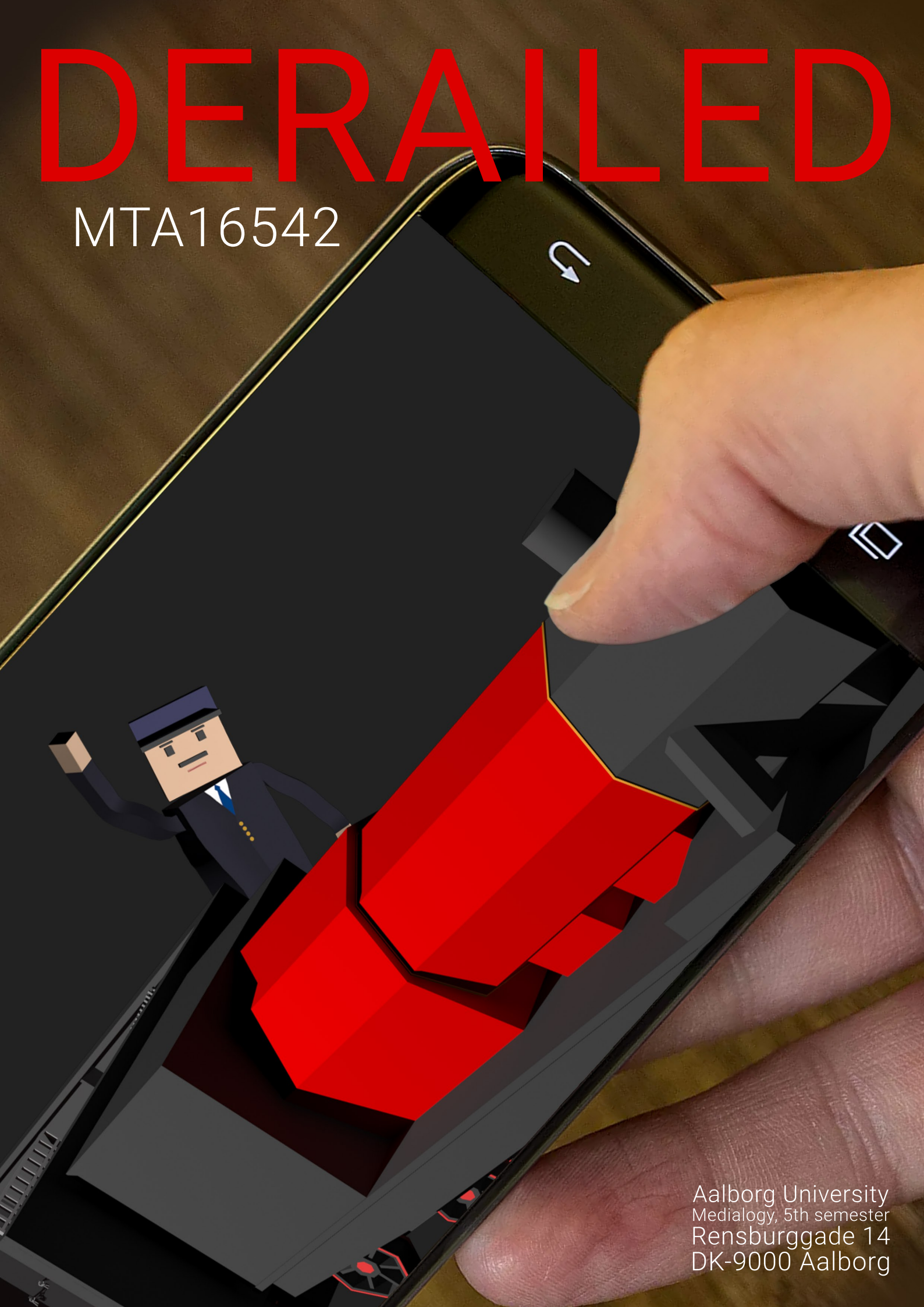
DERAILED

MTA16542



Aalborg University
Medialogy, 5th semester
Rensburggade 14
DK-9000 Aalborg

AALBORG UNIVERSITY
STUDENT REPORT

Department of Architecture, Design and Media Technology

Medialogy, 5th Semester

Title:
Derailed – A multi-screen platform game

Project Period:
P5, Fall 2016

Semester Theme:
Audio-Visual Experiments

Supervisors:
Kasper Hald

Projectgroup no.
MTA16542

Members:
Anders Johansen
Daniel Bruun Hansen
Mathias Wessmann
Mikkel Damgaard Vindum
Nicolai Bruhn Lauritsen
Thor Vest Tidemand

Abstract:

This project investigates if visual cues can be used as a means to reduce the reaction time of players, when having to swap focus between two screens whilst playing. Multi-screen gaming platforms face the unique challenge of players having to manage their focus between screens as they play. To test if visual cues can help players manage their focus, the game Derailed was developed for the platform AirConsole. This game was then tested in a control group experiment with and without the use of visual cues. The project concluded that visual cues shows promise in helping players manage focus, and proposes new areas to investigate in future work.

Contents

1 Introduction **1**

2 Project Description **3**
2.1 AirConsole's Project Proposal . . . 3
2.2 Problem Description . . . 4
2.3 Initial problem Statement . . . 4

3 Background Research **5**
3.1 A TV/mobile multi-screen . . . 5
3.2 Methods Used to Evaluate Previous Games . . . 6
3.3 State of the Art . . . 6
3.3.1 WiiU . . . 6
3.3.2 Nintendo 3DS . . . 7
3.4 Game Research . . . 8
3.4.1 Multi-screen . . . 8
3.4.2 Micro-games . . . 9

4 Final Problem Statement **11**
4.1 Final Problem Statement . . . 11
4.2 Success Criteria . . . 12

5 Design **13**
5.1 Game Design . . . 13
5.1.1 Players . . . 13
5.1.2 Objectives . . . 14
5.1.3 Procedures . . . 14
5.1.4 Rules . . . 14
5.1.5 Resources . . . 14
5.1.6 Conflicts . . . 15
5.1.7 Boundaries . . . 15
5.1.8 Outcome . . . 15
5.2 Graphics . . . 16
5.3 Level Design . . . 17
5.4 Controller . . . 18
5.5 User Interface . . . 20
5.5.1 Heads Up Display . . . 20
5.6 Sound . . . 21

6 Implementation **23**

6.1 AirConsole . . . 23
6.1.1 Communicating between screen and controller . . . 23
6.2 Unity . . . 24
6.2.1 AirConsole Plugin . . . 24
6.2.2 Functionality and Ease of Access . . . 24
6.3 AirConsole Controller . . . 24
6.3.1 Game Controller . . . 24
6.3.2 Micro-mechanics . . . 25
6.4 Visual Cues . . . 25
6.4.1 Highlights . . . 25
6.4.2 Pop-Ups . . . 26
6.5 Model Creation and Animations . . . 27
6.6 Eventsystem Structure and Handling . . . 28
6.6.1 Event Struct . . . 28
6.7 Testing and Collecting Data . . . 29

7 Evaluation **31**
7.1 Hypothesis . . . 31
7.2 Design of Experiment . . . 31
7.3 Setup . . . 32
7.4 Procedure . . . 33
7.5 Results . . . 34
7.6 Analysis of Results . . . 35
7.7 Discussion . . . 37
7.8 Analysis of Method . . . 38

8 Discussion **41**
8.1 Quality of Solution . . . 41
8.1.1 Success Criteria . . . 41
8.2 Wider Context . . . 42
8.3 Future Work . . . 44

9 Conclusion **47**

10 Portfolio **51**
10.1 Game Concept . . . 51
10.2 Game Art . . . 52
10.2.1 Sketches . . . 53
10.2.2 Character Design . . . 56
10.2.3 Character Icons . . . 58
10.2.4 Train Design . . . 59
10.2.5 Workstation Design . . . 61
10.3 Sending and receiving information . . . 62
10.4 Game Controller . . . 63
10.5 Micro mechanics . . . 65
10.6 Visual Cues . . . 67
10.6.1 Pop-Up Animations and Radial Based Timer . . . 67
10.6.2 Notification animations . . . 68
10.6.3 Highlight shader . . . 69
10.7 3D Graphics and Animations . . . 71

10.7.1 Modeling in 3Ds Max . . . 72
10.7.2 Rigging in Maya . . . 74
10.7.3 Animations in MotionBuilder . . . 75
10.7.4 3D Models and Animations in Unity . . . 76
10.8 Handling events . . . 77
10.9 Tests . . . 78
10.9.1 Consent Forms . . . 78
10.9.2 Mental Model Elicitation . . . 79
10.9.3 Controller Survey . . . 85
10.9.4 Internal Testings . . . 91
10.9.5 Final Evaluation . . . 93

Glossary

Cognitive load: A term from cognitive psychology referring to the necessary effort required to hold, process and manipulate information. Often used synonymously to short-term memory.

Delegate: A delegate (computing) is a type that represents references to methods with a particular parameter list and return type. When you instantiate a delegate, you can associate its instance with any method with a compatible signature and return type. You can invoke (or call) the method through the delegate instance [11].

Goofy: A 'slang' adjective used for describing odd, foolish and hilarious subjects or topics.

HUD: An acronym for Heads-Up Display, meaning any transparent display presenting information to the users without making them look away from their accustomed viewpoints.

Micro-games: A less-known term introduced by Nintendo and indicating a short mini-game or small challenge.

Micro-mechanic: A term coined by this project. It is a shorter version of a micro-game, consisting of only a single game mechanic.

Mini-games: Common term used in game design and game industry as a collective name for subgames smaller and simpler than the main game, but can be more or less self-contained.

N-Dream AG: The company behind AirConsole, located in Switzerland.

Swipe D-Pad: A 4- or 8-way relative swipe-pad used for movement and is one of the components to build controllers for AirConsole. D-pad stands for Directional-Pad.

UI: An acronym for User Interface. Generally it includes everything in a device that the users can interact with.

WebGL: An acronym for Web Graphics Library and is a JavaScript application programming interface for rendering 3D-graphics compatible with web browsers.

Chapter 1

Introduction

Multi-screen gaming platforms are platforms that utilizes more than just a main display for interaction and conveying information. AirConsole is one such platform, where the player uses a smartphone in conjunction with the main display (a SmartTV or a large monitor with a laptop) to play games developed specifically for this platform. The smartphone is both used as a screen and a controller, allowing for some unique design possibilities when compared to other platforms.

Whilst these new design possibilities encompasses great potential for new developments, it also presents designers with a new problem; In order to maximize the potential of the added secondary screen, the player has to actively manage focus between the main and secondary screen. Not managing focus correctly can lead players to miss out on important information or actions whilst playing, and thus tools to help the user change focus when relevant should be implemented.

This project investigates how the use of visual cues can help players change their focus between screens by leading their attention towards it. Specifically, the project investigates if a reduction in reaction time when players need to change focus between screens can be achieved.

