QuadZillion

CS319 Iteration II Design Report

Bilkent University Department of Computer Engineering

Group 1G Supervisor: Eray Tüzün

Berk Güler
Enver Yiğitler
Kasım Sarp Ataş
Melike Arslan
Ufuk Bombar

Table of Contents

1. Introduction

1.1 Implementation Process

Implementation of QuadZillion is divided into three main parts: Core implementation of the game, interaction between panels which is the graphical user interface (GUI) of the game and data management. The implementations of these parts are mostly completed other than maybe some minor GUI related issues.

The core of the game is generally made up of the models, pieces and the levels of the game. In addition to this, the game panes of each modes and settings, moves of the pieces are implemented in the core of the game. Inside models the main board which is made up of the grids and tiles are implemented. For the moves of the pieces a move checker class is implemented that controls whether the move is valid or not.

The GUI of the game is made up of the controllers for each panel and a Utility class that includes the general controls that each panel shares. Inside each controller each scene and on these scenes the actions for the buttons are implemented.

For data management we have a separate package called resources that includes all the images that the core and the GUI of the game contains, the layouts for each scenes, the sounds that are played, the tutorial video that explains how to play the game and the three themes that we have for the game's interface.

1.2 Work Allocation

Two people worked on the core of the game, one person worked on the panels (UI), and two people worked on the data management. However, each one of us supported others in some way in implementation process for maximum efficiency, since there is a deadline for this project, we cannot be stuck at one problem for days. It must be solved quickly and efficiently.

Deciding who is going to do which part was difficult at first. However we overcame that problem by making list of our strengths and weaknesses. After making the list of our strengths and weaknesses, we assigned each member the corresponding part. First we decided to implement the layers of UI, since it is the first part that the user interacts with. And then adding the data management part for soundtracks and saving the players' preferred options. Last, gameplay itself which is the core of the game was decided to be implemented.

User Interface of the main menu part, how to play part, credits part were implemented without problem and were tested in advance. Also there is a music feature that can be turned off and on.

In the beginning of second iteration, we decided to add more game modes to change the original game and make it more appealing. First, we come up with some modes that are accepted by all team members. Secondly, we changed the current UML diagrams to according to modes that we come up with. Changing UML diagrams was necessary, since this course's one of the main objective is programming, that is based on diagrams and reports. Therefore we changed our diagrams and reports correspond to our new game modes. This brings our project flexibility and our code can be modified certain changes in the future as well.

- Berk Güler was mainly responsible for GUI implementation and helped with core.
- Ufuk Bombar contributed both to GUI and core.
- Enver Yiğitler worked mainly on core.
- Sarp Ataş helped with core.
- Melike Arslan helped with GUI and prepared the video. (Snack provider)
- All of us contributed to reports and diagrams

2. Changes & Improvements

We have changed major parts of our code base during the second iteration. We had to redesign our diagrams from scratch to serve our needs better. We tried to apply design patterns and try to follow them as best as we can. Singleton pattern was used in the Util and GameApplication classes and factory pattern was used in the PieceFactory to create pieces.

2.1 Design Changes & Improvements

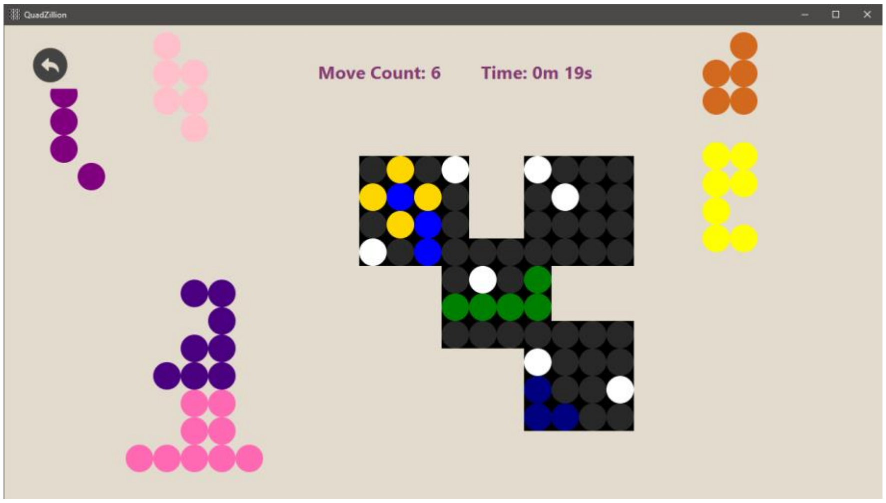
We have changed major parts of our code base during the second iteration. We had to redesign our diagrams from scratch to serve our needs better. We tried to apply design patterns and try to follow them as best as we can. Singleton pattern was used in the Util and GameApplication classes and factory pattern was used in the PieceFactory to create pieces.

