Table of contents

1 Introduction

This document will go into the detail explaining how “The Cave of Caerbannog” is put together, what one would need to be able to run it, and what the main design goals behind the game are.

1.1 Design goals

In this chapter the main design goals for the final release will be discussed, which as a result will have to be maintained throughout the project.

Manageability
As the game can change during development, or even after release, one of the design goals is manageability. To make the code more manageable it will be split in independent parts, which will work together to make the game run. All of these parts should be able to be exchanged for other correct parts which implement the same interface without the game breaking.

Quality
The code must be guaranteed to be of high quality. This guarantee can be made because of the following agreements between members of the development team

- All code has to be tested to reach a line coverage of at least 80%. If this task is impossible an explanation will have to be given as to why this is the case.
- Git and Github will be used for all code related to the project.
- Pull requests will have to be reviewed and accepted by at least two members of the development team before they can be merged.
- All methods and classes have to include javadoc explaining their functionality.
- The code may not contain warnings generated by checkstyle/FindBugs/PMD. If this task is impossible an explanation will have to be given as to why this is the case.
- The program will be built by a continuous integration system after every push to the online branch. Pull requests that do not build will not be accepted.

Reliability
The game must remain playable and not crash under normal circumstances. This goal is reached by extensive play testing and the aforementioned guarantee that the code must be of high quality.

2 Software architecture views

In this chapter the architecture of the game will be discussed. This will be done by decomposing the project into its independ subsystems, explaining which hardware is used and how it is used, and how concurrency problems have been dealt with.

2.1 Subsystem decomposition

The system is divided up into different subsystems according to the MVC pattern. This adds to our design goal of “manageability”. One of the reasons the MVC pattern is used is because multiple views will be presented on the same game state.

Model
This subsystem consist of the jMonkey engine, as required by the stakeholders. More code can be added to this subsystem if it is deemed fit to do so.

View
This subsystem will be responsible for the GUI.

Controller
This subsystem will be responsible for handling the user input and interacting with the different controllers.

The exact package structure for all of these subsystems will be added once they have been decided by the development team.

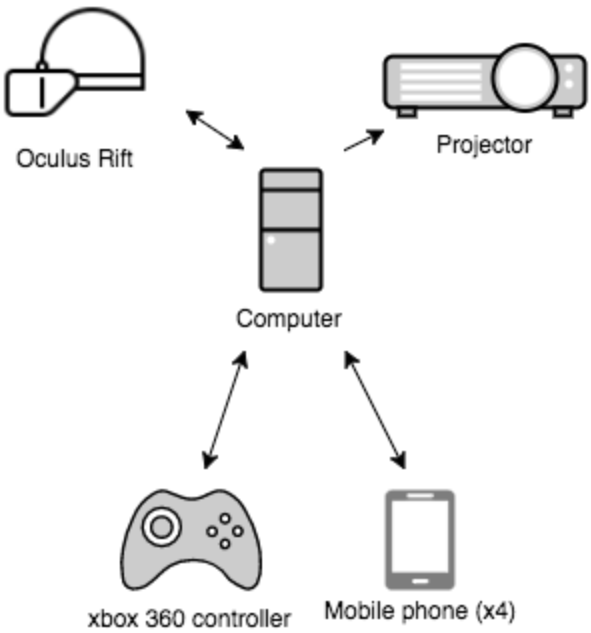
2.2 Hardware/software mapping

There are multiple hardware components which need to interact with each other. There is a single computer on which the game runs. Connected to the computer are the following components

- Oculus Rift (x1)
- Xbox 360 controller (x1)
- Mobile phone (x4)
- Projector (x1)

One player uses the Oculus Rift to view the game, and the xbox 360 controller to control the game.

Four other players use the projector to view an image of the map, and mobile phones to control the game. The mobile phones in turn display relevant information to what is currently going on and the relevant controls for the player.



2.3 Concurrency

As there are multiple people playing the game at the same time concurrency is a something we have to consider. As it is not possible to predict which concurrency problems will occur during the making of the game, this part will be expanded upon once the development team decides a way of tackling these problems.

3 Glossary

Branch
A split from the main code to which people can push changes and additions to the code.

Checkstyle
A static analysis tool which determines if the code complies with the style rules agreed upon by a development team.

FindBugs
A static analysis tool which detects possible bugs in java code.

Git
A version control system used for software development.

Github
A web-based git repository hosting.

GUI
Graphical User Interface, an interface which allows users to interact with electronic devices.

Merge
Blending of the code between two different branches.

MVC
Stands for “Model-view-controller”, A software design pattern for implementing user interfaces on computers. It splits the code into three different parts which all work together but could be interchanged for different parts.

PMD
A static analysis tool which uses a rule-set to determine if the code is erroneous.

Pull request
A request to add or change code and deliverables.

Push
Uploading code to a git repository.