Delft University of Technology

Context Project

Final Report

DuoDrive

Pim van den Bogaerdt	pvandenbogaerd	4215516
Ramin Erfani	rsafarpourerfa	4205502
Robert Luijendijk	rluijendijk	4161467
Mourad el Maouchi	melmaouchi	4204379
Kevin van Nes	kjmvannes	4020871

June 26, 2014

Abstract

This document is the final report for the Context Project of the Delft University of Technology (TU Delft). In this document a video game will be described and evaluated. This is done by us, a group of five Bachelor students at the TU Delft. The video game is developed for use in a context given by the university. The context that was chosen by the team members was the Computer Games context. This context meant that a game had to be developed in which several people within the same waiting room could play together. The goal was to create a communicative and amusing game to entertain those people. The resulting implemented product is called DuoDrive.

DuoDrive is a racing game where two teams of two players each control one car and try to reach the finish. This document outlines the results of the development of this game, including a high-level description of DuoDrive, the developed functionalities and the functional modules. Furthermore, applied Human-Computer Interaction aspects are discussed, a failure analysis is done and an outlook is given.

Contents

1 Introduction

This section is based on the Context Project guidelines (Technische Universiteit Delft, n.d., p.1) and the assignment for the Computer Games context (Technische Universiteit Delft, 2014, pp.1-2).

The Context Project is a project in which students ”develop, implement, validate, present and demonstrate a software product”. Students will work in teams to develop ”a software product that satisfies the needs of an external party in a given non-ICT” environment. As with other professions such as medicine and psychology, practical knowledge is about being rather than knowing, meaning that it can only be taught through experience. The purpose of the Context Project is for students to develop their practical knowledge and skill in software engineering using an agile software engineering process. (Technische Universiteit Delft, n.d., p.1)

The end-user requirements encompassed that the game had to entertain people while waiting in a dentist waiting room, waiting for the train to arrive or waiting in any place where co-location unites these people. On these occasions two pairs of people should be able to play with or against each other. These players have to virtually communicate using some kind of technology like Bluetooth, camera, Wi-Fi etc. However, the game should require minimal technology to be playable. The product must be reachable to a large crowd without the need of too much technology. Furthermore, players merely virtually communicating is not sufficient. A solution to this problem is implemented in the final product. (Technische Universiteit Delft, 2014, pp.1-2)

Firstly, an overview of the developed and implemented software product, the game, will be given. Hereafter, the developed functionalities of the product will be described. Next, the Human-Computer Interaction surrounding the game will be discussed. After this, the functional modules and the failure analysis will be evaluated. Finally, an outlook is given, containing ideas and improvements for the future of the product.

2 Overview of the Developed and Implemented Software Product

The implemented product, DuoDrive, is a top-down, 2D racing game that can be played on portable Android devices. In this game, two pairs of people each control one car. Specifically, the game is designed for waiting rooms where two lines of two chairs are placed back-to-back. Two pairs of people team up and each pair chooses two chairs placed back-to-back. Each pair will control one of the cars: one person controls the steering wheel (he/she is the ”driver”) and the other controls the throttle (he/she is the ”throttler”).

This concept agrees with the design document (Bogaerdt, Erfani, Luijendijk, el Maouchi, & van Nes, 2014). During development, this concept has been implemented concretely. The throttler and driver do not see the same view; instead, the driver sees the car with a small area of light in front of it, i.e. a car headlight. The remaining space around the car is dark. On the other hand, the throttler sees an overview of the whole track, but the car is shown only as a dot. Both players also see signs: the throttler is given signs on how the driver should control the steering wheel (through driver signs of e.g. a U-turn), and the driver occasionally sees signs when the throttler should slow down. Because a player does not see the screen of his teammate, physical communication is enforced in order to reach the finish.

During the race, the competitive aspect is apparent: the goal of the game is to reach the finish at the end of the track first and each team should try to achieve this goal. The collaborative aspect is also apparent, because within each team the two players have to collaborate through physical communication.

