SportZ

Design Report on Software Maintainability

Version 1.0

By:
Chia Songcheng
Chew Poshi
Wong Jing Lin Fabian
Lim ShengZhe
Lin Zixing
Hermes Lim
Chee Zi Hoe

Document Change Record

Revision	Description of Change	Approved by	Date
0.1	Initial template	Chew Poshi	23/3/22
0.2	Add Design Strategies, Architectural Design Patterns	Chia Songcheng	27/3/22
0.3	Added Software Configuration Management Tools and figure for Architectural Design Patterns	Fabian Wong	30/3/22
1.0	Revised Edition	Lim ShengZhe	30/3/22

Contents

Contents

1. Design Strategies

1.1. The Planning Phase Before Development

From the beginning, we made analysis and prediction of future interesting game features that could be implemented after the release of the game application. Once the game launched, new additions of features and gameplay are mandatory to keep players enticed with the game. Game features such as new maps, character skins, etc. Hence, we targeted new product features as one of the important factors to consider for the foreseeable future.

Since SportZ is a single-player game that is mainly dependent on user experience, we decided to adapt the Model-View-Control(MVC) model as our architectural mode. This system infrastructure framework will be beneficial for our software because the game is essentially divided into Graphic User Interface (GUI) and game logic controller. Any updates can be done easily on the individual components without affecting the others.

1.2. The Process of Developing

We are conducting the test in a small and test driven development. To ensure that our game is user-friendly for most people, our targeted audience consists of a wide range of people with ages ranging from 13-45. From teenagers to young adults and working adults.

Although the safe distancing measures that were previously put in place to cope with the COVID-19 situation are mostly lifted, we still try to minimize contact to prevent unwanted spreading of cases. Hence, we are unable to get a large portion of the targeted users to play-test our game in person. Instead, online discrete game testing was arranged to provide continuous feedback on the design and usability of the game. Furthermore, team members' feedback are also taken into consideration.

1.3. Correction by Nature

Corrections will be made to the software during testing. These are the faults that we pay attention to:

1.3.1. Corrective Maintainability

Fault detection done through testing.

1.3.2. Unplanned Corrective Maintenance

Fault detection done through users reporting bugs.

1.3.3. Preventive Maintainability

Features are implemented in components to achieve less coupling, e.g. player controller, enemy controller, etc. The object oriented programming nature also provides assistance to manipulate, enable or disable individual components easily for troubleshooting.

1.4. Enhancement by Nature

We will enhance our application while testing the application. And this is what we will look out for:

1.4.1. Adaptive Maintainability

Able to take in new features without major bugs

1.4.2. Perfective Maintainability

After product delivery, quick error detection and correction allows for reduction of maintenance cost and time required.

1.5. Maintainability Practices

To uphold quality in both process and product, we have implemented the following maintainability practices over the course of our project:

- Readable Code
- Version Control
- Standardized Documentation
- Modularity
- Decoupling

