

Emergent Architecture Design Document

DuoDrive

Pim van den Bogaerd, pvandenbogaerd, 4215516

Ramin Erfani, rsafarpourerfa, 4205502

Robert Luijendijk, rluijendijk, 4161467

Mourad el Maouchi, melmaouchi, 4204379

Kevin van Nes, kjmvannes, 4020871

May 15, 2014

TI2805 Contextproject, 2013/2014, TU Delft

Group 5, Computer Games

Version 1 (draft)

Abstract

This document describes the architecture of the system that our product consists of. Firstly, we will describe the design goals for our final product. After that, our Software Architecture Views will be treated. Finally, a glossary can be found at the end of the document. This document will be edited and changed throughout the duration of the project to fit our current system architecture.

Contents

1	Introduction	4
1.1	Design goals	4
2	Software Architecture Views	5
2.1	Subsystem Decomposition	5
2.2	Hardware/Software Mapping	5
2.3	Persistent Data Management	5
3	Glossary	6

1 Introduction

In this document we will describe the architecture of the system that we will be creating during the Computer Games Context project. The architecture will be explained in a form containing high level components with the sub-components and sub-systems. This will give more insight in what the system is composed of and how the several elements of the system work together.

1.1 Design goals

There are four design goals that will be maintained throughout the project:

Availability

During this project our product will be *continuously integrated* with the goal of having a working product running at any time. This allows us and the client to test and work with the product at any time, even before the final release. After the final release we will want to make sure that the server is up as much as possible, especially during the time that people are visiting our client's company or area.

Replayability

A design goal that we consider to be specific for our product is replayability. We want the players to enjoy the game in such a way that the game will be replayable in both the short and long term. We want the players to find our product memorable, so they may even look forward to playing the game the next time. To reach this goal, the game should make a solid first impression, after which elements such as randomness and other unpredictable events will keep the game interesting to play.

Usability

The game is designed in such a way that without, or with few, instructions the player will be able to play the game. After a few minutes, or even seconds, the player will have learned how to work with the system and play the game even better. The physics in the game are set to make the user experience better and making the efficiency of playing the game at its best.

Moreover, with feedback given during the game, the user will be able to see progress and see how well he is playing the game.

Performance

The system will have a central server which is situated in the same room/building/area as the players who are playing the game. Also, all the players should be connected on the same network as the server is. Since the server will be the machine sending and receiving the data of the players, it is necessary that the connection between the player and the server is fast enough. This is needed to ensure that the players will not encounter any *latency issues*.

In a future release we might implement a system where some of the server's calculations will be calculated locally on the player's device. This will take load off the server and may improve the connection speed throughout the game.

2 Software Architecture Views

This chapter will discuss the architecture of the system and how it is decomposed. In the first paragraph the subsystems will be decomposed and we explain each subsystem. In the second paragraph we will elaborate upon the relations and mapping between hardware and software. In the last paragraph the data management will be explained.

2.1 Subsystem Decomposition

The software architecture of the system consists of three different subsystems: the server, the system for the steering player, and the system for the player who controls the throttle. These different subsystems are explained in this section.

- **Server**
The server in the system maintains the data flow. It will send the data of the positions of the cars to all players, so that every player has a near real-time experience, where they can see other players' positions. All data sent by a player will first be sent to the server, which distributes it to the other players. There is no player to player data flow at this moment.
- **Steering player system**
The system for the steering player will only contain the view that the steering player should see. This will be a limited view of the track and not being able to control the throttle. It will only have the ability to steer using a *gyroscope* available in the device of the player.
- **Throttler system**
In the throttler system the abilities to perform certain actions is also limited. The player will only be able to control the throttle, i.e. accelerating forward or backward, and hitting the brake. By limiting the view and the possibilities to perform actions, communication will be enforced.

2.2 Hardware/Software Mapping

The hardware used is very various in terms of the player-held device. That device must have Android running and must contain a *gyroscope* and *accelerometer*. The only other piece of hardware that will be required is a central computer that will serve as a central server.

The software that is being used on the described hardware is the same. Depending on the functionality, so either being a steering player, throttler, or server, the interface will be different. The server has a simplistic interface which will make it possible to start or end the server.

2.3 Persistent Data Management

Persistent data management is not needed in our product. There is no data that needs to be saved persistently.

3 Glossary

Accelerometer (Oxford dictionary (British & World English))

An instrument for measuring the acceleration of a moving or vibrating body.

Continuous Integration (Wikipedia, 2014)

Continuous integration is the practice, in software engineering, of merging all developer working copies with a shared mainline several times a day. This type of integration is used to make sure a working copy is always available at any moment during a software engineering project.

Gyroscope (The American Heritage Dictionary of the English Language, Fourth Edition)

A device consisting of a wheel or disc mounted so that it can spin rapidly about an axis which is itself free to alter in direction. The orientation of the axis is not affected by tilting of the mounting, so gyroscopes can be used to provide stability or maintain a reference direction in navigation systems, automatic pilots and stabilizers.

Latency Issues

In a network, latency is the amount of time a packet needs to get from one point to another, i.e. from a server to a client. Latency issues arise when a packet takes too long to arrive at the endpoint, the fluency of the program may be compromised, because packets may take too long or be dropped completely.

References

Accelerometer. (n.d.) Oxford dictionary (British & World English). Retrieved May 15, 2014, from <http://www.oxforddictionaries.com/definition/english/accelerometer>

Continuous integration. (n.d.). In *Wikipedia*. Retrieved May 15, 2014, from http://en.wikipedia.org/wiki/Continuous_integration

Gyroscope. (n.d.) The American Heritage Dictionary of the English Language, Fourth Edition. (2003). Retrieved May 15, 2014, from <http://www.thefreedictionary.com/gyroscope>