3 Description of the Developed Functionalities

The software product that is developed, like stated in the previous section, has several implemented functionalities. Every functionality contributes to the product in its entirety. Moreover, the functionalities have the purpose of making the game more interesting and enjoyable to play. In the upcoming sections, the game is split up in three main components. In each subsection the corresponding functionalities are elaborated on. In the final section the functionalities are combined and a description will be given about how they contribute to the product attributes.

3.1 Visual aspects

In a video game several visual aspects should be visible to the players. These aspects provide the player with information. However, they can also show views which encourage attributes such as communication. The following subsections will discuss the different views and their corresponding functionalities.

3.1.1 Introductory view

The first view that is shown when the game is started contains, in total, five buttons. At the top a screen-wide button is placed, which allows a user to view the tutorial. The other four buttons correspond to the roles and teams the player wants to join. The tutorial contains a brief description of what the goal of the game is and instructs the players how to perform in either the driver or throttler role. The tutorial is intended for the players who do not have an idea of how to play the game yet.

3.1.2 Driver view

The view for the driver in the game is very limited. The driver sees two buttons that allow him to steer either left or right. He only has a limited view in front of the car. This limited view encourages the driver to communicate with the throttler. It is of major importance to communicate with the other team member to have knowledge of where to steer to.

During the game the track can have road marks that might deceive the driver. An example is that sometimes the road marks might appear as if the road will continue in a straight line, while in fact a turn is coming up. This feature encourages the throttler to communicate to the driver about where to steer to.

3.1.3 Throttler view

The view for the throttler on the other hand is not so limited. The throttler has an overview of the whole track. Additionally, he has a road mark sign in the upper-left corner of his screen. This sign is shown from the perspective of the driver of the car. Moreover, the throttler has two gas pedals at the bottom-left and bottom-right corners. The left one allows the throttler to decelerate or drive backwards and the right pedal is for accelerating.
On certain tiles spread over the track, mud has been placed. Depending on the speed of the car, the car will be slowed down if it drives into a mud pool. If the speed of the car is very high, the car will be slowed down by a drastic amount in an instant, whereas a car that approaches the mud more slowly will drive over it with the same amount of speed, without being slowed down by the mud. This once again enforces communication between both players, as to make sure the car does not get slowed down by the mud. The driver receives a *"SLOW DOWN"* sign on his screen whenever the car approaches a part of the road with mud on it.

3.2 Gameplay

In the game the player will have different ways of controlling the car depending on his role. A player can either be a driver or a throttler within the game. By splitting up the controls, communication is enforced between both players. In the following subsections the goals for both roles and the functionality they contribute to are explained.

3.2.1 Driver

The driver in the game only has control of the steering wheel. With the given buttons on his view the car can be steered either left or right. However, the driver does not have the ability to throttle. This has to be done by the throttler. Since the driver can not control the car on his own, the players have to communicate with each other to reach the finish.

3.2.2 Throttler

The throttler in the game only has control of the throttle. With the gas pedals on his view the car can be moved either forward or backwards. However, the driver has no ability to steer the car. This has to be done by the driver. The throttler needs the help of the driver and the other way around. This way a pair of players can work together to reach the finish.

3.3 Finish screen

When the first car has reached the finish the players will be shown that they have done so. The time they took to complete the track will be presented to them. If the other team has already finished, all players get a view of the ranking board, in which both times are shown. This ranking board contains the time the players have finished in and whether they finished in first or second place. The team that finished first will have their time in yellow on the ranking board.

3.4 Crucial product attributes

With the presented functionalities stated above, the crucial product attributes described in the design document (Bogaerdt et al., 2014, pp.5-6) have been achieved. By having user tests (discussed in section 4.4) throughout the product, it has been assured that these functionalities work as intended. This is of utmost importance, because the described functionalities are crucial for the game.

4 Human-Computer Interaction

This section discusses the measures taken to comply to the Human-Computer Interaction (HCI) principles. These principles are designed to improve the way people use computers by making the computers more intuitive to use.