To investigate this, the game Derailed for the AirConsole platform was developed. This game sets four players on a train moving between stations, having to work together to keep the train from derailing along the way. To effectively test the impact of visual cues, the game was designed so the user needs to frequently look at both the main and secondary screen. Visual cues were then implemented to help the user swap focus to the secondary screen when relevant.

This solution was evaluated in a control group experiment, testing a version of the game with implemented visual cues against a version without. The conclusions of the evaluation along with the process of making the game and implementing the visual cues can be found within this report.

Chapter 2

Project Description

This chapter includes the project proposal from AirConsole, elaborating on the unique design challenges and possibilities that it brings. Secondly, it contains the problem description and initial problem statement. These sections set the stage for conducting the following background research.

2.1 AirConsole's Project Proposal

Multi-screen is used today in many different variations, but there are still areas with un-utilized potentials. One such area was provided by a game designer from N-Dream AG through the introduction of AirConsole. AirConsole is a online platform meant to change browsers into video game consoles. The concept was to use equipment already accessible to most people, monitor and smartphones, and use them to play games together. This does not require people to buy extra hardware or be somewhere specific to have the hardware necessary to play together. Instead users are able to use a laptop, smart-television or tablet and connect to the game through their browser or downloaded app. Nearly everyone are walking around with their smartphones today, and therefore AirConsole makes it easy to enjoy local multiplayer games through a simple multi-screen mechanic. Because once connected, your smartphones become your controller and your browser becomes the console filled with free games from all kind of developers.

N-Dream AG requested for more developers to make games and were ready to give advice and support throughout the process. Specifically, the company was looking for games with local multiplayer, intuitive smartphone controls and WebGL. The AirConsole platform also introduced unique game design challenges, for example with multi-screen designs and features, personalized controllers, advanced controls like swiping or gyroscope and other innovative choices. They provided guidelines on what makes a good AirConsole game:

- Intuitive controls complementing the platform
- Replay value and interesting gameplay
- Intuitive gameplay and implemented tutorial

The AirConsole project proposal introduces many interesting design possibilities. This project saw unique design challenges in developing a intuitive and playable game with both replay value and real-life interaction between the players. Especially, the cooperative aspect seemed important regarding the local-multiplayer theme. Furthermore, many already established AirConsole

games were just using the smartphones as a controller. This project sought to implement the multi-screen feature as a way of also showing or hiding information from other players.

2.2 Problem Description

Over the years second-screen viewing has greatly increased, especially combined with TV[19], and now both mobile devices as well as televisions are capable of performing more advanced tasks [1]. State of the art smartphones and tablets are constantly evolving and increasing in power, thus prompting new usability, for example utilizing their sensors to function as game controllers [9]. There has already been designed apps for second-screen experience, for example mini-apps to share experiences or go behind the scenes [20]. Likewise, "companion apps" for games have been developed for smartphones and tablets [3], often implemented as a means to supplement the in-game content or interact with the game without actually being at your gaming device.

Instead of developing second-screen content as a supplementary element to an already existing application the company N-Dream AG came up with AirConsole: The TV, PC or tablet functions as your web-based console, while your smartphone is a controller and a second screen [12]. AirConsole does not only provide multi-screen gaming, but a new approach where every player combines their own personal screen with the main screen. This introduces the TV/smartphone multi-screen possibility, while also facing different challenges beyond traditional multi-screen gaming. Challenges such as how to utilize the second-screen for more than just a touch-controller or redundant information, and how to direct the players focus between the screens in an intuitive and natural way. Hence, the opportunity to investigate ways of manipulating attention between screens and whether visual cues could play a relevant role in this process.

2.3 Initial problem Statement

How can animated visual cues help the user direct attention on the relevant screen, on a multi-screen game platform?

- Can users achieve a higher score when using visual cues for maintaining focus on the relevant screens?
- Do visual cues have to be direct or indirect to get the user to focus on the relevant screen?
- Do the size of the individual screen affect the performance of the user in a noticeable way?
- Can visual cues help the users to react faster than they normally would?

Chapter 3

Background Research

Background Research attempts to investigate previous projects and game designs, and study these to avoid making the same mistakes. This research also helps the project transition from initial to final problem description, while investigating tools to evaluate and measure solutions.

3.1 A TV/mobile multi-screen

Second-screen viewing is a growing trend and has been so over the past years, with people watching television while utilizing smartphones, tablets and laptops. In a Nielsen survey in 2013 regarding connected device owners 43 percent of the tablet owners and 46 percent of the smartphone owners reported to use their devices as a second screen on daily basis[19]. Furthermore, Google concluded in their New Multi-screen World report that televisions did not require full attention anymore, and that TV in general was the most common device to be combined with second-screen viewing. Research shows that second-screen viewing leads to a higher cognitive load, thus reduced understanding and comprehension of what they are watching. Further studies indicated that there were no difference in the cognitive load between relevant and irrelevant second-screen usage [19]. With the continuous growth and technological advancement, mobile devices are now fulfilling tasks that previously demanded a PC. The same can be said with televisions going from passive monitors to interaction smart-TVs. The fact that both these artifacts have grown further than their original purpose intended could contribute to the new second-screen state of art [1].

This also explains why the game developing company N-Dream AG has set their focus on this new platform type. They came up with the name AirConsole and made it a free, web-based gaming platform focusing on local-multiplayer aspects. Here, everything between a laptop, PC and smart-TV can act as your console. Connecting a smart-phone using the provided AirConsole code will change the phone into a controller. Local-multiplayer games have more or less been disappearing over the last years. This is what AirConsole wants to change by offering a platform put together by things that you often have with you at any time [12]. With new functionalities comes new challenges as well, in this situation being issues as synchronization between game screen and smartphone, distribution of this new concept and alignment of user interfaces between all the different smartphone types [1]. Working with the topic could eventually lead to other problems being discovered, for example how to direct users attention between the two screens.

3.2 Methods Used to Evaluate Previous Games

Eye-tracking is used in order to understand what parts or areas of the screen that the user focuses on during a play session. When playing the game, what draws the user's attention and do they direct their attention where they are supposed to. Eye tracking can be done manually by watching a video recording of participants, although this is not the most accurate way of doing it. There is different software available for web-cams that enables them to do eye tracking. Essentially, the eye tracking tracks the movement of the user's eye to give an idea of what they are focusing on [4].

3.3 State of the Art

The state of the art section is about investigating and researching currently existing games, projects or platforms that utilize multi-screen features. To take this project into the design and final problem stages it is necessary to be aware of the most recent state of developments.

3.3.1 WiiU

A gaming console developed by Nintendo and released in Europe in 2012. The unique element in this console was that its GamePad controller also had an implemented touchscreen. This second screen would function even though the television was turned off or someone else was watching another channel. Nintendo made it so that the current Wii controllers could be combined with the new GamePad. The GamePad was a technical advancement and the users could point, tilt, shake, tap, swipe and talk to it. The possibilities of having a primary and a secondary screen changed the way games could be developed and designed [16].

Figure 3.1: This example shows how second-screen features are implemented in Mass Effect 3 on the WiiU.

ZombiU

ZombiU is a WiiU game from Ubisoft released in 2012, made for up to two players. However, the local multiplayer aspect of this game is not the relevant part for this project, rather it is the multi-screen implementation. With similarities to AirConsole, ZombiU is played using both the TV and the WiiU-controller screen. Ubisoft has implemented this multi-screen in a way

to increase the horror. Players are primarily looking at the TV, but direct focus to the WiiU-controller screen when performing small microgames. In this state, the TV shifts to a 3rd-person view and players have to keep an eye on both screens since zombies could be approaching [18].

Figure 3.2: Example of the ZombiU multi-screen implementation.

3.3.2 Nintendo 3DS

Like the WiiU the Nintendo 3DS is designed as a multi-screen gaming device, but in this case both screens are physically connected in the same device. In the Nintendo 3DS the upper display is 3.53-inches and the display below is 3.02-inches, but in the new Nintendo 3DS XL edition the screens are larger (4.88- and 4.18-inches). In both examples, the upper display is the main screen and the other is used as a secondary screen for micro-games, maps, options, interface and items. The upper display is a 3D-screen and can be adjusted with a 3D depth slider on the side. The bottom display is touch-sensitive and is meant to be controlled with the enclosed stylus[14], but can also be navigated by the fingertips.

The fact that both screens are located so close to each other enables new second-screen features which are more intuitive and user-friendly. Examples could be showing the avatar on the upper display while showing the map below. Or circling around the battlefield on the main display and showcase the optional battle-moves on the display below. The Nintendo 3DS also makes it easier to be aware of the the main display while doing micro-games on the second-screen, whereas with the WiiU and AirConsole the player have to actually look up at the main display. However, WiiU's and AirConsole's main display could be better fitted for multiplayer due to its larger screen size.

Figure 3.3: Showcases the Nintendo 3DS XL and how an info-scene can be shown on the upper display while navigating the interface below.

3.4 Game Research

This chapter investigates four games divided into two subsections: Multi-screen games and micro-games. The main purpose for this research is to study these games, how they implement these elements as well as the pros and cons of doing it this way. Prospectively this process is meant to grant a better comprehension of the game project. The two subsections are covering two areas and concepts that are very important to this project.

3.4.1 Multi-screen

Quiplash

In this game you can choose between 3-8 players, but also have up to 10,000 participants that functions as an audience and have the possibility to affect the voting. The screen shows a question, and then you type in an answer. The developers, Jackbox Games, say that Quiplash is a game where "you can say anything" [8], and that there are no rules or correct answers. Later on, your answer will be competing against other answers to the same question, everyone else will be voting for their favorite answer. This creates a situation where the number of players do not matter, because you can all be entertained. It also enables the possibility of streaming it and having viewers join the game. This means Quiplash is suitable for both local and online multiplayer.

BioSpil

This application is developed for people waiting for the movie to begin in the cinema. Instead of just sitting impatient in the dark cinema, everyone can connect to the cinema's wi-fi and enter BioSpil with the unique code shown on the screen. A timer on the screen will count down for the game to start, meanwhile players can move their icon around the screen and write messages. The game itself is often a quiz, puzzle or memory test and there is a prize for the winner. Most people have smartphones with them and it is simple for them to connect to the screen [2]. BioSpil is an innovative, modern way of utilizing second-screen viewing to develop a game, while improving an unsatisfactory situation.

Figure 3.4: Showing the spectators in a cinema connecting their smartphones to the screen.

3.4.2 Micro-games

WarioWare

WarioWare is played on Nintendo DS and was published in 2010. It is only designed for one player, but features a main concept of microgames. Microgames in WarioWare can be categorized as bite-sized mini-games, that are shorter, simpler and have clear conditions. The WarioWare-series consist of different games, for example WarioWare: D.I.Y. where the players can design microgames and upload them [15]. The fact that microgames are shorter and simpler means they are better fitted to be combined with multi-screen gaming. Both in situations where players have to be aware of both screens at the same time and when minigames would have taken longer and more resources to complete.

