BILKENT UNIVERSITY

CS 319

Object Oriented Software Engineering

FINAL REPORT

09 May 2019

QUADRILLION

by The Group

Ana Peçini

Krisela Skënderi

Muhammad Hammad Malik

Ünsal Öztürk

Table of Contents

1. Introduction

In this report we provide details of the implementation progress; the final state of the implementation together with the software and hardware system requirements needed to run the application; user's guide on how to install, run and play the game, as well as design changes and improvement from the very first iteration.

1.1 Tools used

During the implementation of the system, we used GitHub, IntelliJ and Eclipse. We used the e-mail service provided by the university and WhatsApp to communicate for the designation of ad hoc meetings, whenever necessary. We used IntelliJ because of its robust integration with Git, and we used Eclipse for the rich variety of plugins it provides, such as UI builders. Moreover, for all the previous stages and reports, we used Google Docs to keep track of our tasks and deadlines as well as share the work. In one extreme case, we had to rely on Facebook Messenger for communication.

1.2 Implementation Process

The implementation process began after the submission of the design report for the first iteration. The logical layer of the game was the first to be implemented, and did not see many design changes as they are simply static objects with predefined behavior. The UI implementation soon followed, the first implementation being the skeleton code for panel navigation and the Quadrillion game itself. Until the end of the first iteration, we added placeholder buttons and pictures so that the game looked presentable enough.

The second phase of the implementation began after the submission of the second design report, and the UI framework saw many changes, including custom hand-written listeners, serialization, and theme management. We made some minor changes to the class and object design to make the implementation process easier. We introduced the observer pattern to facilitate the implementation of multi-threaded parts of the project and used message passing interfaces for event handling. Later on, we introduced the profile system to the game along with its GUI.

In the last phase of the implementation we introduced an innovative mechanic to the game, which allows the user to swap out a piece with another one, to make the game easily winnable. We also polished the UI with good looking images, icons, fonts, and component placement.

We were able to implement most of the proposed functional requirements in our analysis report, except for the sound slider and the mute button. We were able to finish a large chunk of the project in time.

2. Work Allocation

Ana Peçini:

- Contribution on Analysis report: Sequence diagram for QGame, QMode, QLadderMode and SettingsMenu.
- State diagrams for QPiece and TreasureMode on Iteration 1 and 2.
- Contribution on Design report: Design goals, Packages, partial contribution to Subsystem Decomposition, Contribution in Object and Class diagram model and Class interface description on Iteration 1 and 2.
- Contribution to the overall content of Final report
- Implementation of the QTreasureModePanel and QLadderModePanel for the GUI.
- Implementation of QMainMenu and navigation between modes and panels.
- Implementation of the UI components such as QShopPanel, QSettingsPanel and QThemeDisplay.

Krisela Skënderi:

- Use Case Scenarios for Iteration 1 and 2.
- Mockups for the Analysis Report Iteration 2.
- Sequence Diagrams for Treasure and Levels Mode.
- Activity Diagram for Game Flow Analysis Report Iteration 2.
- Worked on Subsystem Decomposition and Packages for Design Report.
- For the logic package: Wrote the QMode, QModePanel, QTreasureMode, QLadderMode, QLevelMode, QLevelMode and modified the QPlayer, QProfile

and QAward while developing the logic package classes. Wrote the corresponding descriptions for the Design Report.

- Implemented the Change Piece Functionality/ Power up for the core game (wrote the QChangePiecePane for that).
- GUI Design for Level Mode and the Profile Menu (by modifying the QProfileMenu).

Muhammad Hammad Malik:

- Mockups for the Analysis Report Iteration 1 together with the descriptions.
- Boundary Conditions for the Analysis Report Iteration 1.
- Design Goal Trade-offs for Iteration 1.
- Use case diagrams for the Analysis Report Iteration 1 and 2.

Ünsal Öztürk:

- Implementation and design of the core game logic (Game logic subsystem, “quadrillion” package)
- Implementation of shop logic and barebones shop UI (QShopPanel)
- Implementation and design of factories and builders throughout the system
- Implementation and design of QGazillionPanel and its subcomponents (QUtilityPanel, QPlayerInfoPanel, QPieceCollectionPanel) excluding the change piece powerup
- Implementation and design of interfaces: Observer, Observable, QDiskPersistable
- Implementation and design of message passing interfaces between threads
- Implementation and design of profile objects and barebones interface (QProfile, QProfileMenu)
- Implementation and design of sound loading and management (QSoundLoader)
- Implementation and design of themes, theme managers, theme display UI (QTheme, QThemeManager, QThemeDIsplay)
- Implementation of high-level panel swap interface (QFrame, QPanel)
- Partial design and implementation of QPlayer
- Ladder Mode Sequence Diagram

- Design Report Contributions:
  - Performance design goals
  - Hardware/Software Mapping
  - Persistent Data Management
  - Boundary Conditions
  - Object Design Trade-offs
  - Centralized vs Distributed Responsibility for File Persistence
  - Design Decisions and Patterns
  - Class Interface explanations and text
- Analysis Report Contributions:
  - Nonfunctional requirements
  - Pseudo Requirements
  - System requirements
  - Class diagrams for core game logic, and the diagrams for classes personally implemented
  - State diagrams for QPiece and LadderMode, text for all state diagrams
  - Parts of the text of the report

3. System Requirements

3.1 Software Requirements

The system requires a JRE8u212 installation to operate, since it runs on the JVM. It does not require a JDK installation. Any machine which supports Java Runtime Environment will be able to run the game.

