Bilkent University

Department of Computer Science

CS 319 Object Oriented Programming

Project Design Report

Iteration - I:

QuadZillion

Supervisor: Eray Tüzün

Group 1G

- Berk Gürler
- Enver Yiğitler
- Kasım Sarp Ataş
- Melike Arslan
- Ufuk Bombar

Table of Contents

1. Introduction

Purpose of the System

QuadZillion is a 2-D solitaire puzzle game. Its design is implemented in such ways. The motivation behind developing a digital version of a game that already exists in the form of board game is to have an alternative with an appealing and user-friendly interface available on computers. Players can create their custom levels and save them to play later. QuadZillion encourages critical thinking and requires players to come up with creative ways to solve the given puzzles. Further, players can challenge themselves or their friends. They can compare their achievements using the scoreboard feature of QuadZillion.

Design Goals

Design is one of the most important steps to create a game since what is designed will be the main focus of the game. In general, there are various number of highly desirable qualities, however they will be described in more detail under the following main criteria's.

End User Criteria

End user criteria is inferred from the application domain. It includes the user's point of view of using the system. Since our system is a game, the fact that whether the system supports the work of the user is not relevant.

We expect the user to understand and be able to use the system very easily since it is a game system. The user will be provided with a selection of choices on every situation and immediate response or display is given back to the user. There is even an explanation on how to play the game, hence the user is expected to be able to use the system easily.

Performance Criteria

Performance criteria is inferred from the application domain. It includes space and time complexities and space and time requirements. The system is responsive meaning that it is an interactive system letting the user interact with it and be provided with some results depending on the interactions. Memory space will be available for speed optimizations. The response to the user will be instant. When the user requests something, the system will provide an output instantly. Currently the space required for the system to run is unknown, it will be become clear after the implementation of the game is done.

Functionality Criteria

Dependability criteria is another criterion inferred from the application domain. In this criteria the effort to minimize the system crashes and their results will be determined. However, we cannot know how often the system will crash as of right now.

Maintenance Criteria

Maintenance criteria is dictated by the customer and the supplier. Maintenance is an important factor of a game after design and deployment. QuadZillion will not be using any servers or multiplayer options therefore we will not be focusing on updating the software with newly emerging technologies. However, we will release new patches of the game to solve the bugs which might emerge after the release.

Another aspect for maintenance is the features we will add after the initial release. If we want to add a new feature to the system after the release, then we should maintain the system for achieving this goal.

Cost Criteria

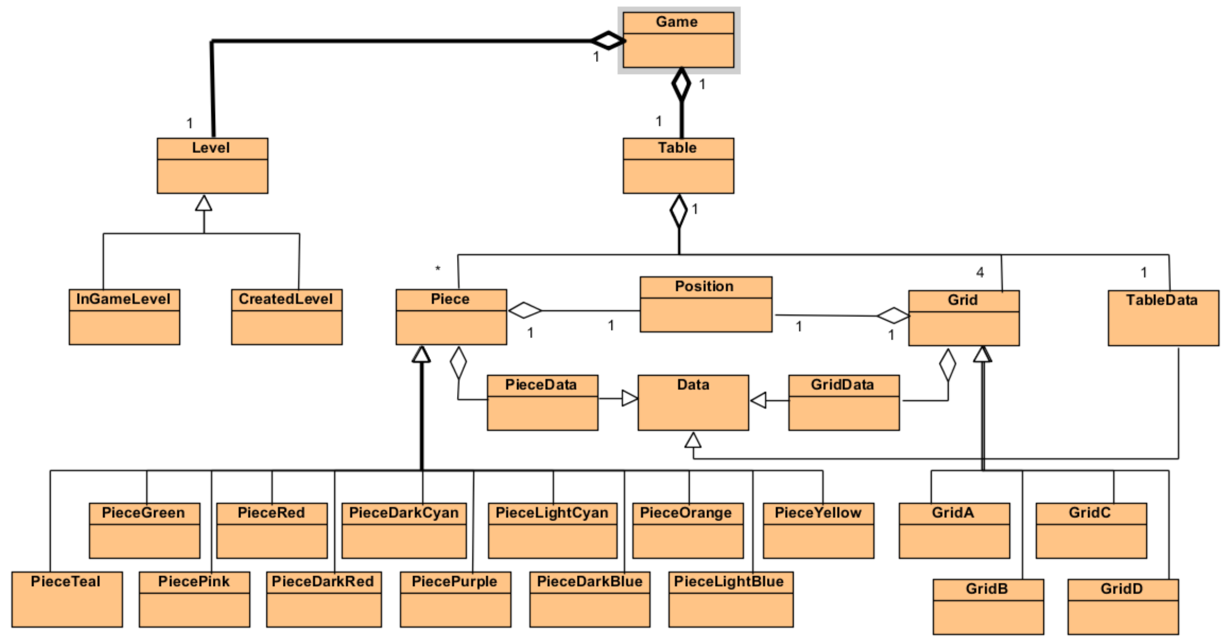
Cost criteria is also dictated by the customer and the supplier. It includes the cost to develop, deploy and administer a system. For our system, the cost criteria is not a major aspect for this system since it is a game system for a course term project.

Programming Language

Since Java supports the “Object Oriented Programming” programming paradigm, it makes a great programming language to games. One of the important aspects for games is modern and appealing looking user-interfaces, this role is fulfilled by the built-in GUI Library of the Java, namely JavaFX. Its’ integration with several IDEs such as Eclipse and IntelliJ makes it convenient to develop GUIs. Another great advantage Java has is the Java Virtual Machine(JVM) and the portability it brings with it. The game can run on any operating system and platform as long as JVM is installed on the target computer.