Move or Die

Move or Die is fast paced and for up to four players, both local and online. The goal of the game is to stay alive for as long as possible, if you stand still you die, and thus you need to keep moving (Move or Die). The game is ever changing, shifting the mechanics every 20 seconds so that the players are constantly on edge. The characters in the game are unique and can be leveled up in different ways during the game. Move or Die is for PC, and players can pick it up instantly and play it together with friends. Furthermore, the game developers encourage players to create their own content as well [17]. It does not require much tutorial, and despite it being simple at first, it constantly changes to keep the players engaged.

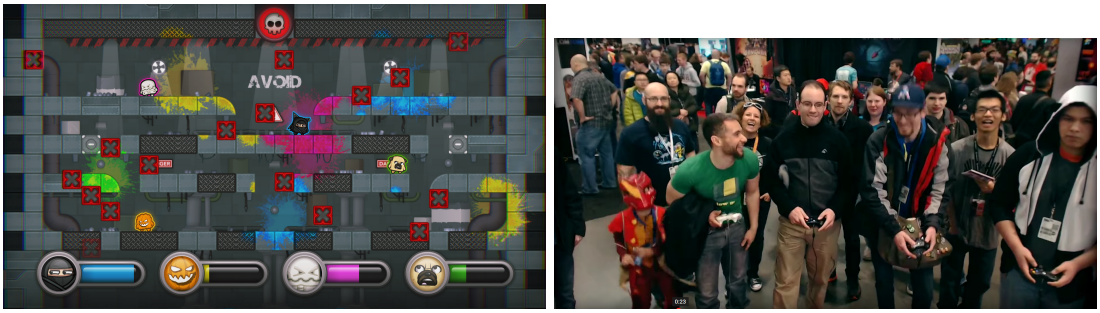


Figure 3.5: A display of Move or Die and players testing it.

Chapter 4

Final Problem Statement

In this chapter, the final problem statement and its research question will be derived. This will be followed up with a list of this project's success criteria.

4.1 Final Problem Statement

There is un-utilized potential in the existing AirConsole games and how they make use of the multi-screen aspect. Most of the existing games use smartphones as simple controllers where the players never have to look down or change their focus, which to this project's research and knowledge is not utilizing AirConsole's potential and strengths. According to the current State of the Art and the conducted Game Research multi-screen games face unique design possibilities, for example using the second display to play micro-games, showcasing secondary visuals while playing the main game or giving the user specific feedback.

Second-screen viewing has greatly increased over the years. This, combined with the fact that mobile devices are constantly growing in power and usability, puts AirConsole's platform in a situation where new designs, concepts and experiments could be highly beneficial. The platform is also unique in the way it uses easy-accessible software and hardware most users already have access to. Challenges could be how to make the second display more intuitive towards the players and how to implement natural transitions between main- and second-screen viewing, while combining this with a local-multiplayer game based on interaction in real life and cooperative play. Finally, the process towards developing a game utilizing AirConsole's potentials could be to conduct an experiment on user's focus while playing the game:

Can a mechanic in a game using multiple screens requiring the player's attention, help the player reduce reaction time when a transition between main- and second-screen viewing is required?

With this problem statement, the goal is to investigate how a game can help the player changing focus when relevant, by effectively reducing that player's reaction time when a transition from one screen to another is required. Specifically, this project aims to investigate if implementing visual cues in the game, can help direct attention to the second screen, and if a difference in reaction time for users can be measured. To test this, several success criteria were set up.

4.2 Success Criteria

- The user has to change their focus from the game screen to the controller screen in order to obtain all necessary information within the game.
- Users playing a version of the game with visual cues should be faster at changing focus between screens when required.
- The controller should not steal attention when there is no relevant information.
- The implemented movement system complements the AirConsole platform in case of any delays.
- The game clearly informs the user where to navigate to when prompted.
- The game should include animated 3D-Graphics.

Chapter 5

Design

In this chapter, the process of going from a a concept to an actual game design will be explained. The design phase consists of several important visual aspects of the game, including graphics, level design, controller and user interface. Finally, the audible effects and soundtrack will be described.

5.1 Game Design

According to Tracy Fullerton, a game is defined as: *A closed, formal system that engages players in a structured conflict and resolves its uncertainty in a unequal outcome[10].* Games can come in many different shapes, but are often similar to that of competitive play in general. This is because a game usually provides the player with a winning condition, structured with an artificial conflict and safe boundaries among other elements. In other words, a game ultimately engages the player in a structured form. This form is known from Fullerton as the formal elements, and they are the essence of a game [5]. The subsections below will be used to explain the game design and gameplay principles using Fullerton's theories.

5.1.1 Players

The game will be played through AirConsole, and the players only need a screen and their smartphones to do so. Therefore, the environment and general ambiance at the start of the game will be more casual and laid-back. This also means that the game's world needs to be easy to relate to and invites the four players into a cooperative game of planning and quick action. The setting and the situation is easy to comprehend: There are different roles, but the four players fight, win and lose together. The part generic, part goofy western design does not require much of the players in terms of playing the fantasy of the game. Therefore it can be used as a quick break and nearly every kind of player can participate. The cooperative aspect and the need for talking and planning during the game enables group bonding and the feeling of achieving a common goal.

5.1.2 Objectives

The overall objective of the game is to survive from one train station to the next. Somewhere along the journey the train starts breaking down, which makes the four workstations launch individual events for the players to complete. Failing five of these events will derail the train, and it does not matter which player fails the event. This also means that the individual objectives for each player can be seen as a group objective, distributed between individuals. The health of the train is shared between all players. Each event consists of a smaller objective in completing the presented micro-mechanic on the smartphone.

The objective itself does not only lie in completing the events, but also how to distribute the tasks between the players. Some events will be shared, meaning that everyone can complete them, but others will be specific to a chosen individual. These individual tasks can only be seen by the relevant player, therefore the objective will be more challenging and perhaps impossible for some if they are not communicating together. If every player just goes after their individual objectives it also means that there will be none left to take care of shared events. This element forces the group to interact together as all of the players share a common goal.

5.1.3 Procedures

The players are constantly given actions to do, but they vary between being movement, interaction and micro-mechanics. A player cannot choose to interact with all the workstations at the same time because they are located in different places within the train. The time is limited, and therefore both the time it takes to decide which action to do and to actually walk there is costly for the players. Furthermore, they also have to complete a micro-mechanic for every interaction with a workstation. It is constantly a dilemma-based choice to choose what action to perform, due to both shared and individual events. A player might take a shared event in one side of the train, but meanwhile there is a individual event for him at the other side that his teammates cannot complete. These procedures create interactive and social gameplay.

5.1.4 Rules

The rules are explained by a visual tutorial at the beginning of the game, but are also implicit and implied through the designed visuals. For example, the workstations are familiar real-world objects that the players can interact with and are shown by highlighting them when an event becomes available. When being in reach of the workstation an icon with an ”A” will pop up. This illustrates that the players can press the ”A” button on their smartphone, and that it will work when being within the zone. Pressing wrong buttons when being in a micro-mechanic or failing to press any buttons will fail the event, and this is visualized by a timer counting down. If the player happens to fail an event, it will be visualized through the HUD. This also reminds the players how many events they can fail before the game is lost. The players cannot leave the train which is implied by the walls and the train moving.

5.1.5 Resources

There are two main resources for the players to be aware of, this being health and time. The players have five lives shared between them, and one life is lost every time one player fails to

complete a micro-game. There is no way to regain lives, and once all five lives are lost the game is lost.

The other resource is time and the players does not have direct control of it. There are two different time resources, the first one being the actual time it takes from the train to reach another station. This is invisible to the players, but it takes five minutes for the train to reach its destination. The events, both shared and individual, are also timed and a countdown timer is shown above the workstation with that event. Within this time limit the player have to reach the workstation, activate it and complete the following micro-mechanic.

5.1.6 Conflicts

In order for a game to be engaging for the player, there must be one or more conflicts that the player is presented with. These conflicts are what drives the game forward, and they are also usually what drives the narrative of the game. Within the game there are one major conflict, and several minor conflicts. The major conflict in the game is keeping the train running without stopping or being destroyed between stations. This conflict is fueled by many other minor conflicts, represented in the form of malfunctioning workstations needing to be fixed within a time limit or else they are going to inflict damage on the train.

The other players are not opponents, but might end up being obstacles when everyone is running from workstation to workstation. Therefore, conflicts in the form of dilemmas might arise between the players themselves. This also create mental obstacles perhaps inciting some sort of competition because one player might think one of the shared events is more important than his own individual one. It could also be that one player is not fast enough with a shared event which might stress another player waiting to complete his own individual one, or perhaps he has another event waiting for him in the other wagon. The players cannot ignore the mental conflicts because they need to work together to win, forcing them to employ a range of social skills as well as individual decisiveness and multitasking.

5.1.7 Boundaries

The physical boundaries are the train itself. The players can never exit the train and can only act inside the two wagons visible in the camera. They also have to walk through the doors and cannot go through the walls inside the train. The only way to get from wagon to wagon is to go through the passage in the middle section. There is also the emotional boundary of separating the game from the real life. The interaction of the players in-game might lead to discussions or different opinions on the executed strategy. Therefore, it is also perfect for a group of friends who can really push these boundaries because they already know each other. They are then able to joke around and enjoy multitasking together as a team. Generally, the players achieve wins or losses together in the game, and it should therefore not inflict damage on their relationship. In fact, the feeling of competing in union could potentially make the players feel closer than they were before.

5.1.8 Outcome

There are two outcomes in the game; either the train derails before reaching the station or else it arrives safely at its destination. The first outcome means that all players lose no matter how

many events they have completed during the game. The second outcome on the other hand means that every player wins, even if one of the four players lost four events and the other players did not lose a single one. The players always share the same outcome. Winning the game is always determined after five minutes, but losing the game can be determined at any given moment during the game.

5.2 Graphics

This project decided on a graphic-style based on boxes and box-figures, almost Minecraft-like characters. This was to give the game an unrealistic and goofy style, which was suitable because it is supposed to be easy to jump into and takes little time to finish. The fun graphical style entertains the players and bring them together in this short teamwork game. The graphical inspiration was originally picked up from the danish game Stikbold. Stikbold is a sports game where the players compete in a variation of dodgeball with fast-paced and often unfair gameplay. The characters in Stikbold are also box-figures and designed to be very quirky and entertaining [6]. A full collection of the final game art can be found in portfolio section 10.2.

Figure 5.1: Stikbold is developed by the danish company GameSwing [7].

