TDT4120
Software Architecture

Architectural Description Document

Group X2

Milos Zlatkovic
Dimitry Kongevold
Emil Grunt
Nicolay Thafvelin
Julius Buset Asplin

Contents

1 Introduction

This document contains the architectural decisions related to the quality attributes we have given in the requirements document. We will describe the architectural drivers, stakeholders, architectural views, quality tactics, and the architectural patterns we have chosen.

2 Architectural drivers

The biggest Architectural Driver in this current project is lack of time. This will affect your architecture to make it more compact and easier to implement.

2.1 Modify-ability

We need to have full Modify-ability regarding new characters, stages, moves and etc. This means that we want to be able to change and add characters, stages and moves and etc without having to change parts of the already written code and without having to concerned about how we write the new code.

Because of the multiple people are working on this project, the modifiability is also and Architectural Driver, the implementation should be separable so we could preside the implementation in a parallel pattern. This also a product of previous driver, short time.

2.2 Testability

We need to be able to test the game all the way through the production so that we can make the right touches to the game so it will be good looking, fun and smooth to play. It is also very crucial for balancing of the characters so that even though they have different moves, they are equally good.

2.3 Performance

The code must be efficient enough so that there wont be any lag or delay while the game is played. We must optimise the calculations and logic and not do more operations than necessary.

2.4 Portability

The game should be playable on both Xbox 360 and computer.

2.5 Usability

The game should be very easy both to start and to play. Their should be easy to find help on which button does what and so on. The objective of the game is quite obvious and therefor needs no introduction.

3 Stakeholders

3.1 Developers

- Want a fun playable game as return for their time-investment.
- Need a good grade on the project, and therefor a good architectural structure.

3.2 The professor

- Wants us to learn about and test/use his advises on architectural structures and program design.
- Wants us to get a good grade.

3.3 End user

- Wants an easy learned, easy played and fun game.

The main concern is the balance between the number of features and the actual playability of the game. This is also the reason why we want it easy to modify. We will make the standards and the basics of the game finished before adding all the moves and features. We will make it so that adding moves and characters almost only need parameter attributes and pictures to be ready for use.

4 Selection of Architectural Viewpoint

4.1 Logical view

Logical view shows the Model View Controller architectural pattern. where classes have different functionality. Based on the pattern. the target audience is other architects and stakeholders.

4.2 Development view

The development view describes the static organization of the software in its development environment. You would usually present this in a diagram that presents the different modules/layers of the system.

4.3 Scenario view

Scenario view gives a better understanding over the run time processes and their communications with each other during well known actions in the game. As well as the sequence of the MainGame class and other classes. Target audience is stakeholders as implementation team.

4.4 State machine

The last view is a state machine of the MainGame Window, it explains how the user will navigate himself through the game. Audience is stakeholders.

5 Architectural Tactics

5.1 Modifiability

To make this project easy to modify we are planning to use Model View Controller architectural pattern, and implement the characters and their moves with only parameter attributes. This makes it easy to change/add characters, maps or power ups. Model View Controller will also provide grouped implementation, as Character-Controller will only affect the character model. as well as all needed resources needed for Draw method will be allocated under model. were its easy to check for consistences.

5.2 Testability

We are going to start out very simple, so that the testing, as in playing the game, can start early. Since we are doing it this way, we wont need the computer to perform actual tests for the most part. We can test it simply by building the project and start testing it manually.

To test the individual methods and the logic itself we will try to separate logic from data and GUI, which will make testing of them easier.

5.3 Performance

We will do performance testing to weed out resource hogs, and find out which methods use the most computational power and alter or remove them.

5.4 Portability

We will only use classes which are supported by Xbox 360, and surround any platform specific code with compiler pragma to avoid duplication of code.

5.5 Usability

This has nothing to do with the architecture.

6 Architectural Patterns

These are the architectural patterns we're planing to use

6.1 MVC - Model, View and Controller

We chose Model View Controller architectural pattern because it gave us the best modifiability as well as it separates code into small fragments that are easy to test and distribute the work load through the group.

6.2 SOA - Service Oriented Architecture

The Game class in the XNA Framework provides an infrastructure for registering and providing application level services. By grouping related functionality into classes and offering them as services specified through an interface, the functionality can be provided without compromising modifiability.