3.2 Hardware Requirements

- Minimum Requirements: 512 MB RAM, Intel i5-2xxx series processor, a mouse, speakers.
- Recommended Requirements: 1GB RAM, Intel i5-2xxx series+ processor, a mouse, speakers.

3.3 Installation Guide

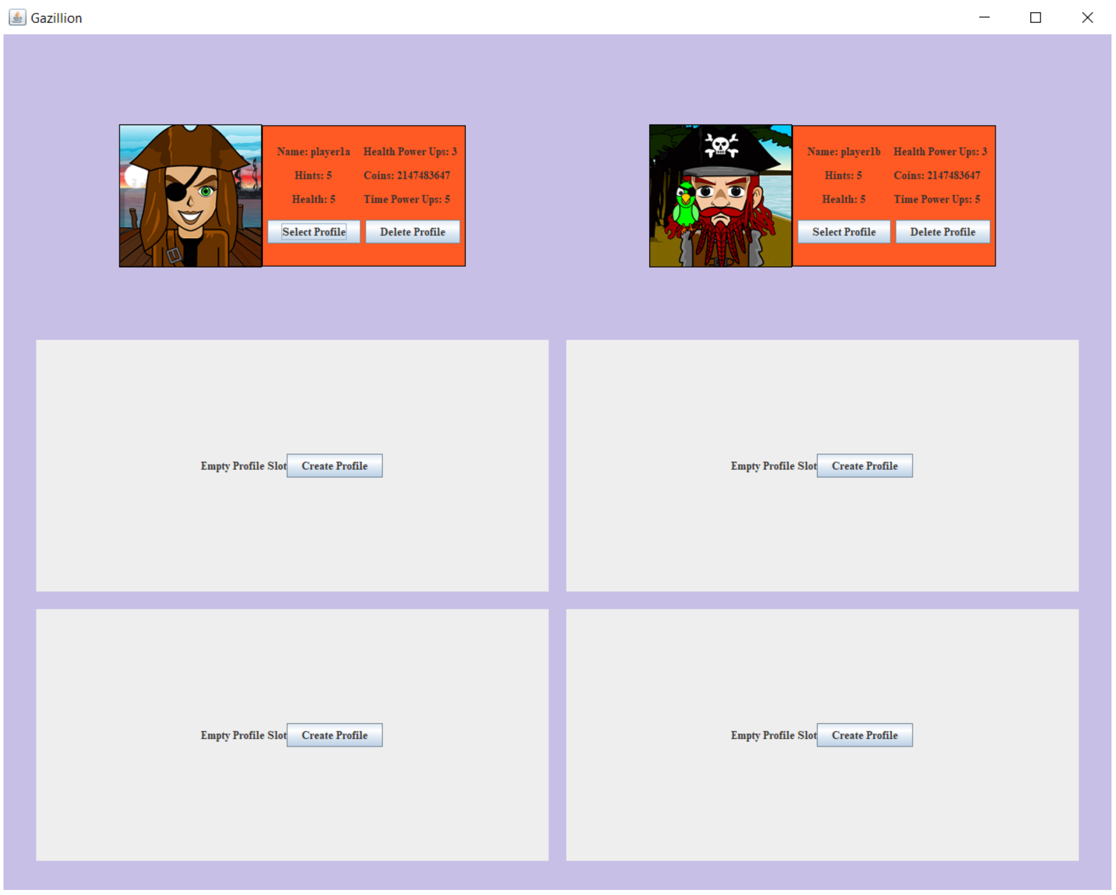
The game can be retrieved from the GitHub repository in the form of a .jar file. The user then may place this .jar file under an arbitrarily named directory (e.g. “Gazillion”) and then the user may run the game by double clicking on the .jar file. Any file that the system has to persist to the local file system on which the .jar file is located will be persisted to a subdirectory called “src” at the same level of the file hierarchy on which the .jar file is located.

4. User Manual

4.1 Launching the Game

The user should double click on the .jar file to launch the game. The game will launch after briefly loading the required assets (icons, images, sounds) from inside the .jar.