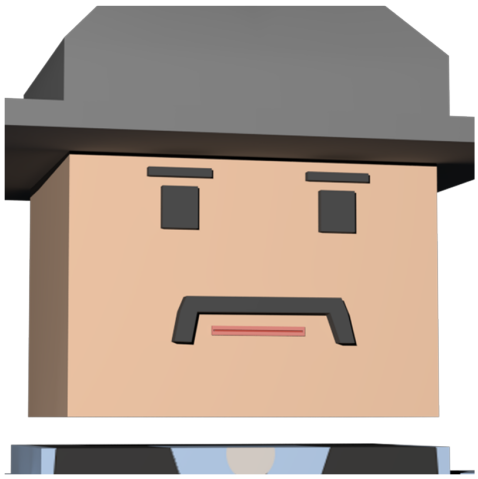


Figure 5.2: An example of the graphic choice inspired by Stikbold.

5.3 Level Design

The level design was created to direct players with visual cues, keeping them on their toes constantly while demanding team coordination and planning. This was achieved with seeing the train from a sloping angle, as well as making the players capable of seeing through the outer walls. From this point of view, the players are able to both see each other, the obstacles and highlighted workstations, but still have to navigate correctly to get through the narrow rooms. This may seem easy if you have a lot of time, but once the workstations begin breaking down, the four players will have to talk and plan together to manage the tasks.

The narrow rooms are both authentic towards a realistic train map, but also functions as a second challenge. Not seeing the level map from above means it is more difficult to notice the walls, which again adds to the stress levels because the players are constantly on their feet. Players might also bump together as they haste from one workstation to another. The level design complements how the overall game mechanic works, because in this game you cannot camp by one workstation. The game will launch events that only a certain character can complete, and therefore one player might suddenly be needed in the opposite side of the train. The map should be seen as a challenge, not a nuisance. Therefore, the active workstations are briefly highlighted when events begin, and an icon with an exclamation mark will make sure the player notices when a new task is available to him. The workshops are placed in their realistic areas and generally the train is divided into rooms and sections designed to make the level map more intuitive.

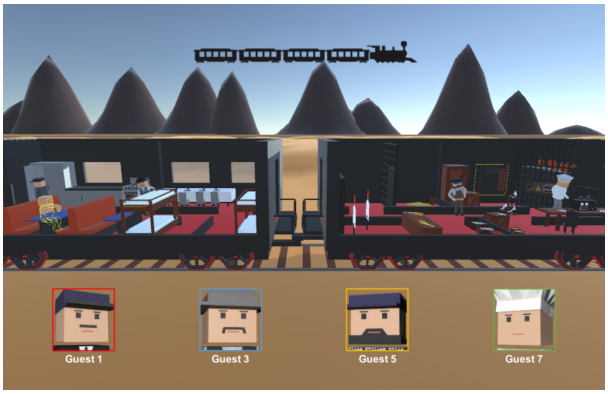


Figure 5.3: Shows the design level seen from the player's angle and the highlighted workstations: Passenger seat, stove, shooting range and pipebox.

5.4 Controller

A central part of AirConsole's project proposal was the concept of using smartphones as the controller. The company already had experience on this topic, regarding its benefits and issues. The mayor problem being the inherent delay created between the main screen and the smartphones. This latency was not a big issue in the slow-paced or turn-based games, but could frustrate the player when facing fast-paced games. This was especially true in games where timing was important or when navigating the controls precisely was necessary.

This project quickly took focus in developing an initial design for the controls. The measurements of the screen depended largely on the individual smartphones and it varied a lot between the many brands and series. Therefore, it was an iterative process to design controls that would fit on any smartphone screen. Using the controls should always be intuitive for the player which meant the buttons and interactive zones could not be to close to one another. The early design-phase was followed up with a mental model elicitation, investigating the current controls at the time (see portfolio 10.9.2).

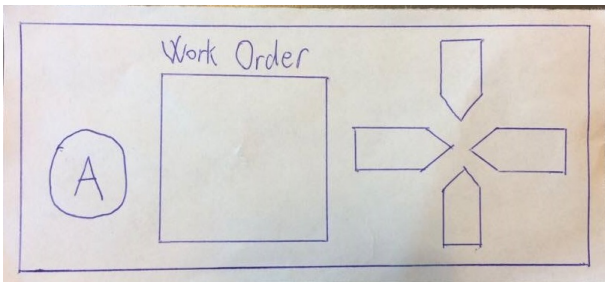


Figure 5.4: The controller design used in the mental model elicitation.

The directional buttons were on the right side and the a-button on the left side. This was changed shortly hereafter, as it was more intuitive for players to control the movement from the left side of the controller. In all probability, this was due to the already implemented controllers used by Playstation, X-Box and Nintendo being designed this way. Players were used to such a control scheme.

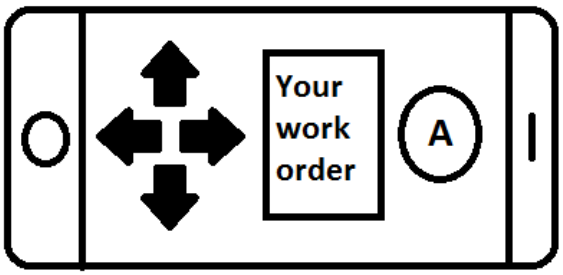


Figure 5.5: Early controller design used in surveys.

Later in the process it was concluded that the current directional buttons were not complementing the TV/mobile multi-screen concept. Participants using these buttons had to constantly look down to ensure they were pressing the intended ones. This caused issues because the experiment was to improve reaction time between the screens by visual cues. If the players already had to prematurely look down, the controller design was ineffective. Therefore the directional buttons were changed to a swipe-pad area where players would not have to look down. Once their finger was placed they could just swipe it around to move their character.

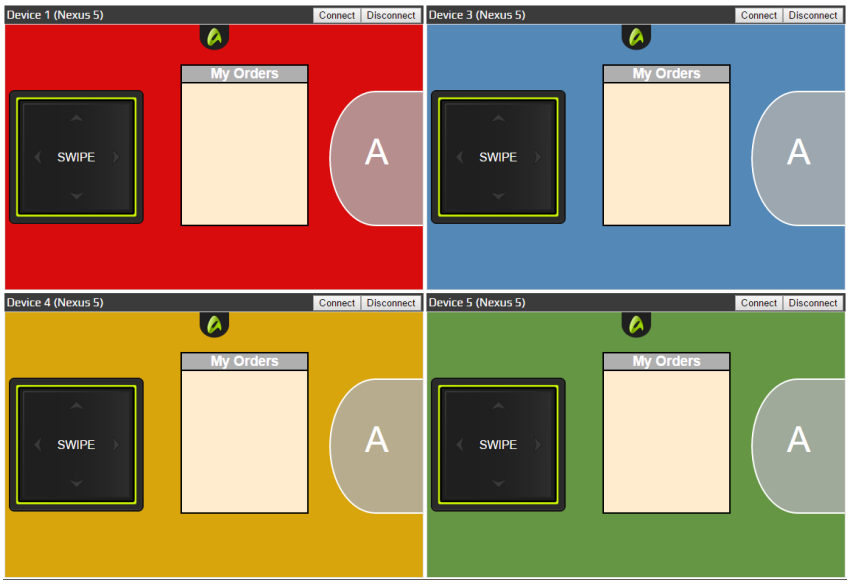


Figure 5.6: End result of the controller design.

As seen above, the final design included a swipe-pad to the left, a work-order list in the middle and a A-button to the right. The swipe-pad is used to move the character, while the a-button activated a workstation once the player is close enough. The individual tasks are hidden from the other players. This is a design choice based primarily on the idea that it is a cooperative game, and thereby forcing the players to share information with each other. Furthermore, including everyone's individual events on every work-order list would be a very cluttered design.

5.5 User Interface

The players are able to interact with the game and all its stages through their controller. The different stages of the user interface (UI) are the ready-check, the game on the main screen and the micro-games on the smartphone-screen.

Figure 5.7: The initial ready-check stage activated when the game is launched.

After finishing the ready-check the level design will appear on the main screen. The players navigate through the train until they activate a workstation by pressing the A-button. After doing this, a micro-mechanic (see portfolio section 10.5) will appear on their smartphone and they will have to look away from the main screen. The players can never physically interact with other elements than the swipe-pad and A-button, but information throughout the game is showed on both screens. There are visual cues like highlights and pop-up icons on the screen to constantly tell the user where new information is given. This is supposed to guide the player's attention and indicate what actions he can perform.

5.5.1 Heads Up Display

Part of the information giving by the UI is through the Heads Up Display (HUD).

Figure 5.8: The HUD displaying how many micro-mechanics the players can fail before the train derails and the game ends. Each respective wagon and the locomotive itself counts as one failed micro-mechanic when removed.

Figure 5.9: The HUD displaying the four characters, their given colors and underneath the tag that each player is registered as.

5.6 Sound

The background music were chosen to complement the goofy graphics, but the noise of a moving steam locomotive were added to ensure authenticity. Even though the game is short and supposed to be fun, the sounds should complement the actual feeling of being on a train. There are also in-game sounds for interacting with the workstations, the pop-up icons, the completion of an event and the failure of an event. These sounds are commonly used and are meant as audio-feedbacks to give the players more information.

Chapter 6

Implementation

In this chapter, the key elements that were implemented to test the hypothesis will be explained. First an explanation of the platform which the game is built on, and built in, then an explanation of how the game utilizes the multi-screen setup and lastly how the games visual cues work.

6.1 AirConsole

AirConsole is a platform with an API that allows for easy communication between a screen device and multiple controller devices. It does this by creating sessions on their servers, allowing the controller devices to just put in a session ID, to be redirected to the screen that runs the game.

AirConsole is a web based platform, meaning that for the game to run on their platform it needs to be compiled into two HTML files, one for the screen to run and one for the controller to run. AirConsole will automatically redirect the devices to the right file, if the host location is properly configured. It is worth noting that given it is a web based platform the game is restricted to using the WebGL API.

6.1.1 Communicating between screen and controller

Utilizing the JSON (**J**ava**S**cript **O**bject **N**otation), in conjunction with the AirConsole call functions, makes it simple to send and receive customized packages of information from screen to controller and vice versa. The information is handled by an "OnMessage(int deviceID, JToken data)" function, where Data is the package of information and the deviceID is the ID of the device that send the message.

An instantiation of AirConsole is then able to parse which deviceID is tied to which active player, allowing for all information parsing, sending and receiving to be done in a single script. For this project all of the message parsing and AirConsole functionality was implemented in a "GameLogic" script that is appended to this report. For more information on how the data is parsed see section 10.3.

6.2 Unity

Unity is a game engine that offers support for multiple platforms, and for this project the focus was on creating a WebGL game to run with the AirConsole platform. Unity natively is able to compile a unity project into a WebGL application which can be hosted on a website, a local server or locally on your computer. Unity offers many prebuild components that highly simplify many timeconsuming tasks, and allows for easy and fast implementation of prototypes. Unity in short is a powerful multipurpose game engine, which perfectly facilitated the projects needs and requirements.