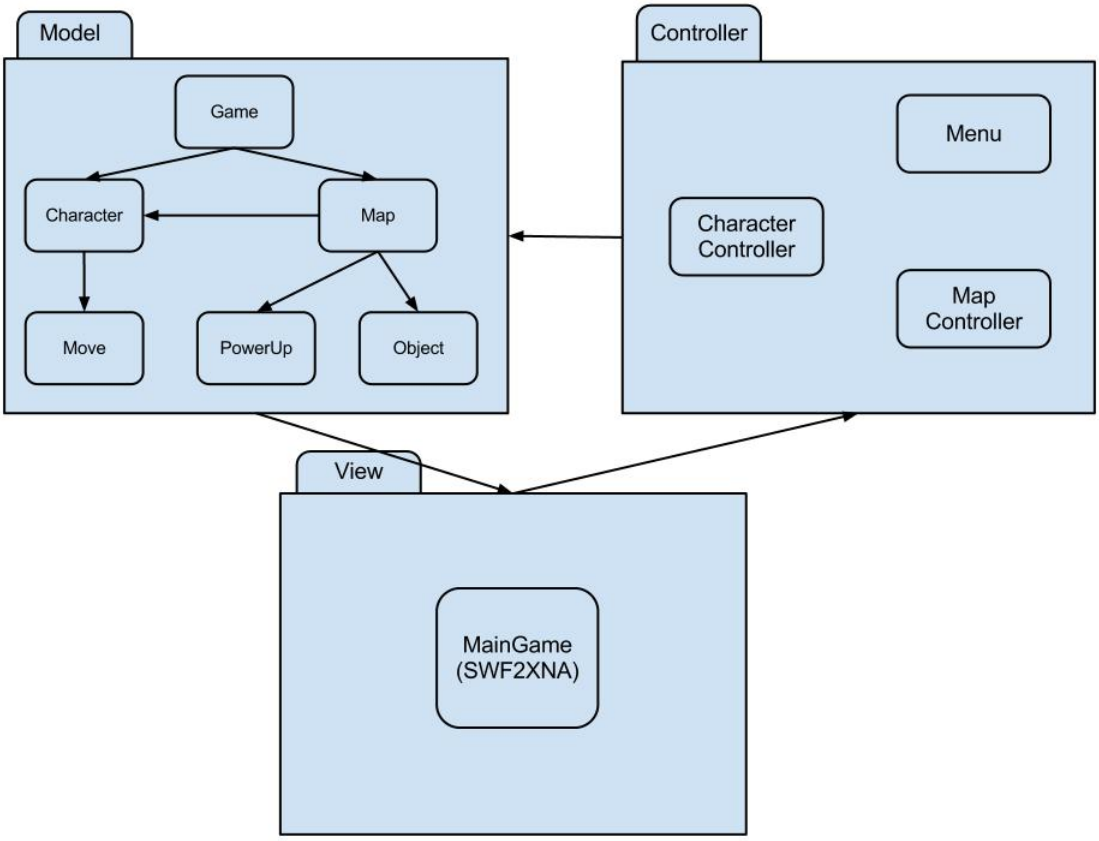
6.3 Observer pattern

The communication within the game will primarily be through firing of events and registration of callbacks. This will improve the modifiability a lot. We also will use state pattern in the characters and the map class.

