Architecture Design - Carried Away

1. Introduction

In this document you can find a sketch for our context project on computer games. We will tank about the design goals and the software architecture. This document will be modified when needed.

1.1 Design goals

Availability
Following the *Scrum* principle, at the end of each *sprint* a working product should be ready for users to try out. This way testers can give us feedback on the current system each week and the developers can adjust the system accordingly. It also makes sure that the game studio doesn't deviate the game from what the users actually want.

Performance
What every user wants in a game is a very good performance. The Oculus Rift can be quite performance intensive for older systems and non gaming laptops. Therefore it's imperative that we focus on higher performance to make sure the game runs very smoothly.

Reliability
Just like performance, running into bugs during their gameplay experience ruins the fun. With each new feature, the amount of entropy in the system increases drastically. It should be the goal of the developers to keep that as low as possible. Every developer is responsible for their own code, so each of them will have to make sure that all their code is nearly bug free.

Manageability
When we want to change a part of the game, we want to be able to do this without any problems. To accomplish this, we want our game to be playable after every sprint, as well as keep our code modifiable. Every developer is responsible for their own code, so they'll each have to make sure all their code is kept manageable.

Scalability
As new features are added, the program not only becomes less and less manageable, the scalability is also compromised. When a developer adds a new feature, he or she should keep in mind if it affects the program as a whole for later expansion. Sometimes spending some more time on a feature to make the program future proof is very much worth it in the long run. Developers should consider this when implementing a feature and sometimes put in some extra work.

Securability

Since our game will be played on LAN only, we don't have to secure our game. When the internet connection that our game will be played on is safe, our game is safe. *Or is it?*

2. Software architecture views

In this chapter we will explain the architecture in the form of components of the system, which will be split into sub components and subsystems.

2.1. Subsystem decomposition

The system has been divided into three subsystems. The Client Interface, Server Interface and Server Resources. The interaction between these systems is illustrated in figure 1.

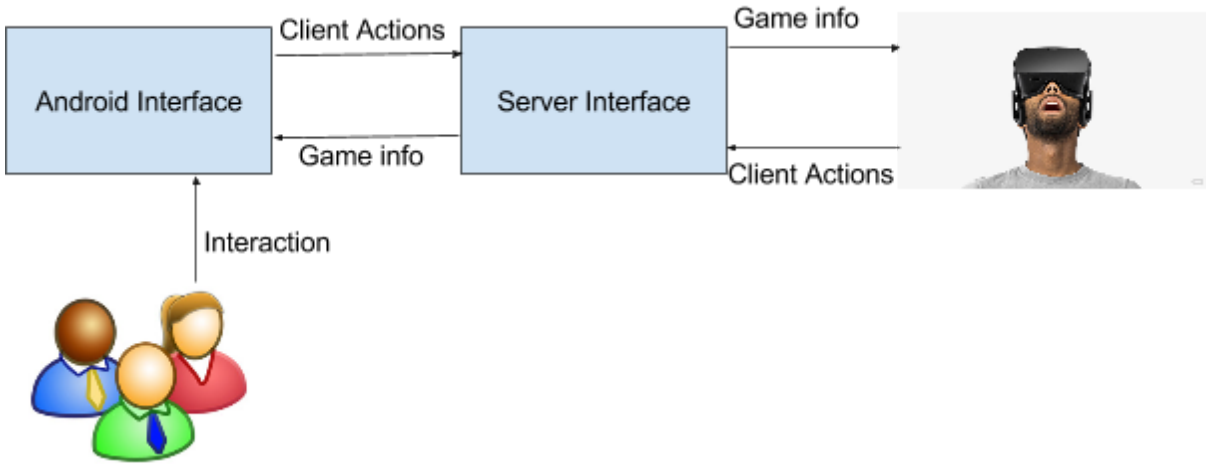

Figure 1. Architecture Diagram

In this section the subsystems are described in more detail.

- The Android Interface allows its users to interact with it. The actions the users commit are send to the server in order to influence the world.

- The Server Interface is used to connect every client with each other. It is responsible for handling the entire game. It sends information of the game (for example the location of the player in the world) to the Android users and the Oculus Rift user. What information the server sends to who depends on the game design.

The Oculus Rift should be directly connected to the machine where the server interface is running on. So everything done by VR-user is directly being handled by the server interface, instead of through an extra interface like the Android interface.

2.2. Hardware-Software mapping

Both the hardware and the software for the server are not the same as the software and hardware for the Android devices. The Oculus Rift should be directly connected to the server using a HDMI and a USB cable. Through the Server UI (which can be operated by any user) a lobby can be started, to which the Android users can connect using the Android UI and the Local Area Network where everyone is connected to.