6.2.1 AirConsole Plugin

The AirConsole plugin is described as: "*Unity-Plugin is a C# wrapper for the AirConsole Javascript API. With the plugin, Unity serves you the needed screen.html file inside the Editor and creates a screen.html file after an WebGL build export.*" [13]

Using this plugin the game can be programmed primarily within the C# syntax, yet still works with WebGL. The plugin also comes with a build in websocket-server for communication between the backend AirConsole system and Unity's editor. The project can be tested within Unity's environment before being compiled and hosted.

6.2.2 Functionality and Ease of Access

Creating a game with Unity is heavily based on object oriented design. It is possible to create many different predefined objects, or create some custom ones, though they all have one thing in common: components. Components and object allows for a flexible work environment with effective modularity, creating scripts, colliders, character controllers and many other components, that are independant or reliant on other components.

6.3 AirConsole Controller

The AirConsole platform uses smartphones as the game controller, which allows for innovative designs. The controller is build using HTML, for the controller elements, designed with CSS and altered with JavaScript. The controller is built the same way as a normal website, making it easy to build and design. The AirConsole API makes it easy for the controller to communicate with the game screen. The game is designed with small micro-mechanics that occurs on the controller.

6.3.1 Game Controller

The game controller standard view is split into three section, left, middle and right. The left has a basic 4-way swipe-pad, which allow the users to move around by swiping in the desired direction. The middle section has a box with a title of "My Orders", throughout the game the box will be filled with active task that has to be fixed during the game. The right section has a single button, which allows the player to interact with specific game objects. The interactive

button works by sending information to the screen, the screen checks if there is a gameObject and sends the name of the micro-mechanic at that location to the controller. This will activate the controller overlay which consists of a micro-mechanic. Based on the event name a specific micro-mechanic will be shown and disable the standard controller function. For more details about the controller setup and communication between controller and game screen, see portfolio section 10.4.

6.3.2 Micro-mechanics

A micro-mechanic is a quick and interactive mechanic which requires small amount of understanding and time. This is different from a mini game which requires more understanding and longer gameplay than a micro-mechanic. The mechanics are built in a different section from all the other elements and is first shown when a message is received from the game screen. The mechanics are shown on the controller when specific information is received, based on the event name a specific mechanics is shown, such as a variable with the name "WS01E01" would activate the overlay for micro-mechanic 1. For more information about the micro-mechanics, see portfolio section 10.5.

6.4 Visual Cues

The game includes visual cues for the user to help them guide their focus towards the relevant screen when something has changed. This is used primarily to navigate the user towards the work-order list on the phone whenever it is updated. The visual cue that notifies the user of a new work-order is a pop-up. Another visual cue in the game is the highlight which makes it easier for the user to locate the workstation that has an active event.

6.4.1 Highlights

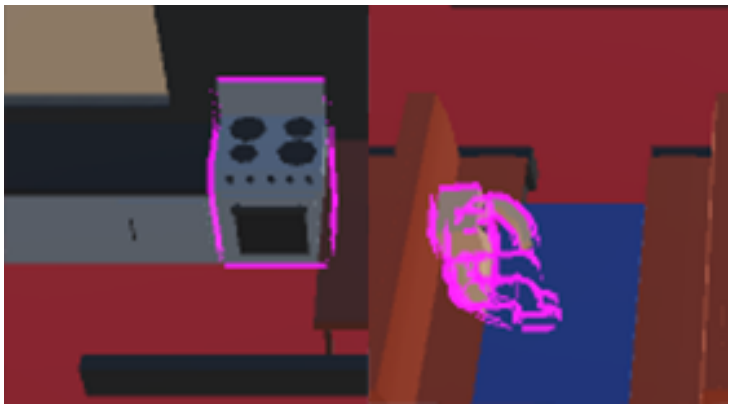


Figure 6.1: The visual cue for a new active event on a workstation.

Highlights appears whenever a new event is active on a given workstation. The highlight pulses slowly once when it becomes active. The color does not change, it is set to be pink to avoid matching any other role color. This cue is not meant to tell the user what event is currently active but rather where it is. Once the state of the workstation changes to inactive, the workstation

will wait to flash the highlight until it becomes active once again. To see how the shader is set up and used in Unity, please refer to section 10.6.3.

6.4.2 Pop-Ups

Figure 6.2: The visual cue for the players to look down on the phone.

One of the pop-ups is an image of a phone with a exclamation mark of the role color as seen in figure 6.2. The role colors are equivalent to the one they are given as they join the game: red, blue, yellow and green. There is also the white exclamation mark that indicates a shared work order appearing on the list. Since the shared ones can be completed by all of them, they all receive this notification. This is different compared to the highlight cue as this is whenever a work-order appears on the list. The pop-up does not have to be active to appear and this cue also only indicates who can do the task, not where. For an explanation as to how these were implemented in Unity, refer to section 10.6.2.

Figure 6.3: The visual cues to help the user locate their character and when to interact.

These pop-ups are meant to help the users locate themselves and navigate the train as well as to interact with the workstations. The first pop-up cues in figure 6.3 are the colored arrows to the left. These are active during the first few seconds of the game to help the user locate their character when first launched. The pop-ups to the right are the A button that tells the user when the character can interact with that workstation. Another pop-up that will appear over the workstation if it is active is the circle that represents the time left to do the active task. The circle will be colored depending on what player role is required to complete the task, just

as the previous pop-up. To see how these cues were made in Unity refer to section 10.6.1 in the portfolio.

6.5 Model Creation and Animations

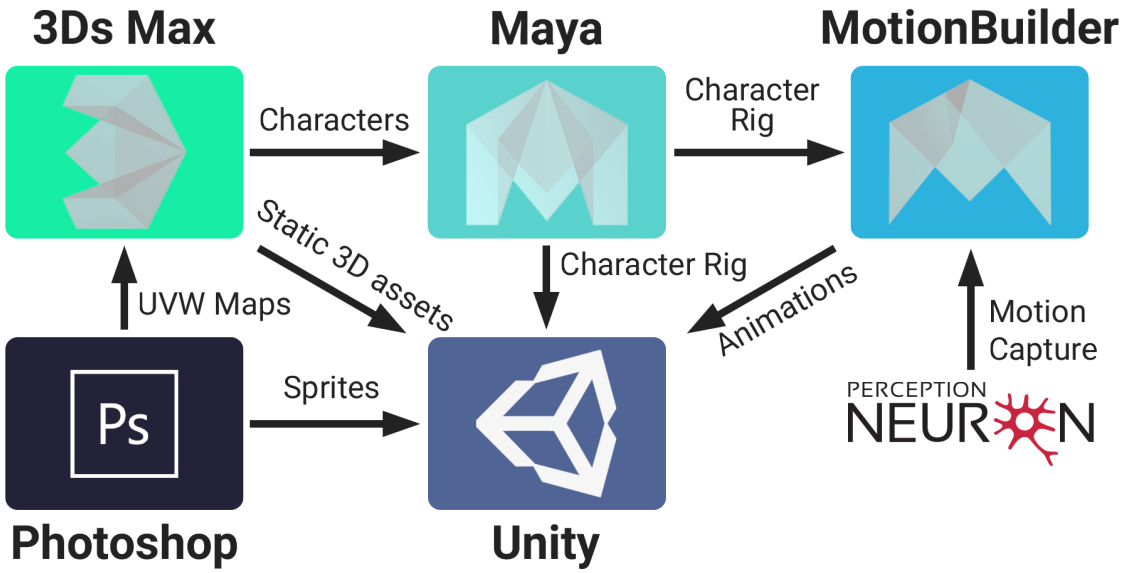


Figure 6.4: A flowchart of how the different assets and animations are made between software.

As seen in figure 6.4, all the 3D assets present in the game are modeled in AutoDesk's 3Ds Max with primarily box modeling, rigged in AutoDesk Maya and partially animated with Perception Neuron using a plugin for AutoDesk MotionBuilder. Perception Neuron is a kit with sensors that is worn as a harness on the body to record your movements, better known as motion capture. The sensors on this kit tracks your joints' position in real time. These animations are applied to a character avatar made in Maya by rigging the model from 3Ds Max. The animation takes are imported separately in Unity and then applied to the character model from Maya which is also imported separately into Unity as well. The raw models of the assets without animations are directly imported from 3Ds max into Unity.

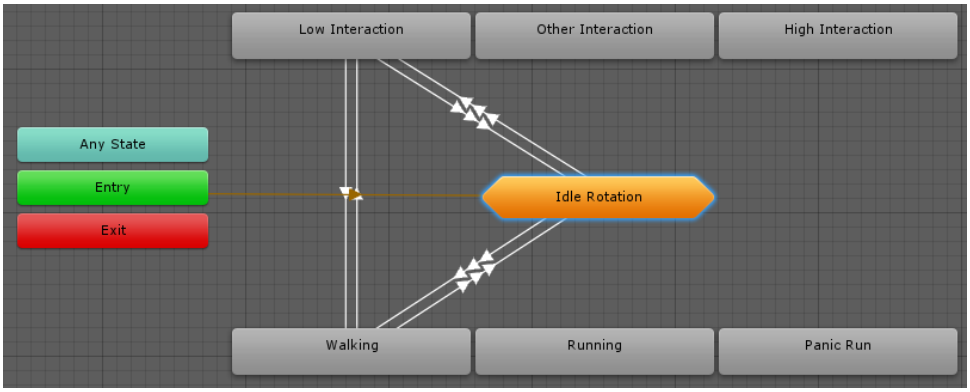


Figure 6.5: The animation controller for the animations in Unity.

Once all the assets are in Unity, each animation take can be applied in Unity using an animation controller. In figure 6.5 is an example of how the base layer of the animation controller looks. The orange layer in the middle is the idle rotation used for the models which consist of six different animations. Transitions between the animations are made when called in the script. Section 10.7 includes further explanation as to how the 3D assets were made, rigged, animated and implemented.

6.6 Eventsystem Structure and Handling

To control when and where the micro mechanics were active, an eventsystem was developed to control the flow of the game, as well as the communication between different workstations and the controller devices. This is achieved using a delegate based event system. A delegate based event system uses a single communication point: the delegate, which other classes can subscribe specific functions to. When a function is subscribed to a delegate based event, said function will always be executed when the event triggers. Using these triggers, multiple classes can be passed the same information, as well as be executed at the same time without being reliant on whether or not they are specifically referenced.

Before detailing what the individual delegate events pass to their subscribers, a quick explanation of the event structure is in order.

6.6.1 Event Struct

When designing a game event system, the data explaining how, when and what of each event consists of needs to be passed to the relevant components. For this, two prominent solutions are available, passing a reference to an instantiation of a class or passing a struct. The major difference is that a class is a reference type, and a struct is a value type. When passing a value type object, both the sender and receiver will end up having an independent copy of the variable, where as passing a reference type, only the original sender will be in possession of the actual variable. Here the receiver is in possession of a reference to the original variable.

