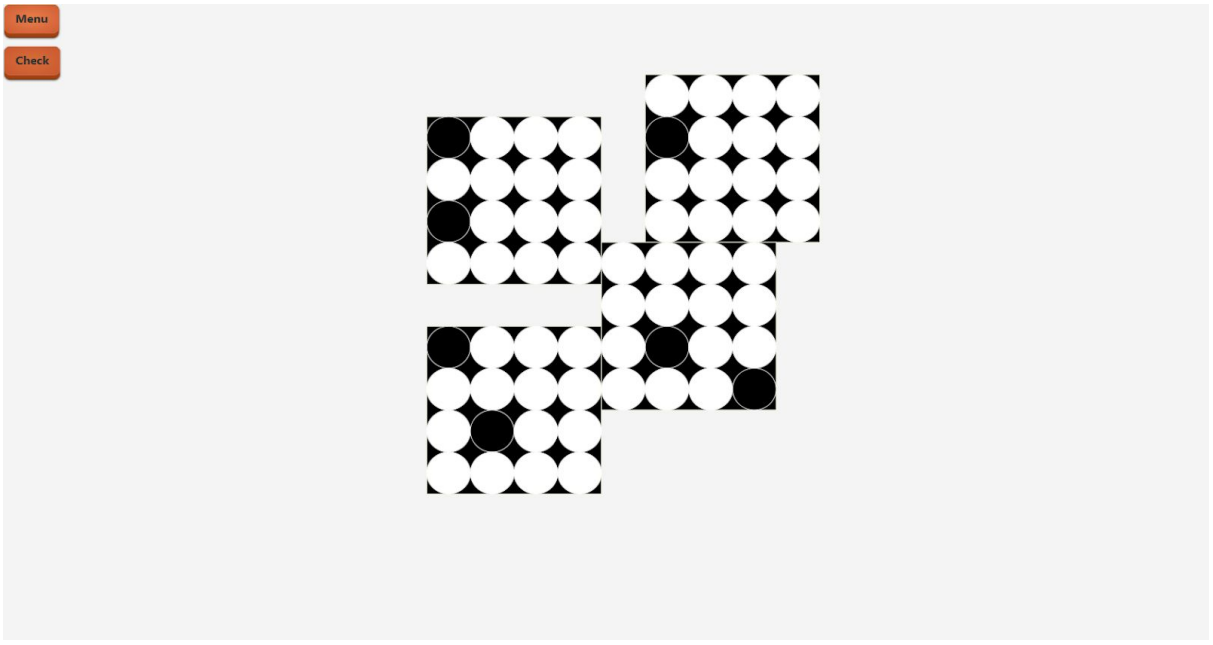

Figure 8: ComposeLevel Screen

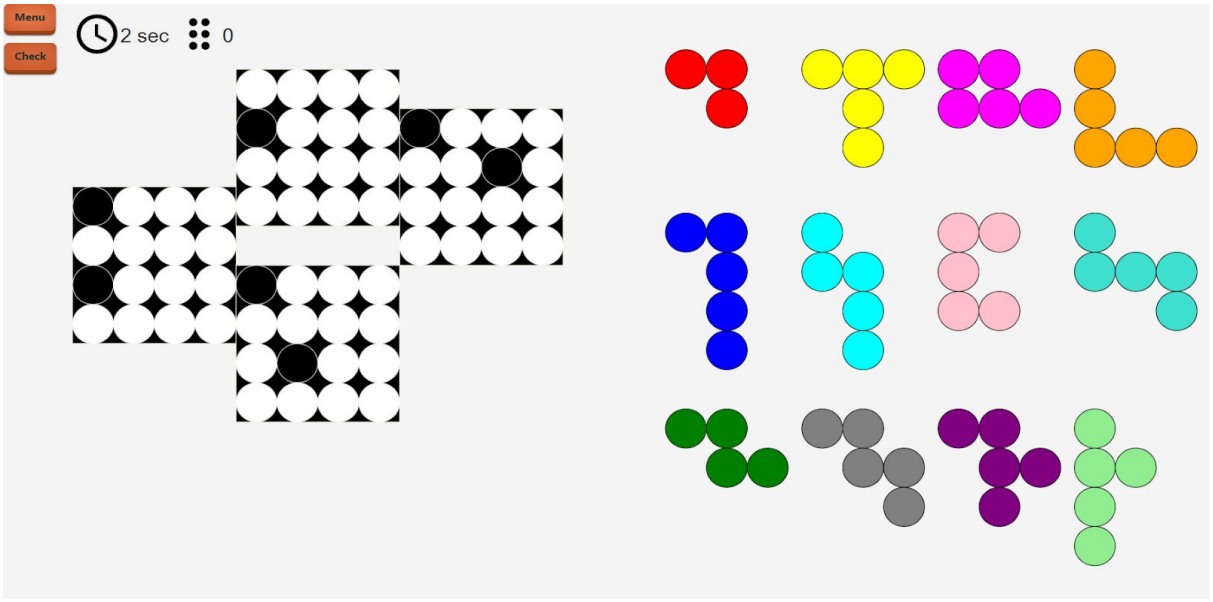

Figure 9: PlayRanked screen