2.2 New Game Modes

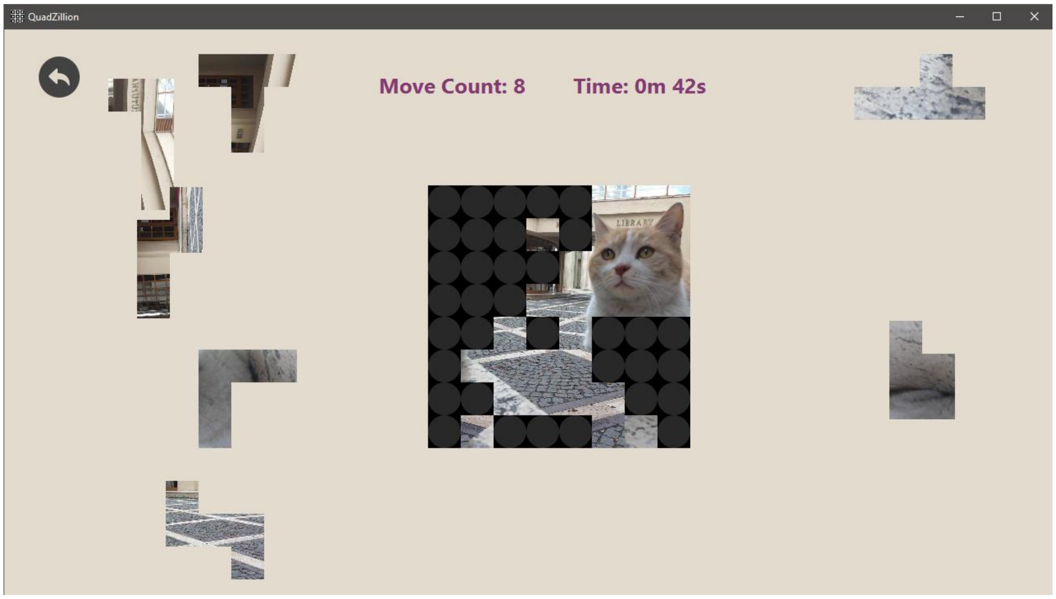
Our game has several different game modes for player to enjoy the game without repetition.

2.2.1 Extended Game Mode

Extended game mode allows this game to not limited only 4 grid. Main levels are created around the grids and pieces generated based on grids.



2.2.2 Puzzle Game Mode



In Puzzle Mode player will provided a picture. Main objective of this mode complete the picture, corresponding to given pieces.

2.3 New Features

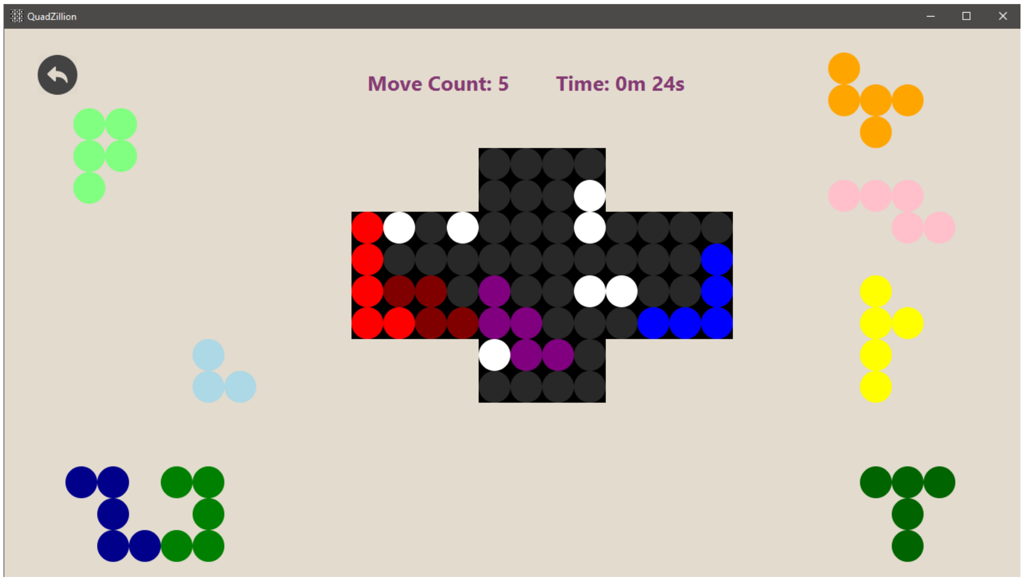
2.3.1 Move Counter

We decided to add a move counter for player to see his or her progress through the game. Main purpose of the move counter is encouraging the player to achieve the best result.

2.3.2 Timer

We decided to add a simple timer that starts when level is started. Main purpose of this timer is user to see its progress and speed when solving the level.

Move Counter and Timer



Move Count: 5 Time: 0m 24s

2.4 What is Missing

We decided against implementing High Score Board and Hint Request. Other than that every promised feature was implemented.