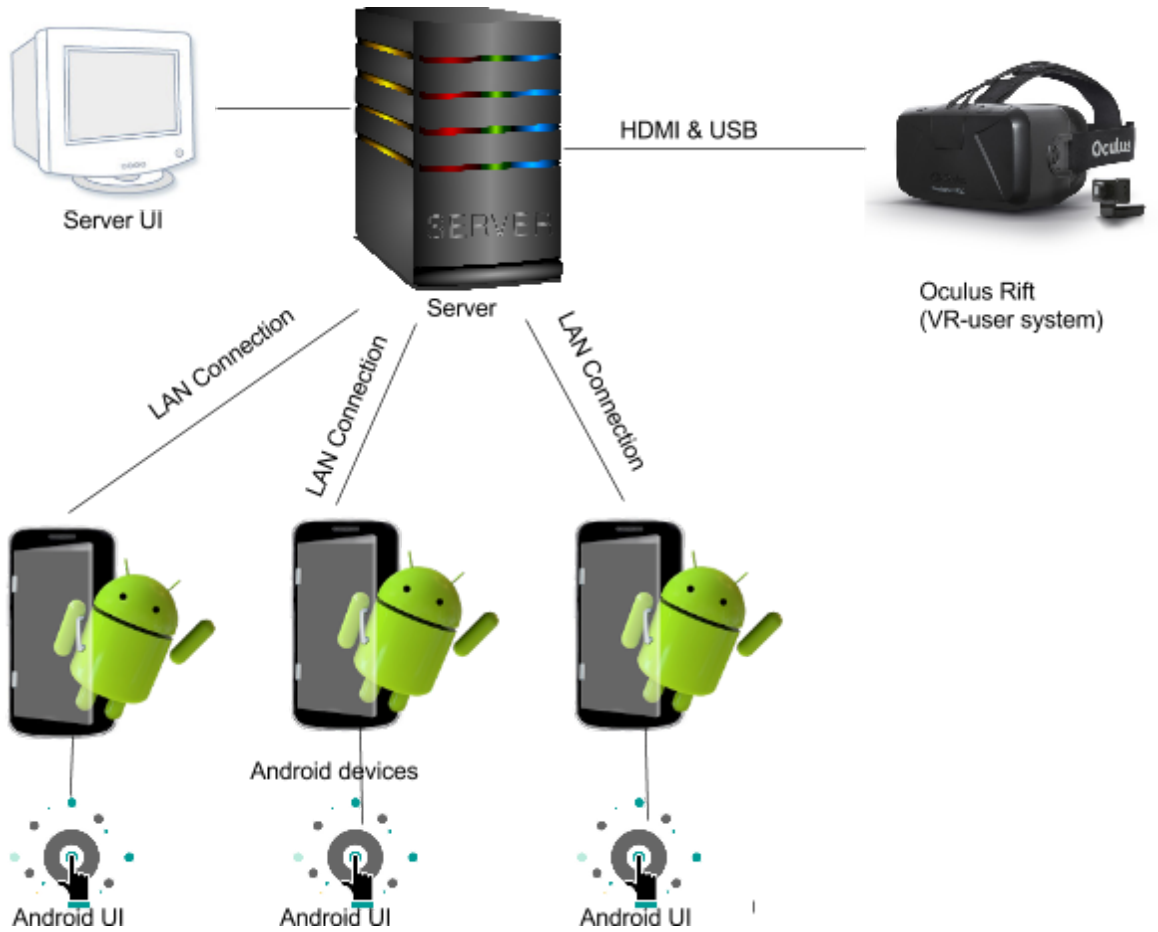

Figure 2. Illustration of the Hardware-Software mapping

2.3. Persistent data management

We won't use large amounts of memory, but we will keep our models, textures, animations, etc. in an assets folder. The highscore feature, if implemented, will require a file where the statistics are saved. This and possible other statistics that need to be saved will all be contained in a *resource* folder.

2.4. Concurrency Control

The Android devices repeatedly send messages to the Server and vice versa. This may cause some concurrency problems if the Server received many messages at once.
To prevent this, a message coming from one device should not depend on message sent by another device. This can be achieved by storing information sent from the clients separately. So a information from a client *A* should be stored on a different location than a message from client *B*. The information should then be processed by one single thread.

3. Project Modules

Because of the clear distinction between the desktop application and the Android application, the project has been split up into two modules. These modules and their relations to each other are discussed in this section.

3.1. Project Architecture

The Android app and the desktop app are contained within their own separate Maven projects. A third Maven project has both of these projects in its project folder and builds them consecutively. This setup is very simple and easily allows the addition of more modules.

3.2. Individual modules

In practice, the module for the Android application and the desktop application are completely separated, which has some implications for the project.
An upside is that it allows each individual component to deal with similar problems in a different way. For example, the desktop application runs on Java 1.8, while the Android Application runs on Java 1.7.
A downside is that you will need to define some things twice, which can lead to problems when forgotten.

4. Glossary

Scrum: A software development framework that uses agile designing and rapid prototyping. It requires a working version of the software to be delivered at the end of each 1-week sprint.

Sprint: An iteration in the scrum process. Every sprint has the same length, with a maximum of time one month. This project uses sprints of 1 week each.