4.2 Creating, Selecting, and Deleting Profiles



Name: player1a Health Power Ups: 3
Hints: 5 Coins: 2147483647
Health: 5 Time Power Ups: 5

Name: player1b Health Power Ups: 3
Hints: 5 Coins: 2147483647
Health: 5 Time Power Ups: 5

Figure 1 Profile Menu

When the game launches, the user will be displayed the profile selection screen. In this screen, the user may create, select, or delete profiles. A profile is required to play the game. After the initial installation, the user will not have any profiles and will have to create a profile by clicking on the "create" button inside one of the profile slots. Upon clicking the button, the user is prompted to enter a name for the profile.

The user may then select a profile by pressing the "select" button situated in one of the profile slots. The user is also able to see the general state of the profile (amount of coins and powerups, profile name).

The user may also delete an existing profile by clicking on the "delete" button situated in the profile that the user desires to delete. As a confirmation measure, the user will be prompted to enter the name of the profile that the user wishes to delete. If the profile name was entered correctly, the profile is deleted and removed from the filesystem, forever. Else, the profile will not be deleted.

4.3 Navigating the Main Menu

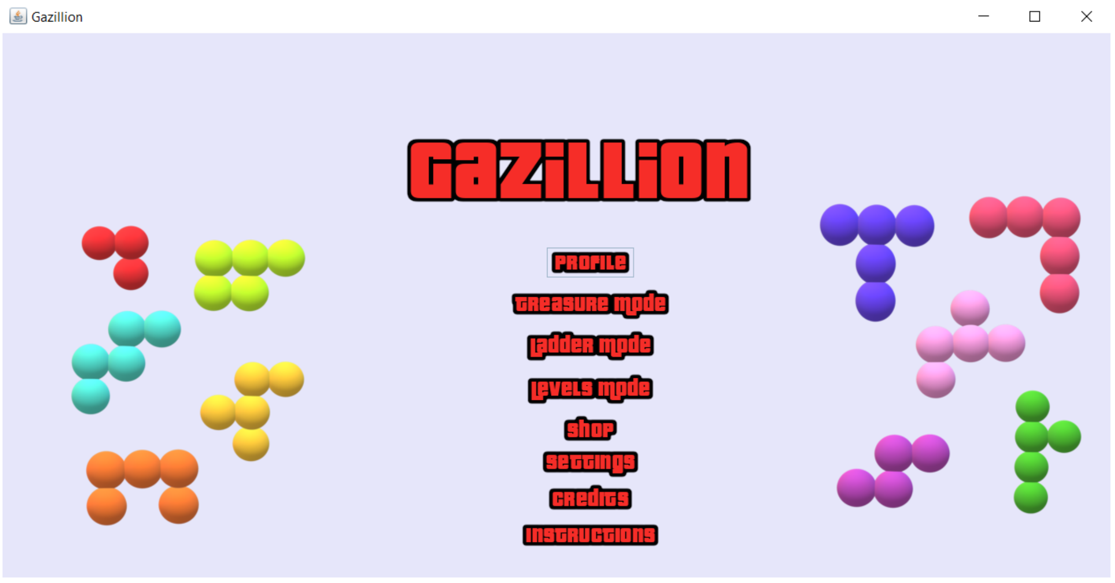
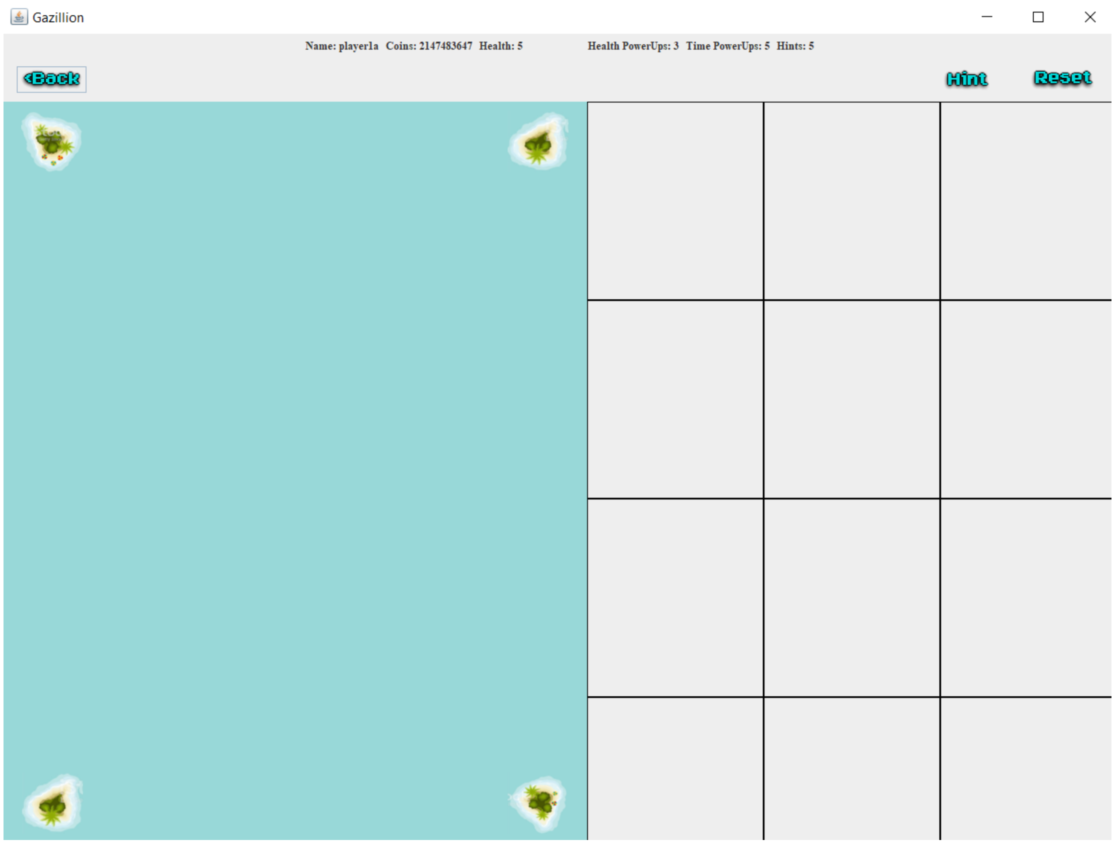