3. User’s Guide

3.1 System Requirements

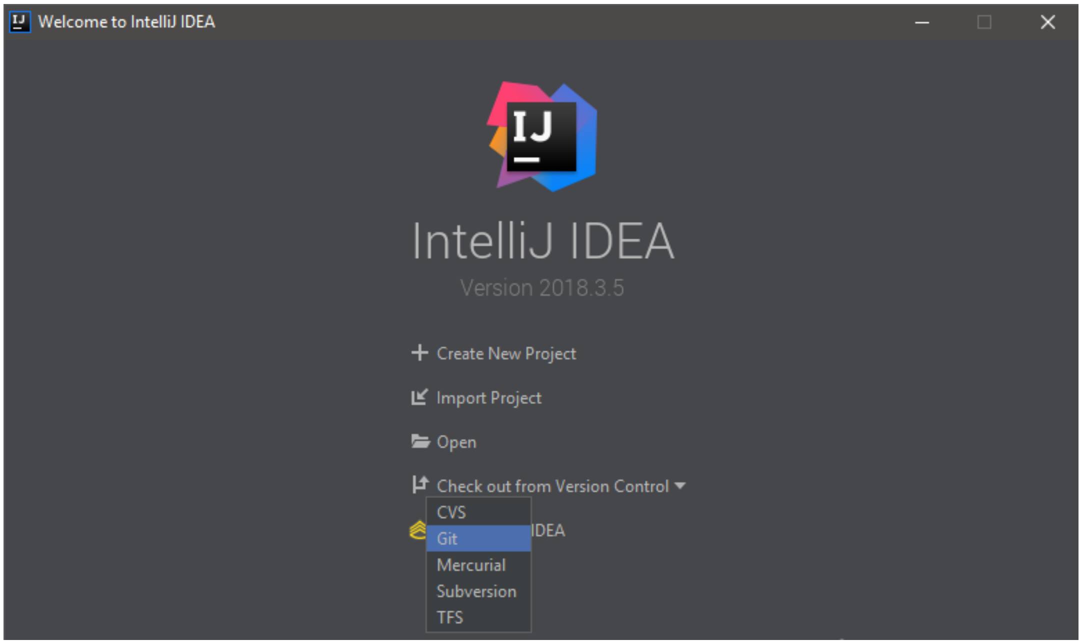
The game can be played on both Linux distros, Windows and macOS as long as Java Runtime Environment(JRE) is installed on the target machine.

3.2 Installation Guide

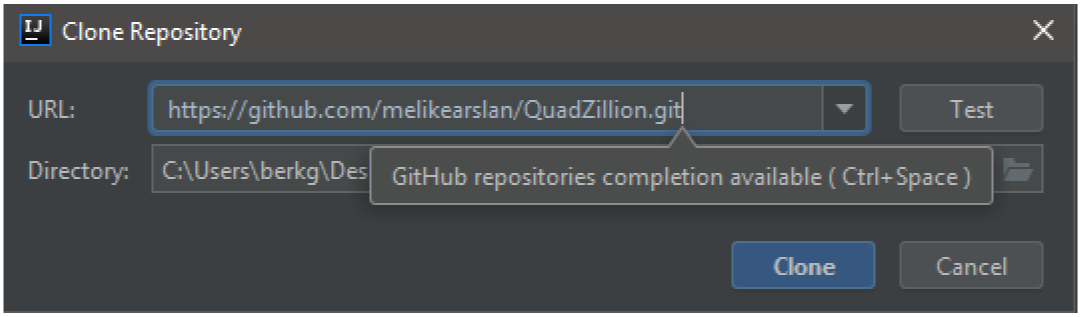
In order to play the game users can either download the JAR file and start it by double-clicking it or clone the repository from Github using Git and compile the source files with the help of an IDE such as IntelliJ or Eclipse on their machine. JDK(Java Development Kit) is also required to compile the source files.

Option 1:

Start by opening the IntelliJ IDEA. Click the Check out from Version Control and select the Git option.



Copy the the Github link (https://github.com/melikearslan/QuadZillion) and paste it to the URL field. Make sure to add .git extension to the end, click "Test" to ensure there is connection and finally clone the repository.