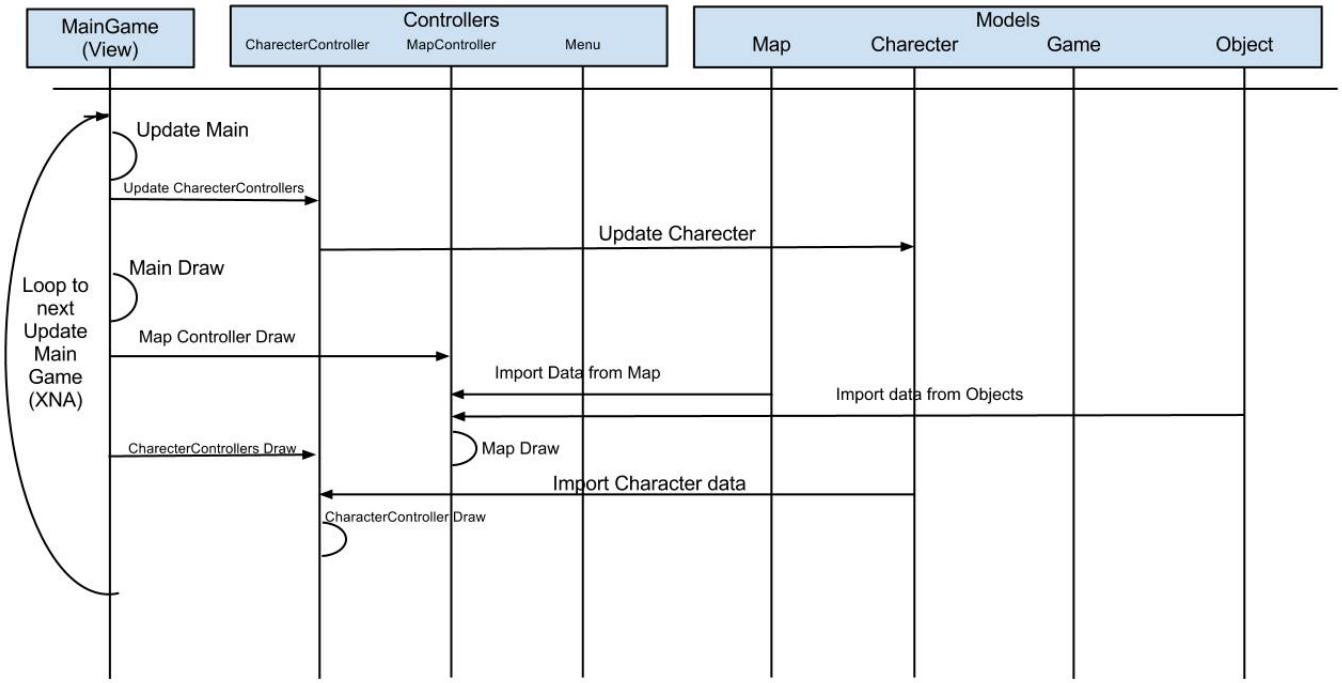
Views

7

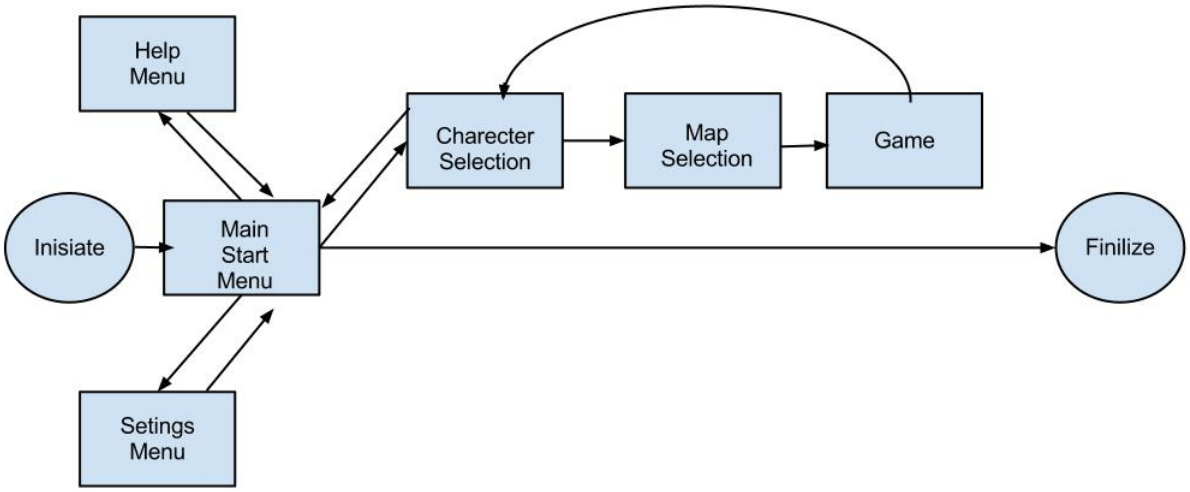
7.1 Logical Views



7.2 Scenario view



7.3 State machine



8 References

This is the references we used to complete this document

8.1 Books

- "Software Architecture in Practice, Second Edition", Len Bass, Paul Clements, Rick Kazman, Addision-Wesley, 2003, ISBN 0-321-15495-9
- "Game Architecture and Design - A New Edition", Andrew Rollings and Dave Morris

8.2 Webpages

- http://en.wikipedia.org/wiki/SOA
- http://en.wikipedia.org/wiki/Modelviewcontroller
- http://en.wikipedia.org/wiki/Observer_pattern