Figure 2 Main Menu

The user will be directed to the main menu after selecting a profile. In the main menu, the user may press on the buttons to further interact with the system. Clicking on the "Treasure Mode" button will redirect the user to the treasure mode, clicking on the "Levels Mode" button will redirect the user to the levels mode, clicking on the "Ladder Mode" button will redirect the user to ladder mode and start the ladder mode gameplay, clicking on “Shop” will redirect the player to the shop, clicking on “Settings” will redirect the player to the settings menu, clicking on “Credits” will redirect the user to the credits menu, clicking on the “Instructions” button will redirect the user to the instructions menu, and clicking on “exit” will terminate the game.

4.4 Playing the Treasure Mode



Name: playerla Coins: 2147483647 Health: 5 Health PowerUps: 3 Time PowerUps: 5 Hints: 5

Figure 3 Treasure Mode

The user will be greeted by a map upon entering the treasure mode. The aim of this mode is to collect all the Quadrillion pieces scattered throughout the map. A level may or may not contain a Quadrillion piece. The user may click on one of the unlocked levels on the map to play that particular level. The user may either win the level or lose the level. The user will lose one health point every time time runs out in the level. The user will be awarded powerups and coins upon winning a level (amounts are determined randomly). The user may use a “Hint” powerup to reveal the location of one of the Quadrillion pieces on the map. The user may also use a

“Health” powerup to increase the remaining health by one. These actions may be performed via their respective buttons.

The user may reset the treasure mode progress at any time by clicking on the “Reset” button. The user will be asked for a confirmation for the resetting of the mode. Upon a reset, the mode will be reverted back to its initial state (only the first four playable levels will be unlocked and Quadrillion pieces will be randomly distributed over the grid again).