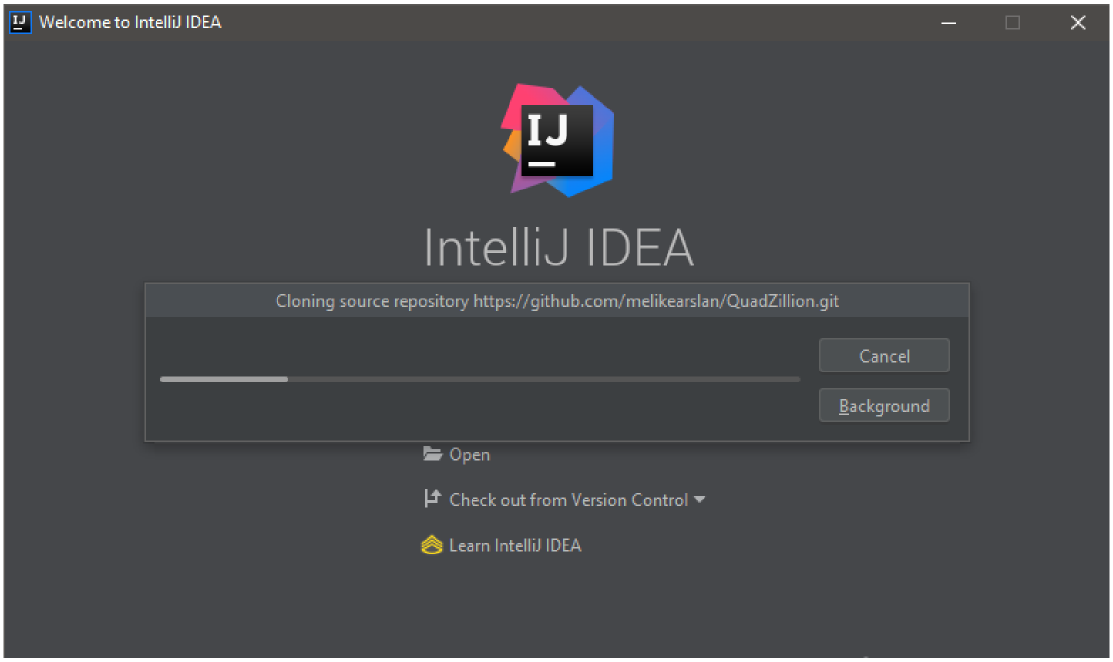


URL: https://github.com/melikearslan/QuadZillion.git

Directory: C:\Users\berkg\Des

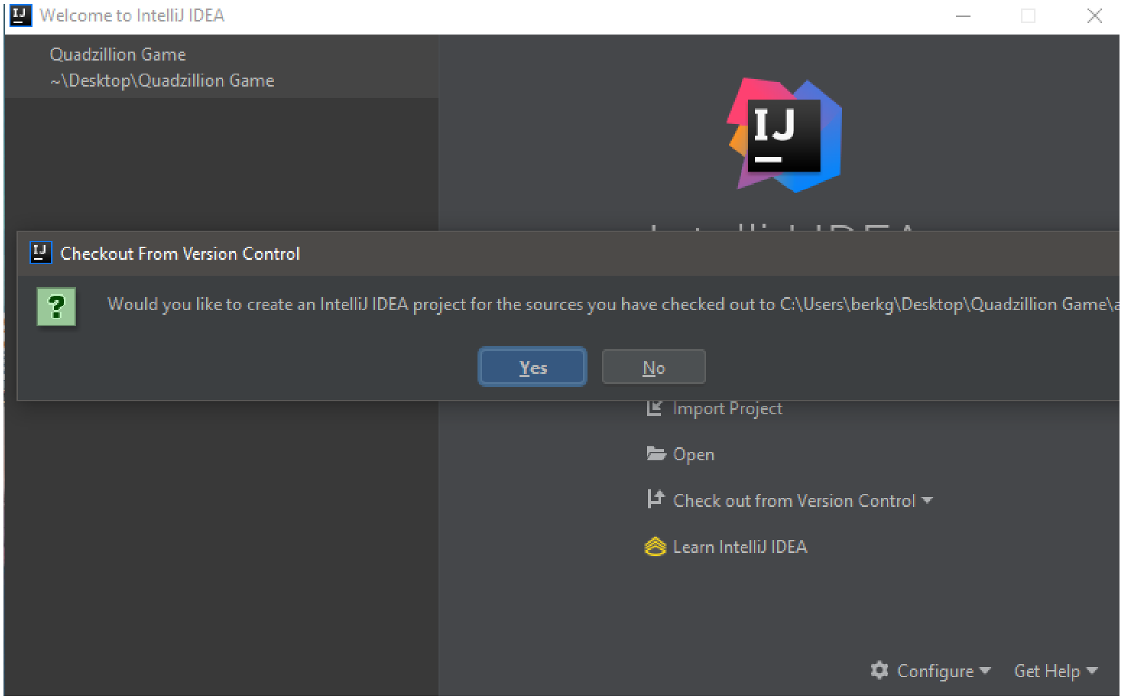
GitHub repositories completion available (Ctrl+Space)

Wait for clone to complete.