Given that the events that are being passed in our game are short lived and are created and subsequently removed, using a struct is the most effective choice. The struct then contains the relevant information needed to execute, track and initialize each event.

For this system, the following four delegate events have been implemented:

- **Start Event.** This passes a new event struct to the subscribed classes indicating a new event has started.
- **Stop Event.** This passes a struct similar to the Start Event, but also a Boolean value that allows the subscribed classes to know whether it was a failed or completed.
- **Start Game.** This delegate event lets its subscribers know that the ready check has been completed and the game has begun.
- **Stop Game.** This delegate event lets the subscribing classes know that the game has ended, whether it be completed or failed.

6.7 Testing and Collecting Data

To efficiently collect data with as little human errors as possible, data collecting class was implemented. The class named DebugToCsv was implemented to write a CSV file with the relevant information, whenever an action relevant to the hypothesis testing occurred. To be able to sort the information for further analysis and evaluation, each event had four associated data points: session time, date and time, player ID and Session ID.

The following events were tracked and written into the CSV files: when an event indicator flashed above a characters head indicating that a new event had been added to the work-order and when a workstation was highlighted upon an event being active.

Chapter 7

Evaluation

7.1 Hypothesis

In order to evaluate the problem formulation, a hypothesis must first be made. Looking at the problem statement: *"Can a mechanic in a game using multiple screens requiring the player's attention, help the player reduce reaction time when a transition between main- and second-screen viewing is required?"* The problem statement centers around improving the reaction time of swapping focus between two screens for the users. To set up an experiment that tests this, a control group experiment would be most suited as the difference in reaction time between users with and without the help of visual cues can be measured. In accordance with the problem statement, a one sided hypothesis can be made, as we assume that users with visual cues react faster than users without. Since it is more scientifically correct to try to falsify a hypothesis, the null hypothesis should be one that can be falsified, where the alternative will correspond to the problem statement. Thus, a null-hypothesis can be formulated:

$$\mathrm{H}_0 : \mu_{\mathrm{ReactionCues}} \geq \mu_{\mathrm{ReactionControl}}$$

As can be seen, the null hypothesis assumes that users with visual cues (treatment group) have an equal or higher reaction time than regular users (control group). The goal is to falsify this hypothesis, as the alternative hypothesis would then be true:

$$\mathrm{H}_A : \mu_{\mathrm{ReactionCues}} < \mu_{\mathrm{ReactionControl}}$$

Stating that the average reaction time of the treatment group is lower than that of the control group, confirming what the project set out to investigate.

7.2 Design of Experiment

To test the hypothesis, a control group experiment was designed. In this experiment the treatment group plays a version of the game where visual cues are implemented to help guide the player's focus between the two screens. The control group plays a version of the game without any visual cues. Thus, the only difference between the treatment and control group is whether the two types of visual cues are implemented (Highlights and Pop-ups as described in section 6.4), with no changes in actual gameplay. To ensure that the participants are not biased

or actively seeking the elements they are testing, they are not made aware of either the presence or lack of visual cues.

As a group of participants play the game, a camera facing the front of the participants records the session, making it possible to track which screen each of the participants are focusing on at any time. At the same time, the game records time stamps for all events when they become active (saved in a CSV file), making it easier to calculate the reaction time from when an event occurs on the main screen until a player shifts focus to the mobile screen. After recording and calculating the reaction time for all events for all participants, the values of the participants' reaction times can be used to test the hypothesis using ANOVA (done in R).

7.3 Setup

For this experiment, a setup with a large wall-mounted TV-screen (main screen) was used as seen in figure 7.1. The participants were placed on a row at the end of a table set facing the main screen and each participant had a smartphone available for use in junction with the main screen. A camera was placed in front of the participants and filming their faces during the experiment.

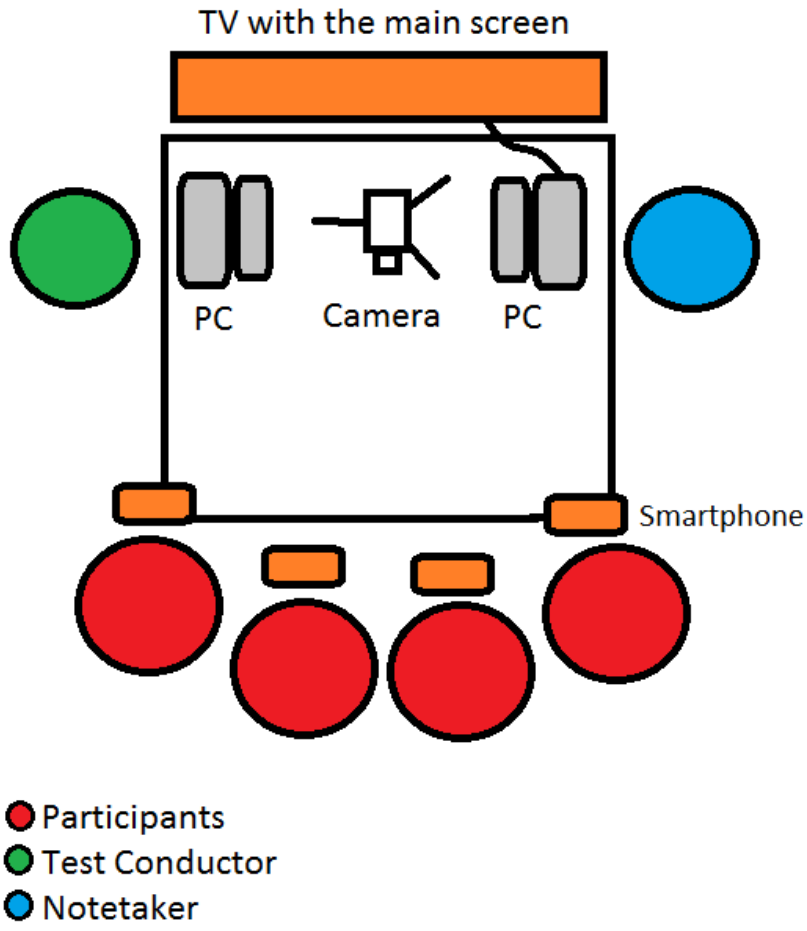


Figure 7.1: An illustration of the setup used to evaluate our system.

7.4 Procedure

Each session of testing with the experiment followed the same procedure to ensure validity and reliability of the experiment. The procedure is as follows:

1. At first the participants are welcomed and asked to sign a consent form, giving permission to record the session on video as well as giving permission to publish the material for scientific purposes. For the consent form, see 10.9.1.

2. Each participant then logs into the game on their smartphone through the AirConsole app, using the code supplied on the main screen. For this test, participants were also asked to log onto a hotspot provided by the facilitators to reduce latency.

3. Having ensured that all participants are logged on to the hotspot and connected to the game (this can be seen when their smartphone has the controller layout in one of the four player colors), the participants gets a short introduction to the game and how to play it. (for more see 10.9.5)

4. After the explanation, the participants proceed to play the first round of the game. This can last up to 5 minutes (after which they win the round) or until they fail to complete five micro-mechanics and lose the round.

5. Once the first round is completed, if the participants where not sure of where the workstations were placed in the train, this would be explained to them.

6. Afterwards, the participants would continue to play the second round of the game (the same level again).

7. After having completed the second round of the game, the participants are presented with a short questionnaire of six questions, concerning the game. Once the participants have finished filling out the questionnaire, the test is concluded.(for more see figure 10.62)

7.5 Results

As mentioned in section 7.2 the data from the tests were recorded on camera (the focus of the participants) and stored in a CSV file (timestamps of events in-game) by a debugger. The debugger would indicate when a new visual cue was activated in the game, together with a time-stamp and a playerID. This would indicate which player the visual cue was meant for and when it would appear. To prepare the data for analysis and presentation, the response times of the participants for each of the timestamped events from the video material were recorded in the CSV file with their corresponding events. From this, the reaction times of the participants could be calculated.

Test Group 1	response time =	time - start time
Participant 1	Green (4)	
Event Time	Look time	Response Time
5	7	2
35	36	1
45	47	2
65	65	0
80	83	3
85	98	13
95	98	3
100	104	4
Game Ends 111		

Figure 7.2: A table of the results from one participant during one session. Event Time indicates when a visual cue was activated, Look Time is the closest time a participant looks down on their phone since the event occurred, and Response Time is calculated by subtracting Event Time from Look Time.

7.6 Analysis of Results

With the data being split into a control group and treatment group, each group had consisted of five sessions, with four people who played the game twice. The relevant thing to look at is the mean and variance of the data. An example of how the data is processed for game one would be; taking the response time for each of the 20 participants in the treatment group. It is important this is done in order from the first to the last participant in game one. Calculating the mean and variance for the collected response time. The mean gives an understanding of how fast the average player understand the games visual cues being tested. Plotting this into a graph, the x-axis shows the index number and the y-axis is the calculated result.

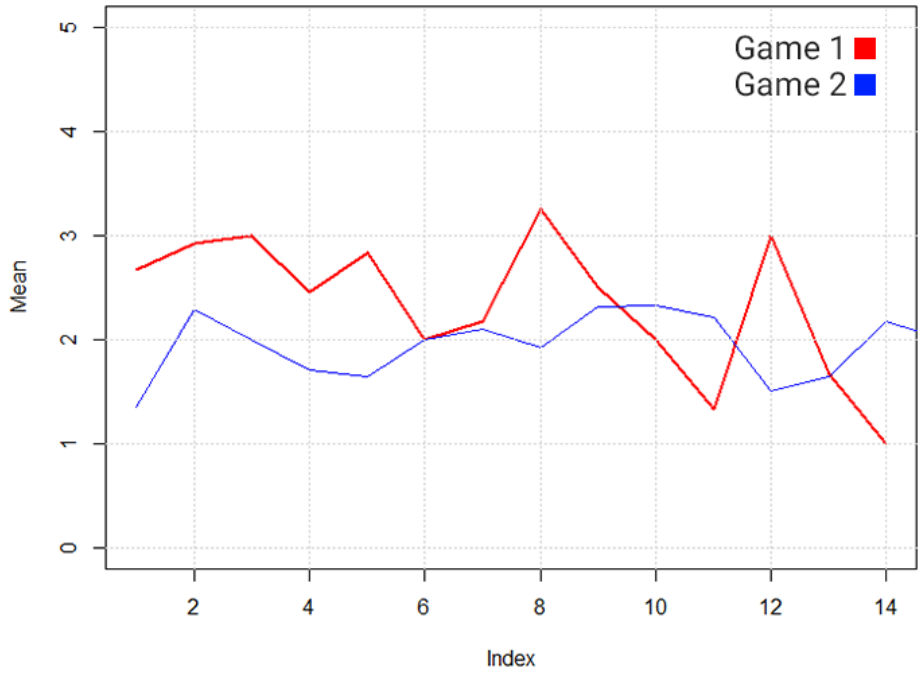


Figure 7.3: Comparing the mean values of game 1 and 2 from the treatment group.