4.5 Playing the Levels Mode

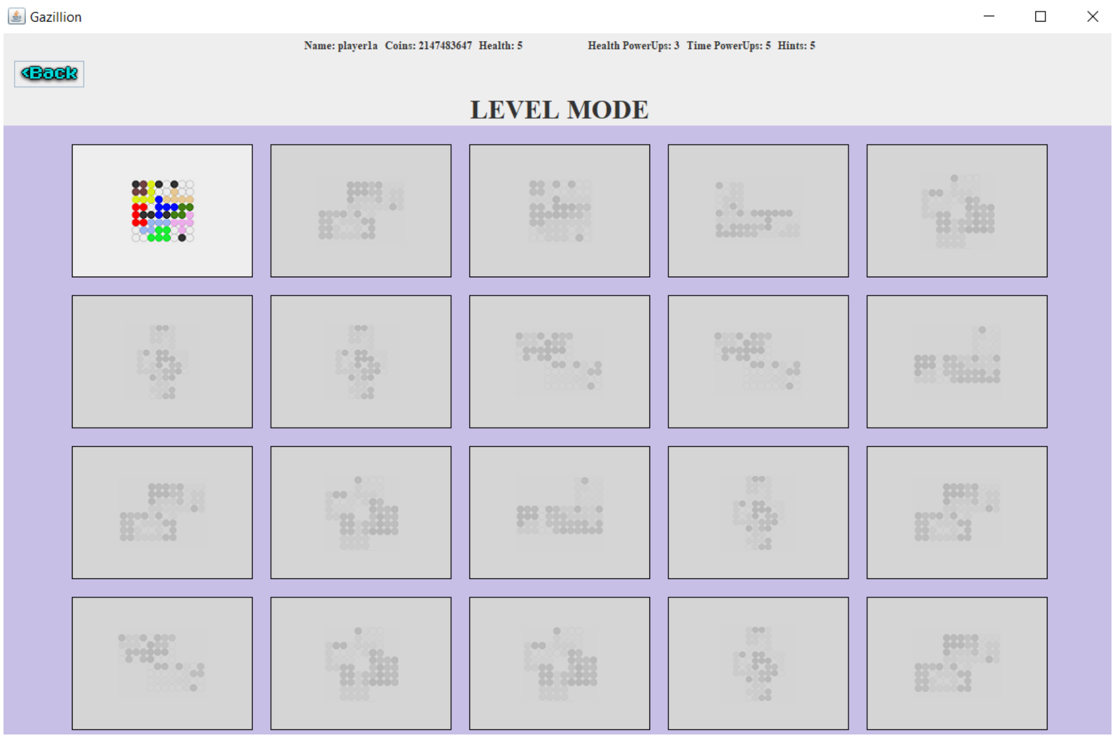
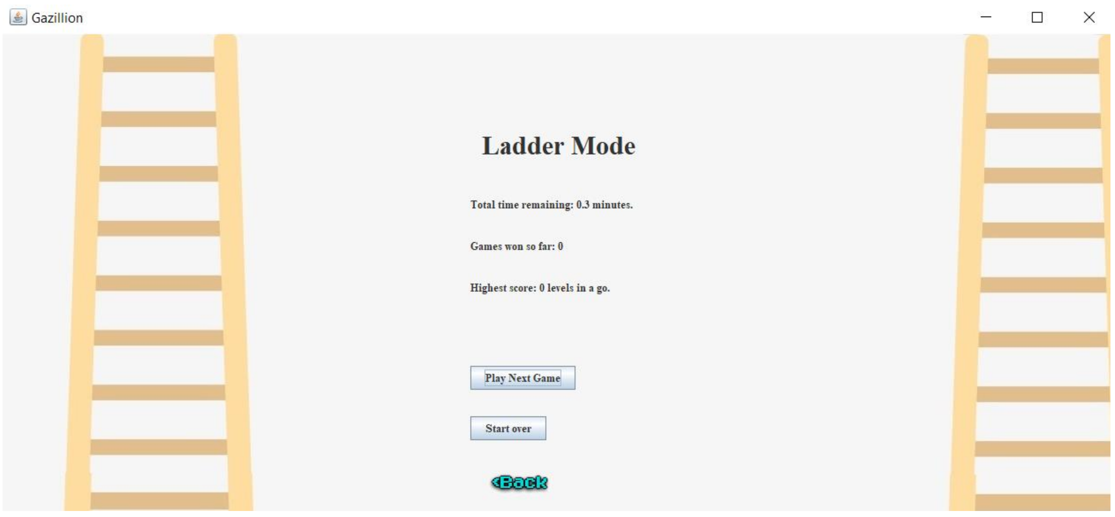

Gazillion
Name: playerla Coins: 2147483647 Health: 5
Health PowerUps: 3 Time PowerUps: 5 Hints: 5
Back
LEVEL MODE

Figure 4 Levels Mode

The user will be greeted by a grid of levels upon entering levels mode. The aim of this mode is to complete pre-defined and simple levels of Quadrillion in a casual manner (no time or health constraints). The user may click on one of the unlocked levels to play that particular level. After the user finishes the level, the next level is unlocked and the player is awarded for the completion of the level. Levels mode cannot be reset. A level yields rewards only the first time it is completed.

4.6 Playing the Ladder Mode



Ladder Mode

Total time remaining: 0.3 minutes.

Games won so far: 0

Highest score: 0 levels in a go.

Figure 5 Ladder Mode

The user will be greeted by a simple information screen upon entering the ladder mode. The aim of this mode is to complete as many games of Quadrillion as possible in a fixed time interval of 20 minutes. The timer starts as soon as the player enters the ladder mode. The user may click on “Play Next Game” to play the next game of Quadrillion. The user will initially have 20 minutes to solve the first level assuming that the user has clicked “Play Next Game” instantly. For the rest of the games the player will have the remaining time as their time constraint.

When the player runs out of time, the player has to restart the ladder mode by clicking on the “Start Over” button. The number of levels that the player wins is set to zero. The user will be awarded progressively higher rewards as the number of levels played consecutively increases.