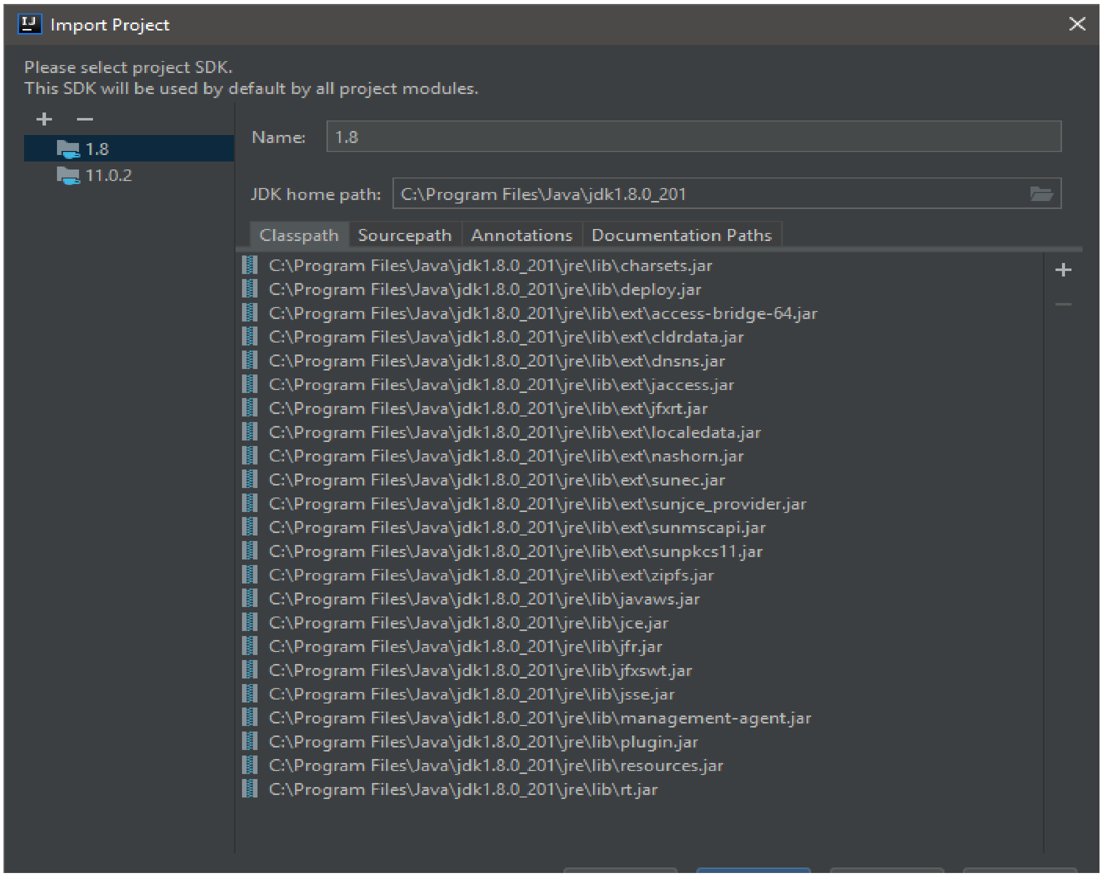
Cloning source repository https://github.com/melikearslan/QuadZillion.git

Select Yes.



Would you like to create an IntelliJ IDEA project for the sources you have checked out to C:\Users\berkg\Desktop\Quadzillion Game\a

Keep clicking next until the JDK selection menu appears.



Please select project SDK.
This SDK will be used by default by all project modules.