4.1 Completed Interaction Design

To start off, there are two people in this group that have already completed the course TI2600 Interaction Design last year:

Kevin van Nes, 4020871 | Robert Luijendijk, 4161467

4.2 Introduction

During this project, it was important to use HCI and its principles. This is because a game had to be created for this project. Games are full of interaction between the user and the device the game is played on. The game that was created for this project is also full of interaction. Communication plays an important role, so certain ways had to be found to make sure that e.g. controls and information are clear to the players.

4.3 Analysis

In a game there are many features that need attention when it comes to interaction. The first feature is the start screen at the beginning of the game. All buttons on the start screen and their functionality must be clear to the player. The product is a racing game, so the controls of the car must be responsive and intuitive to use. Because the game will be played on devices with a touchscreen, players must know how to throttle or steer without or with little explanation. The above described features are main issues when it comes to HCI in this game. These features will be discussed further in the following subsections.

4.3.1 Start screen

The start screen is the first screen that players of the game will see. There are many buttons visible on the start screen and each of these buttons has a functionality that must immediately be clear to players. Since every player will pass through this screen before being able to play the game, the screen must be visually attractive and it has to be obvious for people what options they have to continue at the same time. For the reasons given above, the buttons have been made recognizable and obvious. This is done in such a way that they adjust to the size of the screen of the device the game is played on. The buttons on the screen vary in size and color, initially this will attract the attention of the players. There is also a button that leads to a tutorial for the game. This will help the user to play the game with necessary information about controls and the goal of the game at his disposal.
The game can only be started and played when there are four players connected. The players will be informed whether there is a sufficient amount of players to start the game at all times.

4.3.2 Racing game

After players have gone through the start screen, the players will enter the game itself. To avoid confusion, players need to see and understand what car they are controlling. To reach this goal each car has been given a different color. Moreover, there is a clear separation between the views of the two different roles. There is a clear separation between the track and the grass, so players will not be confused about where to drive. The controls of the car and their responsiveness are also important factors. The speed and steering of the car must act responsively and intuitively to the actions given by the player.

4.3.3 Touchscreen

The touchscreen is the actual connection between the game and the player and thus also subject to the HCI principles. The buttons on the touchscreen need to be pressed. These buttons are, depending of the role the player chooses, either to drive or to steer. These buttons must be clearly visible to the user. The users must be able to find out what the buttons do without too much thinking. They are designed in such a way that the user will know what a button will do, before the user pushes the button. The player who controls the throttle will see arrows on road signs to communicate to the driver about where to steer to. The arrows are designed in such a way that the function of these arrows is obvious.

4.4 User testing

Throughout the development several user tests have been done. The intention of our user tests was to receive feedback of users who have never played the game before. This is important because the perspective of players differs from the one of the developers. Because of this, ideas and improvements are presented which can be taken into consideration. The user tests were held in a room where communication is freely available. The users were asked to play the game, afterwards questions were asked regarding several features in the game. The questions were based on a premade question sheet. The answers to those questions were analysed afterwards and taken into account throughout the development. The results from the user tests are given in the upcoming section.

4.5 User test results

The beta of the game had two different versions for both roles; a version with a complete overview and one with a limited overview. The versions were made for the purpose of user testing. The two versions made it possible to gain more accurate feedback. Based on the feedback received on both versions a choice was made on which view would be implemented. In the following subsections the two different versions are elaborated on and the results of the versions are given.

4.5.1 Complete overview of the track

The complete overview makes it possible for the throttler to see the whole track. However, when the overview is turned off, the throttler is only able to see traffic signs in the upper-left corner to guide the driver.

Turned ON The tests where the complete overview was turned on, resulted in useful feedback. The players that tested this version, gave the feedback that the communication came mostly from the driver. However, the gameplay was more fun according to the test results, which will be explained below.

Turned OFF The results with this overview turned off were slightly different from the ones with the overview turned on. The only major exception was that the players of this test experienced relatively more chaos. This resulted in an effect with which the throttler had the feeling that he was more busy shouting directions instead of playing a game. However, without an overview, players would quickly feel that the game became too hard.

After doing an analysis of the test results, a conclusion was made. The version with the complete overview turned on contained more positive feedback. This resulted in implementation of version where the complete overview is turned on.