[This space has been intentionally left blank]

4.7 Navigating the Shop Menu

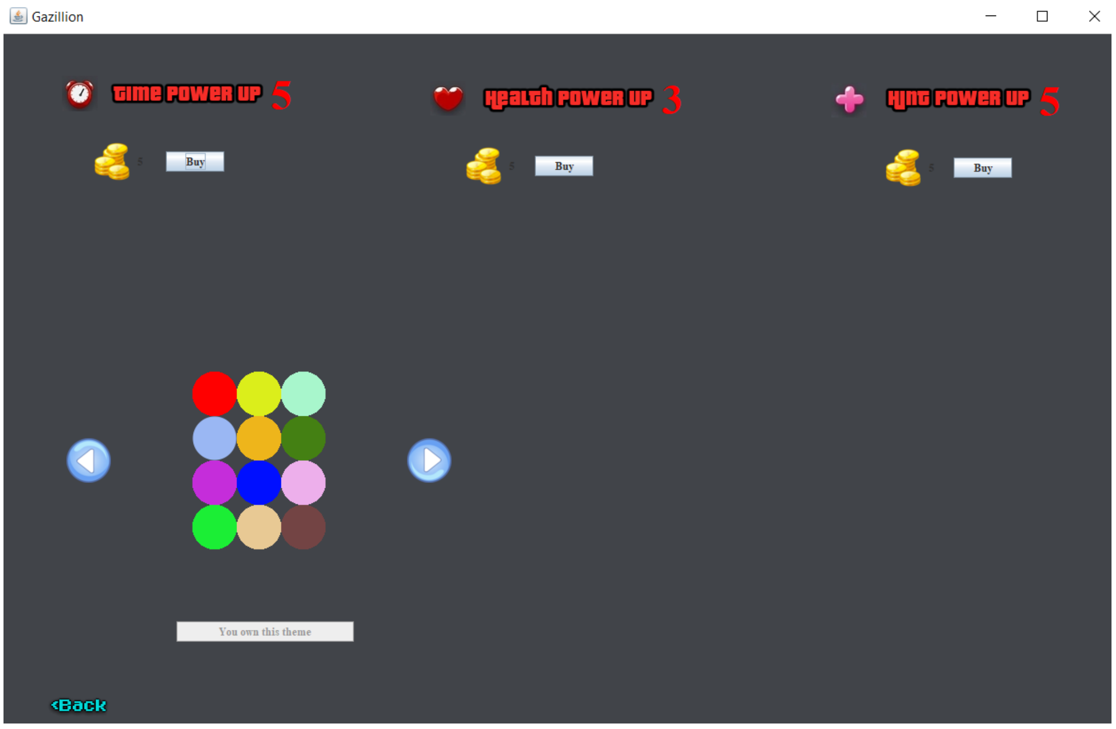


Figure 6 Shop Menu

The user will be greeted by a shop screen with three buttons, labels that provide information about the number of coins the player has and a carousel-like menu from which the user can select a theme. The player may purchase the powerups by pressing the relative buttons. The player can also view themes available for purchase and purchase the currently viewed theme.

[This space has been intentionally left blank]

4.8 Navigating the Settings Menu

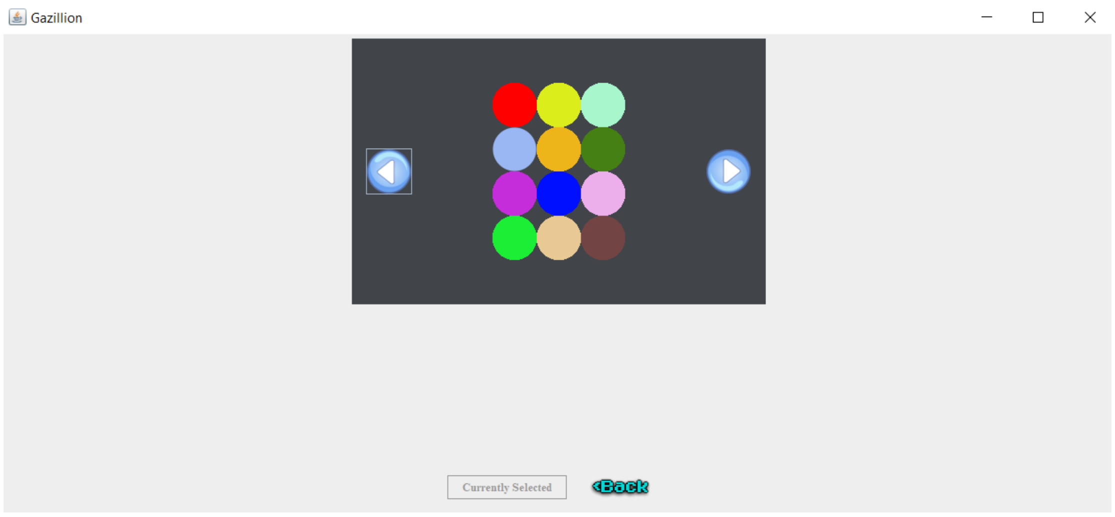


Figure 7 Settings Menu

The settings menu contains the same carousel-like menu, and next to it there is a button. Upon clicking the button, the currently viewed theme will be set as the current theme for the player.

4.9 Navigating the Credits Menu

The user can see the developers in this menu. It is a static image, and a back button is available to go back to the main menu.