2. Software Architecture

Overview



The object class diagram summarizes the general software architecture of our program.

Subsystem Decomposition

We separated the models and the views into packages. The ‘main package’ contains the core game classes which requires for game to operate. The ‘gui package’ on the other hand is requires for us to present the game to user visually.

The model-view-controller (MVC) pattern become the main development pattern in our project. This is because the rate of interaction between the user and the program is high.

Hardware/Software Mapping

QuadZillion game will be implemented in Java environment, and use JAVAFX libraries for GUI purposes. That’s why QuadZillion will require the Java Runtime Environment in order to execute the corresponding code, QuadZillion also require at least the Java Development Kit 8 or any other new versions of the Java Development Kit since JAVAFX libraries include Java Development Kit.

QuadZillion will require the keyboard as a hardware requirement, since user interact with the game with using keyboard (This will be explained in Boundary

Conditions). Keyboard hardware and specified java programs will be enough to run QuadZillion.

Storage of the game, maps, score list, user preferred settings will be containing in the files. Therefore “QuadZillion” will not require any database nor internet connection.

Persistent Data Management

QuadZillion does not require any complex data storage system, since it stores its maps option selections in files it does not uses any sort of databases. Scoreboard holds scores in text files (saving purpose) that’s why QuadZillion does not require much saving space for users to change the game’s initial optimization. We will use themes and pieces as PNG format.

Access Control and Security

QuadZillion is a single player game, so it does not use any internet connection. nor database. After loading and starting the game user enters an isolated space since there are no internet connection nor database usage. Since QuadZillion will have an .jar executable, anybody how has computer knowledge can access the source codes of the game, and modify the codes as s/he desired. Therefor there is no “Security” nor “Access Control” units.

Boundary Conditions

QuadZillion will have an executable .jar. It can be converted to an executable .exe with the help of JSmooth. Since creating executable jar file makes game easier transfer to another device, it’s become easier to share the game. QuadZillion can be terminated by clicking the “QUIT” button. In gameplay user can open pause menu by clicking the pause symbol. In order to close the pause menu in the game, user can press “Esc” button. If user presses the “Return to Main Menu” button, user will be redirected to main menu of QuadZillion. User can disable the music and SFX by pressing corresponding buttons. User will use the “A” and “D” buttons on the keyboard in order to rotate pieces counter-clock wise and clock-wise respectively. User will also use the “Space” button on the keyboard in order to rotate pieces horizontally. If game collapses before level finished and user returned to the main menu the score of the user get in the stage will be lost and cannot be reached in scoreboard.

Initialization

When the game is opened the constrains are initialized, these includes the option files are user created save-games.

Termination

If user closes the game or game crashes without finishing the game user will lose all the progresses in that specific level.

Failure

When user try open game and any one of the required files are corrupted game cannot be opened. Player can’t open game if java environment is not installed, or cannot access the level if some files are corrupted inside the corresponding file.

3. Subsystem Services

Interface Service

User interface handled by the ‘gui package’. This package contains the files required to create the proper user interface. These interface files are coded in a language very similar to xml but named FXML since the parser parses them by JavaFX standards.

Interactions between the user and the program kept simple, we acknowledged the minimalistic and materialistic approach towards the user interface. Minimalistic approach states the more compact the better and the materialistic approach enables us to create interfaces which attracts the user and looks clean. These design disciplines combined gives us the user interface which looks simplistic and well designed.

Game Provider Service

Core game mechanics will be separated by view and controller classes. Core game will be used like a API that takes input and produces outputs. This separation provides less coupling and more cohesion for the subsystems. The connection between core game and boundary classes will be done via controller classes as suitable for MVC design pattern.

File Service

File service is an essential part for our program. Our program has mechanics which requires file input and output. For example, the mechanic of saved creations enables user to create new levels. This additional feature forces our program to have a file system because when user opens up a new program he/she needs to see the old saved file.

Another use case is for the options file. In that file the options are stored, when the program initiated first thing to do is to read the option files and save them to the memory. For that purpose, the file service should both have functionality over different file types and both input and output.

4. Low-Level Design

Overview

The main problem of the design is to implement the core game. Game board and pieces are represented by matrices. This choose of design provides simplicity for implementation of game rules and moves due to nature of the game is matrix-like. GUI is handled by the JavaFX library. JavaFX provides useful tools for animation and controlling visual objects. Game board and pieces are visualized by the JavaFX library. Canvas object is used for rendering. For separation of models and views, renderer interface is constructed and used.

Trade Offs

On the design stage, we have settled for two different implementation ideas. They both resulted as different space and memory complexities. In this chapter we will discuss how these designs differ from each other and how do they perform on different tasks.

Memory vs Performance

Game's program size and amount of computation is very low and can be handled by every modern PCs. Game will be run at 60 fps and memory requirements will be less than 1MB. Game is very small and can run at any personel computer without problem..

Simplicity vs Number of Options

We acknowledged minimalism as our design principle in user interface as well as system design. Simplicity and clear design is the goal we want to achieve in this program. In the

5. Conclusion

Our main goal with this project is to use the advantages that OOP paradigm provides with a programming language that supports OOP, such as Java, to develop our game. These advantages can be summed as follows: Easier maintenance, faster development, modular structure and reuse of code when necessary. We are aware of good design being a crucial part of every software project(except trivial ones), so we analyzed our game and came up with implementation ideas and discussed them briefly in our design project.

6. Glossary and References

- OOP: Object Oriented Programming
- JVM: Java Virtual Machine
- JavaFX: Built-in GUI Library of Java
- MVC: Model View Controller Pattern

1. https://openjfx.io/javadoc/11/
2. https://www.smartgames.eu/uk/one-player-games/quadrillion