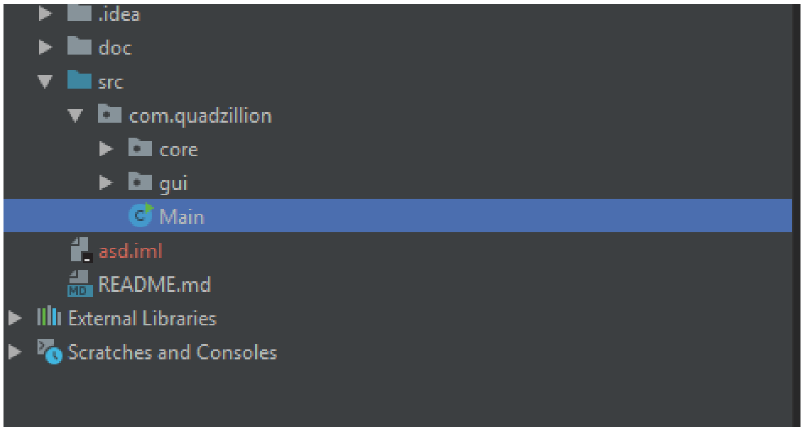
Name: 1.8

JDK home path: C:\Program Files\Java\jdk1.8.0_201

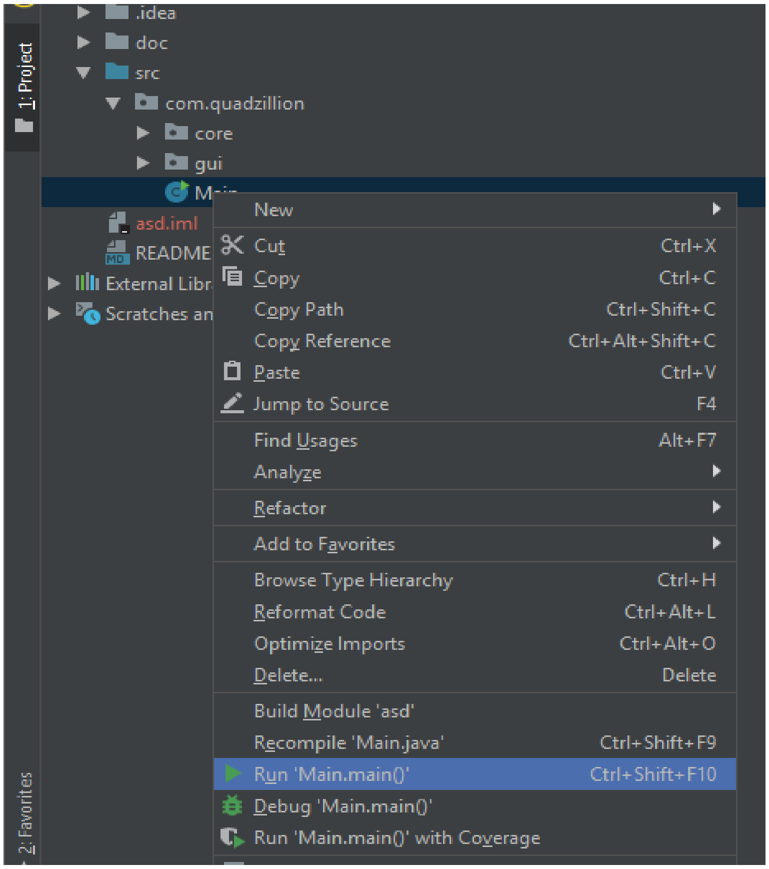
```
C:\Program Files\Java\jdk1.8.0_201\jre\lib\charsets.jar
C:\Program Files\Java\jdk1.8.0_201\jre\lib\deploy.jar
C:\Program Files\Java\jdk1.8.0_201\jre\lib\ext\access-bridge-64.jar
C:\Program Files\Java\jdk1.8.0_201\jre\lib\ext\cldrdata.jar
C:\Program Files\Java\jdk1.8.0_201\jre\lib\ext\dnsns.jar
C:\Program Files\Java\jdk1.8.0_201\jre\lib\ext\jaccess.jar
C:\Program Files\Java\jdk1.8.0_201\jre\lib\ext\jfxrt.jar
C:\Program Files\Java\jdk1.8.0_201\jre\lib\ext\localedata.jar
C:\Program Files\Java\jdk1.8.0_201\jre\lib\ext\nashorn.jar
C:\Program Files\Java\jdk1.8.0_201\jre\lib\ext\sunec.jar
C:\Program Files\Java\jdk1.8.0_201\jre\lib\ext\sunjce_provider.jar
C:\Program Files\Java\jdk1.8.0_201\jre\lib\ext\sunmscapi.jar
C:\Program Files\Java\jdk1.8.0_201\jre\lib\ext\sunpkcs11.jar
C:\Program Files\Java\jdk1.8.0_201\jre\lib\ext\zipfs.jar
C:\Program Files\Java\jdk1.8.0_201\jre\lib\javaws.jar
C:\Program Files\Java\jdk1.8.0_201\jre\lib\jce.jar
C:\Program Files\Java\jdk1.8.0_201\jre\lib\jfr.jar
C:\Program Files\Java\jdk1.8.0_201\jre\lib\jfxswt.jar
C:\Program Files\Java\jdk1.8.0_201\jre\lib\jsse.jar
C:\Program Files\Java\jdk1.8.0_201\jre\lib\management-agent.jar
C:\Program Files\Java\jdk1.8.0_201\jre\lib\plugin.jar
C:\Program Files\Java\jdk1.8.0_201\jre\lib\resources.jar
C:\Program Files\Java\jdk1.8.0_201\jre\lib\rt.jar
```

Select the JDK version installed on your machine. Click next and then finish.

Next navigate to left of the screen and find the Main file under the src directory.



Right click Main and select Run “Main.main()” and game should start. Enjoy the game!



Option 2:

Download the .jar file and run it by either double clicking on the jar file or navigate to the directory that contains the .jar file and run the command “*java -jar Quadzillion.jar*” from the terminal.

3.3 How to Use

3.3.1 Game Objects

There are several objects of the game that the player is required to know and get familiar with. These objects might vary according to each mode of the game.

3.3.1.1 Pieces

For the Vanilla mode, there are 12 pieces made up of circles each with a different shape and color. The shapes and colors of these pieces remain stable throughout the Vanilla levels. Each can be rotated and mirrored according to the level the player faces. The colors are

- Blue
- Dark Blue
- Light Blue
- Green
- Dark Green
- Light Lime
- Maroon
- Orange
- Pink
- Purple
- Red
- Yellow

For the Extended mode, the pieces may vary in shape and don’t remain stable throughout the levels which means that there might be different shaped pieces on each level. They can be rotated and mirrored according to the level the player faces.

For the Puzzle mode, the pieces are made up of squares rather than circles. There might be different shaped pieces on each level. They are a part of a picture, so when they come together they form a whole picture.

3.3.1.2 Main Board

In Vanilla mode, the main board is made up of 4x4 grids with forbidden points.

In Extended mode there is limitless possibility of grids. They can be interfered with each other and can have a custom size.

In Puzzle mode, the main board is made up of one grid that contains some parts of a picture. Forbidden points are used to give hints to the user.