4.10 Navigating the Instructions Menu

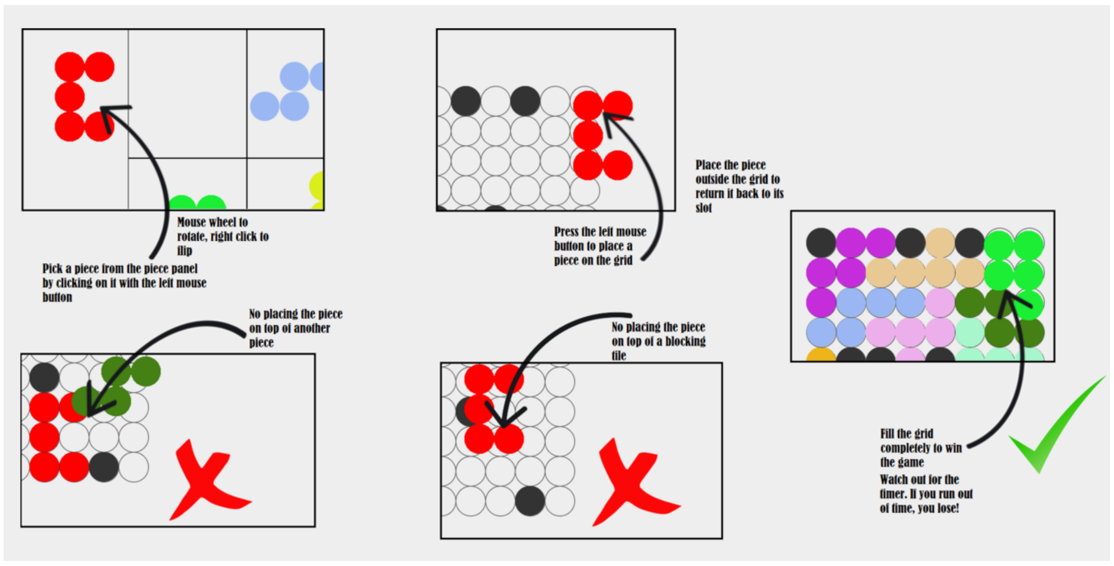

Mouse wheel to rotate, right click to flip
Pick a piece from the piece panel by clicking on it with the left mouse button
Press the left mouse button to place a piece on the grid
Place the piece outside the grid to return it back to its slot
No placing the piece on top of another piece
No placing the piece on top of a blocking tile
Fill the grid completely to win the game
Watch out for the timer. If you run out of time, you lose!