4.5.2 Limited view of the driver

The limited view is meant for the driver. In this view, when turned off, the driver has a clear view surrounding him. However, when turned on the driver only sees light from his headlights in front of him.

Turned ON The results of the user tests where the limited view was turned on resulted in the player having more trust in their partner. Hereby, they steered towards the direction told to them by their partner. Another result from the test was that the limited view did not bother the players and increased the experience in gameplay.

Turned OFF After the test with the limited view turned off the users told that they could see the corners coming up theirselves and did not need the communication with the throttler. This contradicted with the design goals of the product.

After collection of the test results, a conclusion was made. The communication between the partners is a key element and therefore the version with the limited view turned on was chosen.

5 Evaluation of the Functional Modules and the Product in its Entirety

In this section, an evaluation will be made of the functional modules and the product in its entirety. Each of the components of the product will be evaluated separately. After this, a failure analysis of the system will be done.

5.1 Evaluation

In the software system there are several components which are put together based on their responsibilities. These components each have a main responsibility, but they still communicate with other components. In this section the components will be evaluated.

5.1.1 Behaviours

This component exists out of classes which handle the behaviour of the created GameObjects. A car is a GameObject and one of the things that are handled within this component is that a car should collide smoothly when colliding with the bounds of the track. The elements associated with the car, like position, rotation and the notification of whether a car has finished are all handled within this component.

5.1.2 Car

The Car component consists of classes associated with a Car object. It contains a Car class and Player class, where the Player class can be split up into a Throttler and a Driver. A Car consists of a maximum of two Players. The Car has a behaviour attached to it. The Driver and Throttler classes are focused on their functionalities and perform actions based on their Role in the game. In this component it has been made sure that the Single Responsibility Principle of Software Engineering is not violated.

5.1.3 Controllers

As the name suggests, this component exists of the Controllers that have been made for the software product. This component has been made to take off some load from e.g. the network. In this way it is possible to have more control over what happens and have the classes within each of the components focus on their respective responsibilities.

5.1.4 GraphicalUI

The GraphicalUI (GUI) component is the largest component of the software product. It has been created in such a way that it is possible to create additional GUI elements in the future. The component's main focus is the Graphical User Interface. By splitting up the GUI in smaller parts it has become more controllable and maintainable. By using the GUI 'Parts' it is simple to adjust, tweak and improve the elements within the GraphicalUI component.

5.1.5 Main

This component does not have a very active responsibility. Instead, it has an overview of the other components. The MainScript class within this component contains a GameObject. By passing through the MainScript class it is possible to gain information about the game.

5.1.6 NetworkManager

The NetworkManager component manages everything around the network. The 'network' consists of a server and clients, which each have their own, respectively named class. The Server class controls which player should get what role. It also controls the spawning of the cars and everything around it. The Client class is attached to a Player, since every Client is in fact a new Player.

5.1.7 Utilities

This component contains classes with either constant data that does not change in the software or classes that contain methods that are used very often. By putting frequently used methods together in appropriate classes, it was much easier to gain access to them. At the same time this reduces code duplication and the Single Responsibility Principle is maintained.

5.2 Failure Analysis

Analyzing possible failures of the system has been crucial throughout the whole duration of the project. From the moment the game concept was picked (and even before that), a lot of thorough research had to be done on whether Android devices would be able to offer a real-time online multiplayer experience in terms of their hardware capabilities.

It was decided that Unity would be the main development tool for the game, because Unity offers a lot of freedom and potential for creating a game in a rather short time span. During the first phase it was found out that Unity could, at some points, be rather limiting instead of simplifying. Two of the major limitations that were points that could (and still might) make the system fail will be discussed below.

5.2.1 Unity's Development Server

One of the first possible things that could fail is that the server, created by the game, has to communicate with Unity's 'Development Server'. The game's clients (the Android devices) also have to communicate with this Development Server. The problem with this is that the game is dependent of whether Unity has their Development Server running at all times or not. This means that the Development Server might fail at some point, although the odds of this happening are not big enough to pose a problem.