3.3.1.2 Levels

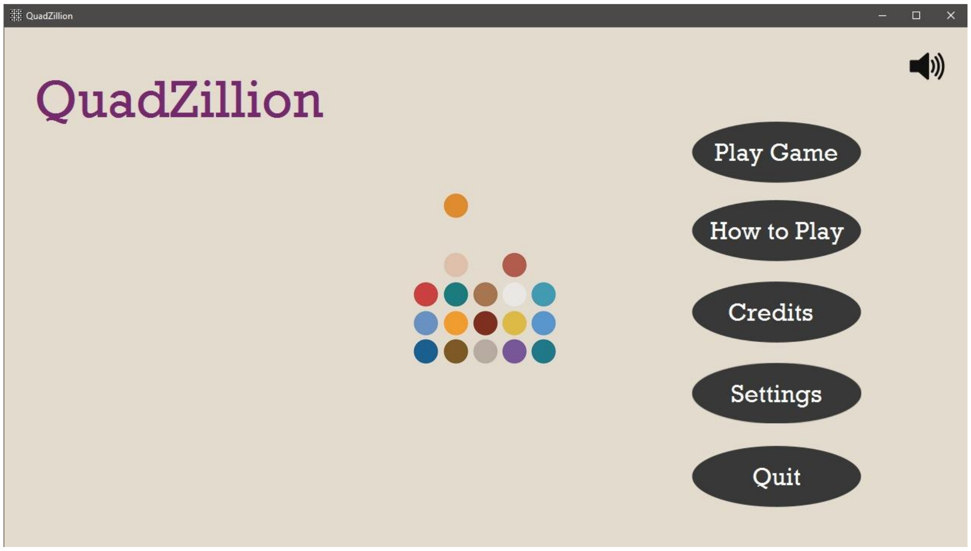
Every level contains different combinations of grids. Therefore, every level has a different solution.

Levels in Puzzle mode contains four 4x4 grids. That forms a square. The main difference of puzzle levels is the picture that user tries to solve. Every level contains different picture for user to solve.

Every levels in Extended mode contains different forms and number of grids that forms a unique board for the level.

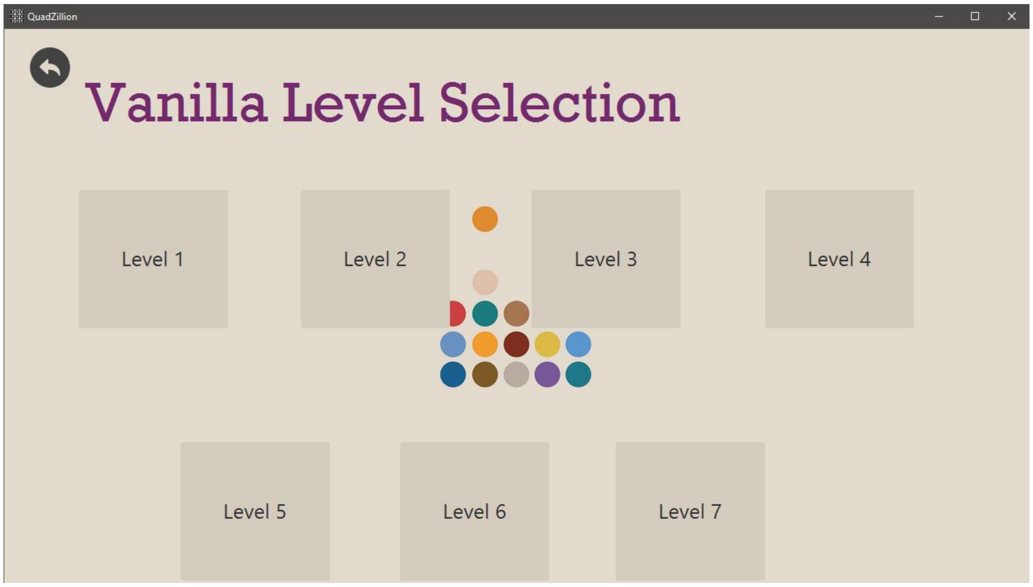
3.3.2 Menu Options

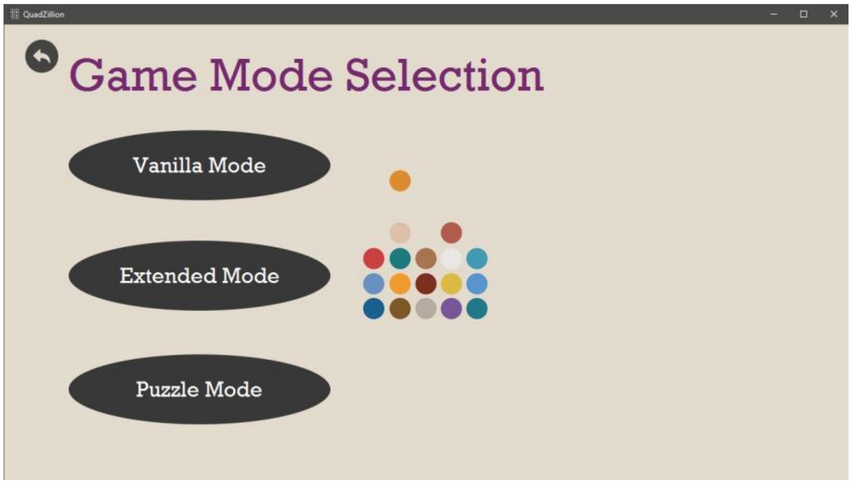
The player is provided with some options to choose on the main menu.