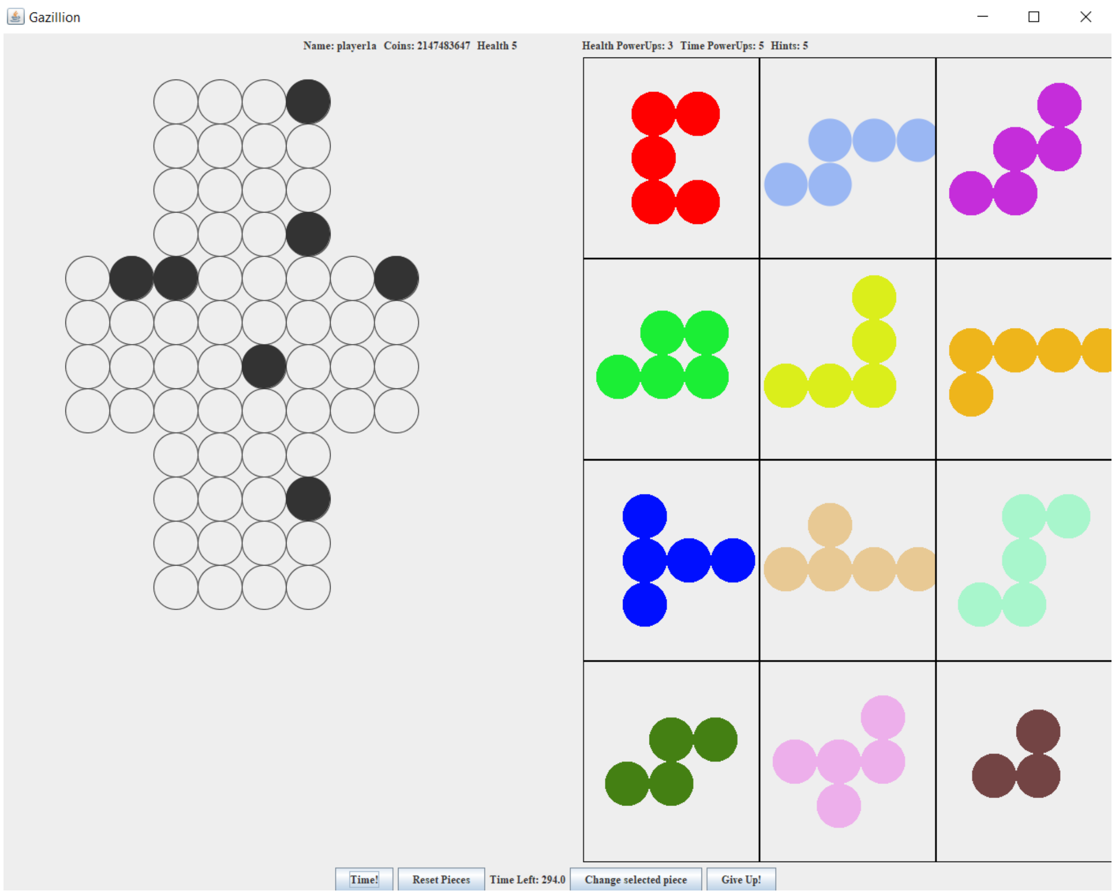
Figure 8 Instructions

The user can see the instructions on how to play the game in this menu. It is a static image, and a back button is available to go back to the main menu.

4.11 Exiting the Game

The player may terminate the game by pressing the cross on the top right corner of the game.

4.12 Playing a Game of Quadrillion



Name: player1a Coins: 2147483647 Health 5

Health PowerUps: 3 Time PowerUps: 5 Hints: 5

Time Left: 294.0

Figure 9 Game Panel

Upon clicking on a treasure mode level, a levels mode level, or a ladder mode level, the user will be greeted by the Gazillion Panel. To the right of the panel the user may select a piece to be placed on the board and then may place the piece on the board by clicking on the board with the piece selected. The piece will be placed on the grid according to the point it is anchored to the mouse pointer. The user may rotate the piece counterclockwise and clockwise by rotating the mouse wheel up or down for the selected piece. The user may flip the piece with respect to y-axis by pressing the right mouse button. The user may not place a piece if there is a piece blocking the piece or if there is a blocking tile over the piece. The piece is returned back to its slot if the user attempts to place the piece outside of the grid. At the bottom of the panel, there is a utility panel from which the user can increase the remaining time for the level, give up, or use a unique powerup once in a game. The unique power up allows the user to switch one piece for another once per game, for free. To access this panel, the user must select a piece, and then click on the button. Upon clicking the button, the user will be displayed another panel from which the user can select the piece that the user wants to swap with the previously selected piece. The user can view their stats on top of the usual panel (name, coins, health, number of relevant powerups). The user is notified when the user wins the game, runs out of time, or gives up. All of these cases have their corresponding popups.

5. Design Changes and Improvements

The core gameplay and the design for the final implementation is mostly the same. There was only some minor refactoring from the original design.

5.1 QProfile and QProfileMenu

The major changes that we had to introduce in the implementation are the implementations of the classes QProfile and QProfileMenu, with the purpose of proper and transactional serialization of player progress. In our implementation of serialization, we used the observer, prototype, and memento patterns. The observer pattern was used to make the QProfile object observe the changes in the QPlayer object it’s representing so that whenever there is a change to QPlayer, the QProfile should also be changed as a solution domain constraint. The prototype pattern was used to reduce the number of disk accesses in the case of multiple profile changes during a single session. The profiles read from the disk are encapsulated in the main memory so that whenever the user requests to change a profile, the profile is not read from the disk but instead copied from the prototype object. To ensure fault tolerance, transactional integrity, and software stability, we used the memento pattern to save the initial state of the QProfile (and hence the QPlayer) object, so that if there is a problem persisting files to the disk, the changes are rolled back and reexecuted.

The QProfileMenu simply provides a GUI for profile selection and management.

5.2 Piece Change Ability Implementation

Throughout the tests we ran for the game, we realized that it was difficult to actually solve a level of Quadrillion from scratch. It turned out that it is even more difficult when one encounters a random level, which is arbitrarily difficult. In most of the cases, we were able to reduce a level of Quadrillion to such a state that we would be able to complete the level if the last remaining piece was another type of piece. This was quite common and occurred for almost 80% of the time in levels we've played during the testing phase. Therefore, we decided to implement this as a feature. We allowed the player to swap a piece with another piece only once, for free, per game.

This implementation made it actually possible to win random levels of Quadrillion in a reasonable amount of time. The average time it took for us to solve a level of Quadrillion with this upgraded functionality was four minutes after learning how to play the game for at least an hour.

5.3 Missing Features

The only missing features in the game are sound control features: The system is capable of playing sounds and music; the system can also stop playing these sounds on command. However, the system cannot mute or reduce the amplitude of these sounds. Due to time and manpower constraints, and also due to the "high effort, low reward" nature of this feature, we decided to leave it out.

6. Conclusion

Generally speaking, as a group composed of four people, we believe that we've done what we could for the final implementation. We tried to forward engineer every single aspect of the product and tried to stick to the common design patterns and applied them wherever possible.

We think that we were able to learn much about software design and working as a group in this project, and we are confident that we will be able to apply the skills we acquired in our new projects.