5.2.2 Unity and Remote Procedure Calls

The second feature of Unity that is perceptible to failure is the use of Remote Procedure Calls (RPC). As explained in the Concurrency paragraph of the Emergent Architecture Design document, a first idea to implement network play was to use state synchronization. This is a feature in Unity that allows for state synchronization of Unity GameObjects. This posed a problem in combination with the main game idea (two players controlling one car) and so a different solution to tackle this problem is implemented: using RPC. A problem that comes with this solution is that RPC can cause network congestion if too many calls are made at the same time. Congestion would cause latency issues and this would completely break the game experience at any time. This makes network congestion the main threat for system failure in the game. The risk of network congestion has been noted as something that should be fixed or at least reduced in the future. This will be expanded on in section 6.1.1.

To conclude, the system is perceptible to failure in a few ways. These failures might be caused by Unity on one hand, but it has been made sure that these problems are either worked around or reduced to a minimum.

6 Outlook

The final product is stable and all the crucial product attributes have been achieved. However, there is always room for improvement. The possible improvements and scenarios for the future of the game are listed below.

6.1 Improvements

For the game there are several improvements that can be made. Some of them are based on the gameplay. Features that have been thought of are random track generation and power-ups. Other improvements are based on the software part of the product, e.g. improving the network capabilities and having an own server. The mentioned improvements will be discussed in the upcoming subsections.

6.1.1 Network capabilities

The currently implemented network might fail due to latency issues (lag), which could disrupt the player's experience. The lag that can occur is caused by the amount of data that is being sent and received through the network. The network capabilities should be improved in the future.

6.1.2 Random track generator

The game contains a single track at the moment, which may result in players learning the track. This reduces the replayability of the game, which stands in contrast to the fact that replayability is one of the game's design goals. An improvement that can resolve this issue is a random track generator. With this, tracks can be randomly generated whenever a new game is started. This means that the players will not be able to learn tracks by heart, so they must communicate at all times, without relying on priorly gained knowledge.

6.1.3 Powerups

Based on the results of the user tests, the conclusion was made that the game can get boring for the users after having played it several times. A solution to this is to add power-ups to the game. With this, a player can get an advantage over the others, which adds an element of competitiveness to the game. This improvement will also make the game more dynamic and less predictable. Each time the users play the game, they will get a new and different experience.

6.1.4 Own server

As mentioned in section 5.2.1, the game currently runs on Unity's Development Server. However, this makes the game dependent on whether their server is up and running or not. An improvement and solution to this is creating an own server. This resolves the dependency on Unity's Development Server and any issues regarding the server can be resolved more quickly.

6.2 Future of the game

As described in the above section, there are several improvements that can improve the gameplay and user experience. In the upcoming subsections, scenarios for the future of the game will be discussed.

6.2.1 Further development

A possible scenario is that the product will be further developed by the developers. This could mean that either one team member continues to develop the product on his own or multiple group members continue developing the game as a group.

6.2.2 Delft University of Technology

The second possibility is that the game can be used by the Delft University of Technology (TU Delft). The product might be used as a project for the students to implement new features. This might include improving their programming skills.

6.2.3 Open source project

A final possibility is that the product might become an open source project. This scenario will make it possible for the game to be picked up by other developers. They can improve the concept and add or improve elements of the game. However, this scenario should be discussed with the TU Delft, since the product is in their possession.

References

[1] Bogaerdt, P., Erfani, R., Luijendijk, R., el Maouchi, M., van Nes, K. (2014). *Design Document DuoDrive*.

[2] Technische Universiteit Delft. (n.d.). *Context Project Guidelines 2013/14*. Retrieved on 25 June, 2014 from https://blackboard.tudelft.nl/bbcswebdav/pid-2201336-dt-content-rid-7406373_2/courses/30183-131404/ContextprojectGuidelines2014.pdf

[3] Technische Universiteit Delft (2014). *Computer Games*. Retrieved on 26 June, 2014 from https://blackboard.tudelft.nl/bbcswebdav/pid-2230196-dt-content-rid-7592291_2/courses/30183-131404/Assignment.Games.Project.2014.pdf