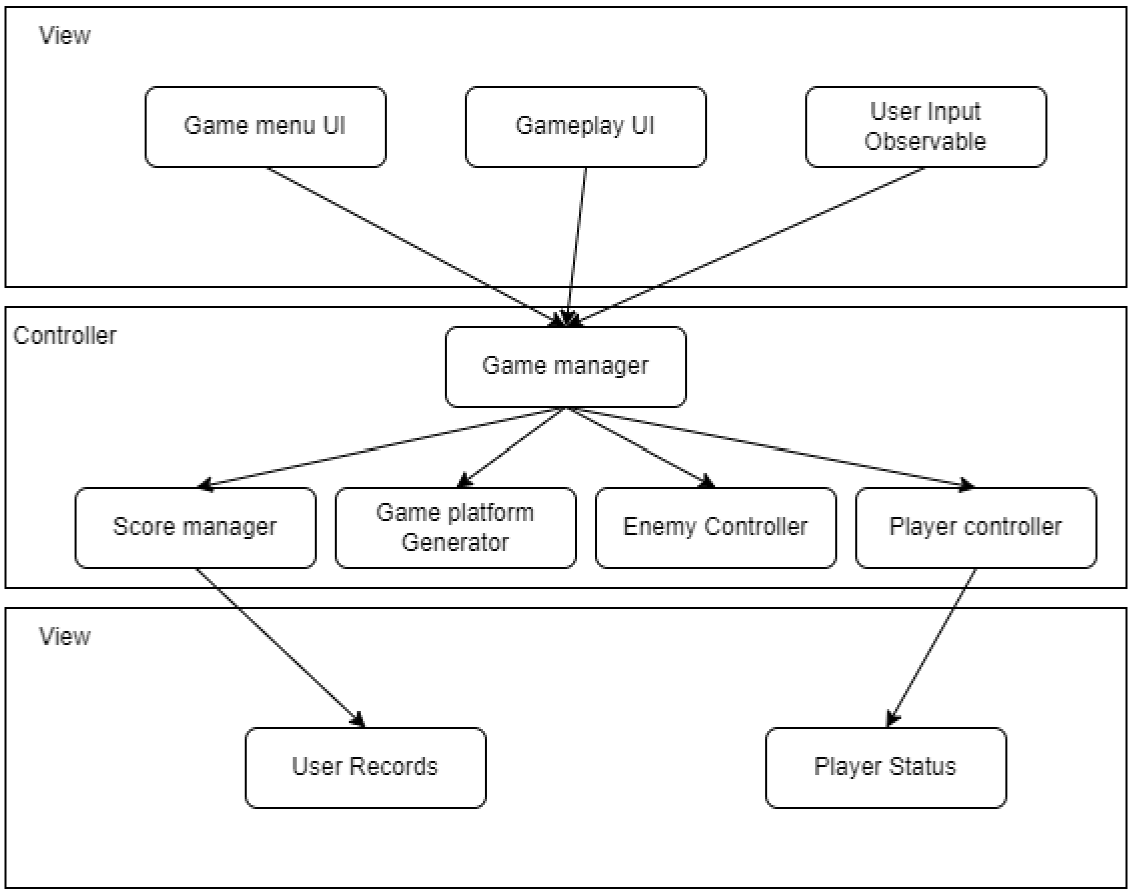
2. Architectural Design Patterns

SportZ uses the Model-View-Controller (MVC) architectural design pattern.

The Model layer holds in-game player status - health points, coin collected, current score and character themes as well as user data - saved user settings and historical high-score which are crucial data required for the gameplay.

The View layer contains the component used for displaying gameplay UI and receiving user input for in-game character jump or attack. Furthermore, buttons from the game menu, setting menu and the main game scene are also stored in the view layer.

Lastly, the Controller layer is made up of many individual controllers, e.g. playerController, enemyController, platform Generator, etc. Each controller is responsible for a part of the gameplay. Taking platform Generator as an example, it generates platforms of random length and positions them at random locations.



3. Software Configuration Management Tools

This is where we will discuss version control management, and tracking on who made what changes and when.

3.1. MediaWiki

MediaWiki is a free and open source application. It is a lightweight tool for hosting various document related software such as Change Request Forms, Logs and other necessary documentation. For anything non-source code related, MediaWiki provides an easy to use common platform for all members of the development team to access the latest information or baseline documentations to ensure that changes are properly tracked and documented.

3.2. GitHub

GitHub is a reliable and easy to use hosting platform for version control. We chose GitHub for its user-friendliness and the variety of features it offers. One example is the branching feature, it allows us to work on different parts of the game without interfering with one another. We also had a close call when the game was corrupted in the main branch and we managed to salvage the game by reverting back to a previous commit through commit history.

3.3. Google Drive

Google Drive offers quick and easy sharing of files and most importantly online editing documents like Google Docs and Google Slides. Google Drive also comes in handy for data and file exchange between the technical team and the design team.