3.3.2.1 Play Game

When the player clicks play game, they will be provided with game mode selection menu.





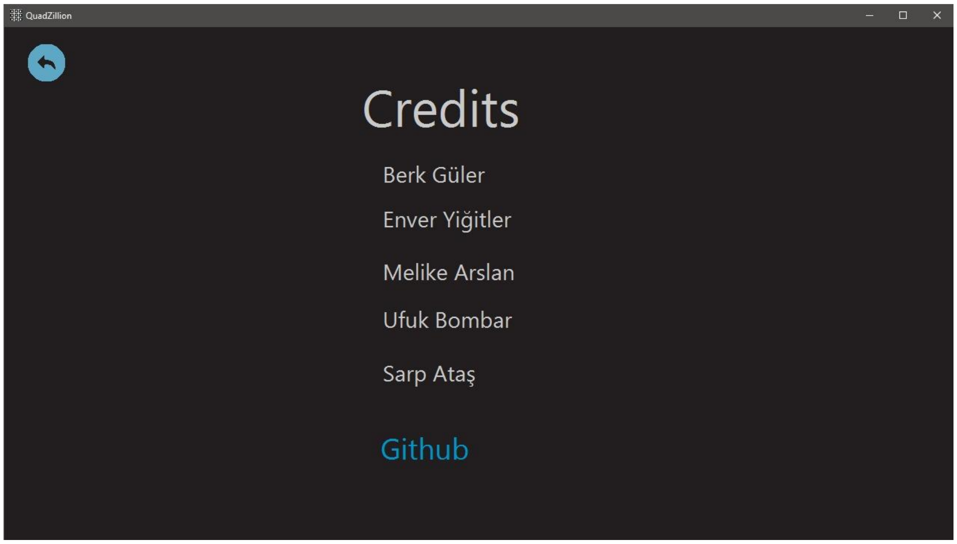
On each mode the player will be provided a selection of levels.

3.3.2.2 How to Play

When the player clicks How to Play, they will be provided with a video of how to play the game.

3.3.2.3 Credits

When the player clicks Credits, they will be provided with the names of the contributors and the github repository link of the game.



QuadZillion

Credits

Berk Güler

Enver Yiğitler

Melike Arslan

Ufuk Bombar

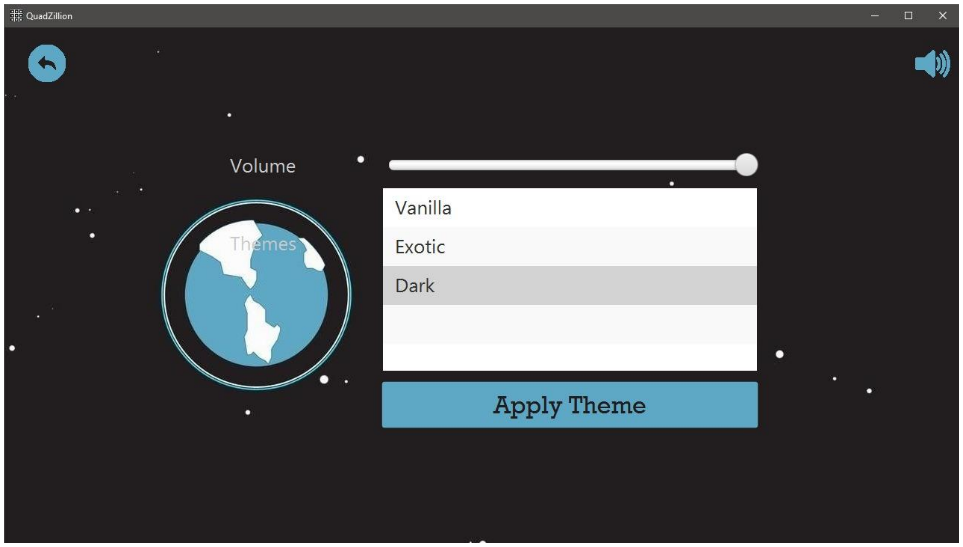
Sarp Ataş

Github

3.3.2.4 Settings

When the player clicks Settings, they will be provided with the selection of themes and the adjustment to sound.

According to the selection of the player, the theme will change, for example in the figure below the theme is Dark.



3.3.2.5 Settings

When the player clicks Quit, the screen will close and the game will be terminated.

3.3.3 Gameplay Operations

3.3.3.1 Mouse Controls

- Mouse Right-Click: Rotate the piece clockwise.
- Mouse Left-Click: Select a piece and move it across the screen. Hold Left-Click drag the pieces wanted place and release the mouse button for drop.
- Mouse Middle-Click: Horizontally rotate the piece.

3.3.3.2 On Screen Controls

- Back button: Returns back to mode selection menu.