Figure 7.3 shows the mean results from game one and two. The graph shows slight improvement from game one into game two.

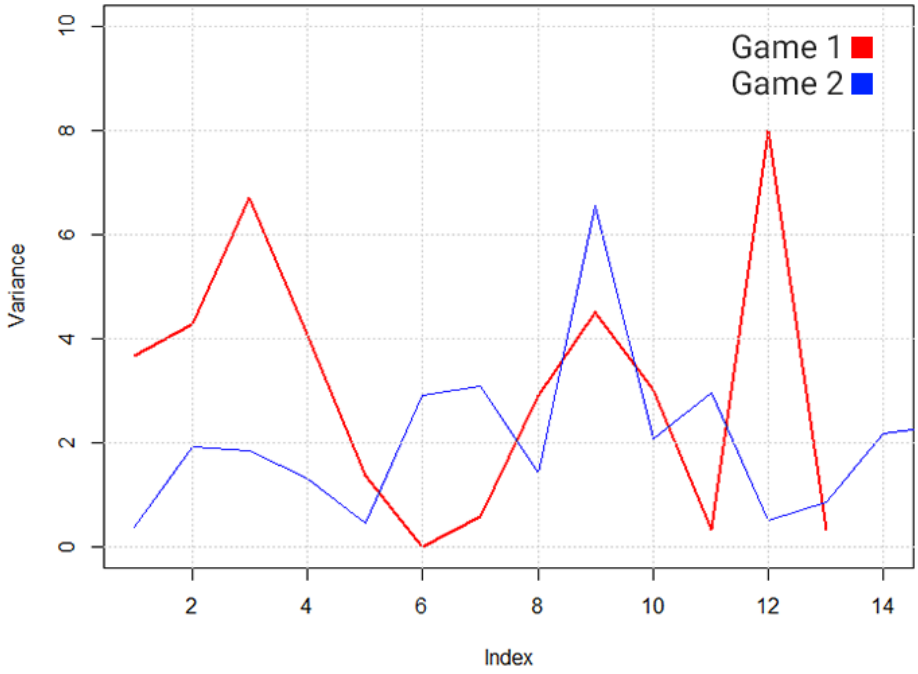


Figure 7.4: Comparing the variance values of game 1 and 2 from the treatment group.

The variance gives an idea of how fast the participants understand the game. Figure 7.4 gives a different look at the data. The graph shows a lot of spikes and comparing the two games, there is a difference: the red starts high with big spikes but the blue is lower in general. Again, the variance shows improvement from game one into game two, as people get more experience.

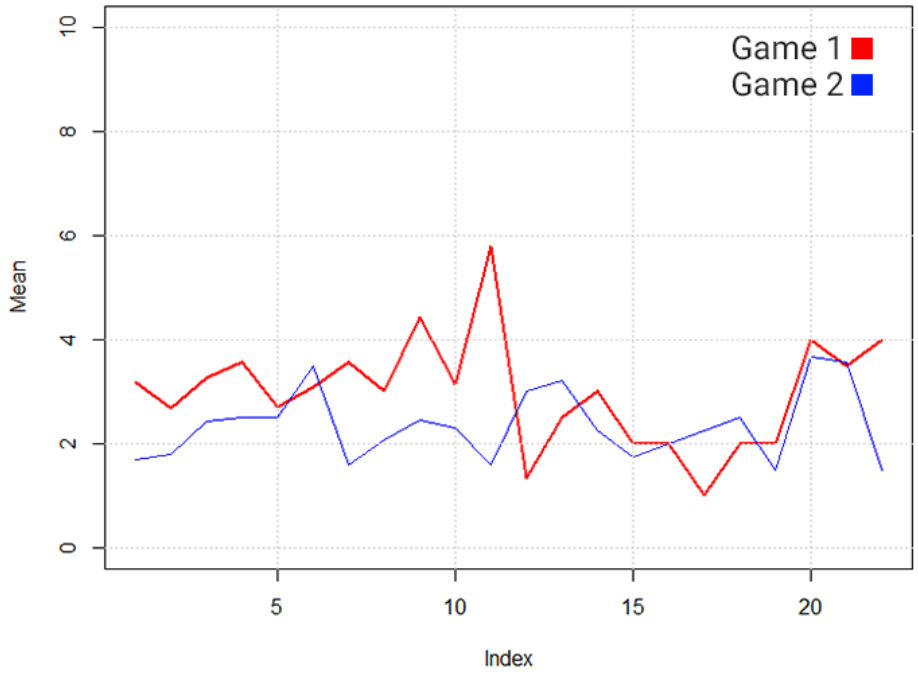


Figure 7.5: Comparing the mean values of game 1 and 2 from the control group.

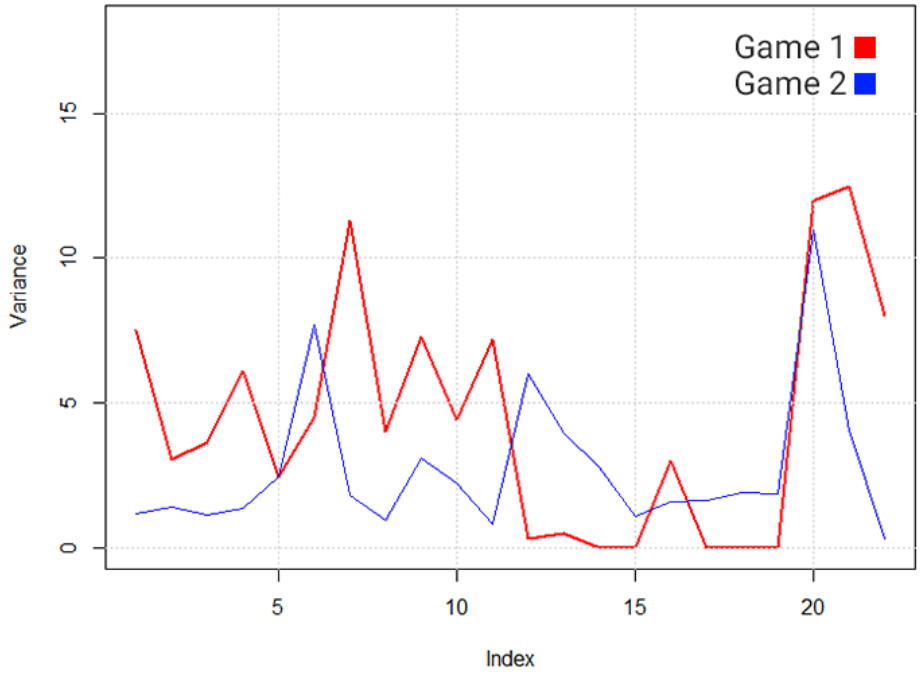


Figure 7.6: Comparing the variance values of game 1 and 2 from the control group.

The same procedure was done with the control group data, resulting in the graphs shown in figure 7.5 and figure 7.6 . Given there was more data from the control group, the graphs are closer to a smooth curve.

Both the treatment and control group showed improvement over the time they played the game. However the treatment group showed better results than the control group, based on the graphs processed from the response time data.

7.7 Discussion

According to Analysis of Results in section 7.6 this project is able to falsify the null-hypothesis:

$$\mathrm{H}_0 : \mu_{\mathrm{ReactionCues}} \geq \mu_{\mathrm{ReactionControl}}$$

And therefore also accept the alternative hypothesis:

$$\mathrm{H}_A : \mu_{\mathrm{ReactionCues}} < \mu_{\mathrm{ReactionControl}}$$

This means that the average reaction time of the treatment group would be lower than that of the control group. The players in the treatment group would, according to this analysis, have a reduced reaction time when directed by certain visual cues. This can be concluded due to the players in the control group having a mean value higher than the treatment group. The mean value in this situation reflects on how fast the average player changes focus which in this case should be as low as possible. The variance is how far each player in the index is from the average player. In other words, the variance reflects how close the players are to each others reaction time, and it should also be as close to zero as possible. Otherwise it means that not

everyone is reacting as fast as the average player or that some of the players did not understand the visual cues. The variance would rarely be zero or close to zero at the beginning, but ideally should be getting closer to zero during the games. When looking at the graphs from section 7.6, the variances from the control group have more spikes and are higher than the variances from the treatment group.

These findings should bring the project closer to confirming and answering the final research question, confirming that a multi-screen game could reduce players reaction time with the implementation of visual cues. However, it cannot scientifically be proven that the difference is significant enough. Nor can it be precluded that unknown factors could have contributed to the results. This could primarily be because of a small sample size and inefficient evaluation designs. Many unusable data points were excluded from the results because of players already looking at their smartphone the moment when the reaction time was measured. Furthermore, most of the first indexes had a lot of usable inputs, but as the games went on some of the groups lost while others continued playing. This left the last indexes with fewer inputs to calculate from, which reduces the value of the results.

The findings could be seen as a step in the correct direction, hinting that the reaction time theoretically could be reduced and that visual cues might be a tool to achieve the intended effect. Further investigation and evaluations could be conducted with similar hypotheses to expand the findings and validate the current results. In such experiments, it would be important to randomize the participants to avoid impacting factors like game experience.

7.8 Analysis of Method

The evaluation was conducted with a control and a treatment group, each group consisting of 20 participants. Each individual evaluation was carried out with four participants at a time, with the participants consisting of students from Aalborg University. The reliability is important so that the gathered results, and its scientific process, can be repeated by others. Also, this ensures that the findings of the project is not a one-off finding. Others should be able to replicate this evaluation, due to the written procedures and questionnaires, and should get similar results in doing so. This said, they might get better results or findings due to inconsistent design and execution in this evaluation.

Issues in the evaluation design are factors that could eventually have been predicted and prevented, such as the game length depending on whether the participants lost or won. As a result some participants ended up playing the game over a longer duration than others and got more experienced in the game. Also, some participants lost faster than others. Furthermore, some participants were already looking at their smartphone when the visual cue appeared on the main display. This made a lot of the data irrelevant as the reaction time could not be found in such cases. The evaluation ended with a small sample size due to the removal of the irrelevant data.

The execution caused issues because the participants were not randomized, increasing the chances that there could have been an unknown factor affecting the results. The results clearly states a difference in reaction time between the control and treatment group, but it affects the validity negatively that it cannot clearly be determined if a possible unknown factor is the reason, or that an unknown factor could at least have had an influence. Such an unknown factor could for instance have been some of the players being more experienced than the others. It could be stereotyped that the Medialogy students were more experienced in games than

participants from other studies.

The method of having a control and a treatment group should heighten the external validity, but in this case not-fully-randomized participants and a small sample size are compromising the internal validity. No methods can be completely successful, but the current design and execution opens up for possible unknown factors as well as inconsistent design affecting the reliability and validity.

Chapter 8

Discussion

8.1 Quality of Solution

The chosen game design of a western theme and box-like graphics mixed with multi-screen aspects and micro-mechanics were well received by the participants, engaging them in a co-operative and real life interactive game. The game itself as well as the required software and hardware is easy to set up and can be played without requiring much dedication from the players. Therefore, it is also entertainment that potentially can be used in many different situations and contexts, for example during breaks, by a group of friends hanging out or by people trying to get to know each other. After the evaluation a few new implementations were made to the game. This included new coding compatible with iPhones, new HUD-animations and a tutorial at the beginning of the game. When the train took damage it would now be visualized by one of the wagons on the HUD falling down and the camera being shook. The tutorial was implemented due to participants at the evaluation being thrown into the game to fast and without any introductory help.

The highlighted workstations did not work as expected, primarily because the participants often were focused on the smartphones when the highlighting appeared. The workstations are only highlighted once for every active event, and it was therefore easy to miss by the participants. This could also happen if all four players were inside one wagon and the camera angle excluded the other wagon. The pipebox workstation, and to a lesser extend the shooting range as well, were also difficult to detect. The stove and passenger seat were more iconic and participants had an easier time navigating to these workstations. Participants were fast to guess where the stove and passenger seat should be located inside the train, whereas when reading ”shooting range” and ”pipebox” it seemed harder for them to estimate the location.

8.1.1 Success Criteria

In general, ineffective experimental design and the lack of randomization compromises the ability to validate success criteria. The findings can therefore be related to the criteria, but in most cases not be concluded with a definitive answer. It could be beneficial for the project to conduct new experiments with improved methods and designs to test some of the current as well as new criteria. This said, the design- and implementation process were still carried out with focus on the success criteria and with frequent advice from AirConsole on how to fulfill them.

It was shown in the evaluation that users *changed their focus between the displays as a way of obtaining the game's information*, following the visual cues giving to them on both screens. Likewise, the success criteria regarding the *inclusion of animated 3D-graphics* is achieved through the animated characters implemented in Unity (see section 10.7). The evaluation indicated that *users playing with visual cues were faster at changing focus between screens* than the users with none, though the scientific proof is lacking significance due to lack reliability and validity. Whether *the game clearly informs users where to navigate when prompted* lacks scientific proof in the same way and is therefore partially achieved. The highlight-implementations did not affect the participants as planned and two of the workstations were difficult to detect.

The criteria that *the controller should not steal attention when there is no relevant information* was partially achieved. It was implemented specifically to prevent this with a intuitive design and no attention-demanding visuals happening while the user's focus were supposed to be at the main display. It was designed this way, but it cannot be entirely excluded that it does not draw attention. Similarly, *the implemented movement system should complement the AirConsole platform* and were implemented to do so by using the AirConsole API's build-in codebase. However, the currently implemented movement system does not function well with the variation of delays, that network based application sometimes produced.

8.2 Wider Context

Multi-screen gaming

Working in a multi-screen environment has both advantages and disadvantages. It allows for new ways to use the devices we all carry around in an interactive manner, however it does also require more awareness from the individual user.

Within the AirConsole platform, there are many opportunities to make use of the multi-screen environment, however there are some restrictions as well. Looking at the opportunities, it is possible to create some unique scenarios that are not possible in standard local multiplayer setups. Since the secondary screen (the smartphone) can be used to send unique information to a specific player, that no other player has access to (assuming they do not look at each others' screens), it is possible to create scenarios that either require a lot of cooperation to solve as a team. Each player only has part of the total information (as in Derailed), or the opposite scenario where players work against each other, using information that only they have access to through their secondary screen. This allows for entirely new fields of development for local multiplayer games, where Derailed explores just one of the many new opportunities. However, many of the games found on AirConsole today, do not utilize the ability to give specific players unique information through the secondary screen. Instead, they merely use it as an ordinary controller with buttons for movement and/or interaction, leaving much of the potential of a platform like Airconsole un-utilized. Beyond giving different players unique information, the multi-screen environment can also be used for individual assignments like micro-mechanics or mini games, without taking up room on the main screen that everyone needs to pay attention to, which allows for less visual clutter. Furthermore, having two screens that require attention can add an element of stress to the game, which depending on how it is handled, can be good as well as bad. In Derailed this stress concept was tested to some degree of success, with the conclusion that whilst it does add an element of cooperation and management to the game, the information received on either display must not in any way be confusing for the players as it causes frustrations.

Concerning the technical aspects of the platform that AirConsole provides, it presents developers with some challenges that needs to be considered when designing and implementing a game. First of, the connection between the secondary screen of the smartphone and the main screen can experience high amounts of latency depending on which network is used for connecting the smartphones. During this project, the general experience was that it gives a smoother connection if using a hotspot rather than connecting over Wi-Fi or mobile networks. However, even when using a hotspot, players can still experience high latency from time to time, and setting up a hotspot might not always be possible. Having a high latency whilst playing can lead to some frustrations, as the game reacts slower than the player anticipates after issuing a command. This is especially problematic in games that require some level of precision, such as the free movement system in Derailed, as it becomes difficult to ensure that precision when needed. This is something that is difficult to avoid when working with the AirConsole platform, and as such it would seem better to design around the problem, by replacing the systems requiring precision with other systems that fulfills the same functions, but requires less precision. In the example of the movement system in Derailed, possible solutions could be using a grid based movement system or perhaps even a turn based movement system, as this would put more focus on managing time rather than precision within the game. Likewise, mechanics that requires the player to do something within a strict time limit or timing something perfectly, are best avoided as it will eventually result in frustrations for players experiencing high latency.

Another technical limitation of the Airconsole platform, is the 10 package limit per second. In case the controller sends 10 packages in less than a second (one problem encountered in this project was users changing movement direction rapidly, using up the 10 allowed packages), that controller receives a short cooldown before being able to send packages again. From the user's point of view, this makes the controller unresponsive if they send too many packages too quickly, which is something they are not aware of, thus causing frustrations. As with the latency issue, this is something that should be designed around, by finding alternatives that requires less packages sent. A turn based movement system could possibly prevent the player from reaching the package limit, as they would only be able to send a few commands every now and then. Solutions like this can have other implications on design, but avoiding the package limit is definitely something developers should strive to reach.

Visual cues

In this project, visual cues was used as a way to direct the users attention to the secondary screen when there was relevant information to retrieve. Currently it works through a highlight and a pop-up, but there are many different ways this could be implemented. Another approach that was discussed through this project, was making the visual cues more subtle, such as making the characters sweat as tasks bundled up on the secondary screen, or a workstation gradually break more and more as a timer was about to expire. It could be interesting to see if the manner in which visual cues are implemented changes how players perceive and use them.

Target group

Throughout this project, no specific target group was chosen to be in focus. That however, does not mean that it might not be important to look into how different target groups approach the idea of multi-screen gaming, or how visual cues affect different target groups. Specifically it would be interesting to investigate if this gaming platform works better with different age

groups, as well as investigating if player skill makes the platform appeal more to some groups than others. In this regard, it would also be interesting to investigate if the need for visual cues directing attention is equally relevant for all age groups and player skill groups.

Second screen viewing

With second screen viewing becoming more widespread as a trend, it is only natural to incorporate this behavior in other systems. AirConsole is one attempt at integrating the smartphone as a tool whilst playing games, which generally works well since most people are already comfortable using their smartphone. This could possibly be expanded to other areas as well, and even though this project did not directly explore other ways to integrate the use of smartphones as a secondary screen for other purposes than gaming, the group saw promises that it could work well in other environments.

8.3 Future Work

It was originally planned that the game should have implemented a difficulty scaling, meaning that every time the players would complete a level the next one would get more difficult. This was not implemented due to time constraints, but it would increase the replayability of the game. Likewise, it was planned that the game could be played by any amount of players from one to four. It would have been interesting to observe whether the amount of players could have influenced the impact of visual cues. In combination with this it could be interesting to conduct further experiments on different types of visual cues, also studying how they perform against each other.

The shortcoming regarding the pipebox and shooting range workstation could be solved by making them more visible or designing new workstations to replace them. Designing new, iconic workstations would also make it easier for participants to anticipate their locations from the work-order list alone. Furthermore, a larger variation in workstations could increase the stress-factor and teamwork required to complete a level efficiently, thus adding more depth and replay value to the game. This could also be combined with a larger variation of micro-mechanics.

Improvements to highlight problems could be to implement arrows on the main display directing the players in the right direction, this could help guide players if the camera angle had excluded the active workstation. It could also be visualized above the work-order list on the smartphones. The camera angle could also be changed as it seemed difficult for the participants to detect the walls separating the rooms, but a camera angle from above would instead make it harder to define objects.

Initially it was discussed that the roles could be expanded further than they are in the current implementation. For example, each role could have specific tools available to complete certain tasks and/or multiple events that require two or more specific roles to cooperate. Another idea was to implement environmental effects that would involve every character at specific points, for instance if the train where to run over a bump all players would have to press a button or complete a micro-mechanic to avoid getting knocked down. This could be expanded to many other type of events as well.

The current movement system has some limitations with this type of platform. The package limit for each controller is ten packages per second, meaning that rapid changes in direction

can lead to a lockout on that controller for a short duration. This could be mitigated by implementing a different type of movement system that requires less packages send per second such as a grid-based system. Furthermore, players risk experience high latency which affects the players precision in the game. This could possibly be avoided by implementing systems that require less overall precision.

Chapter 9

Conclusion

Platforms using multi-screen gaming requires players to easily shift focus between screens when needed. Thus, this project set out to investigate if implementing visual cues within a multi-screen gaming platform could improve the process of changing focus between screens. This was tested with a control group experiment, with the goal of measuring users' reaction time when changing focus.

Throughout evaluating the solution, some important lessons about designing experiments and recording data were discovered. First of, ensuring that a balanced sample size is obtained by making all participants play through the exact same amount of content, is important to increase the validity of the results. Secondly, recording when players react to visual cues by changing their focus can be challenging, as you have to ensure that 1) the visual cue can be seen by all players at any time, and 2) that players are not already looking at their secondary screen as that makes it impossible to record a reaction time.

The evaluation carried out in this project suffered somewhat from the aforementioned problems, reducing the validity of the results. This can be fixed by making another round of evaluation, incorporating the lessons learned from the first round. Doing so should not present much of a challenge, as the evaluation had good reliability. Using the results gathered from the evaluation and the overall experiences gained through this project, an answer for the problem statement can be formulated:

Can a mechanic in a game using multiple screens requiring the player's attention, help the player reduce reaction time when a transition between main- and second-screen viewing is required?

Visual cues implemented in a game using multiple screens, seems to reduce the reaction time of players when a transition between main- and second-screen viewing is required. Whilst this cannot be finally concluded with the current data, it does show great promise in helping players manage their focus when using multiple screens.

Going from here, it would be relevant to conduct more evaluation regarding this solution, to ensure the validity of it. Furthermore, it would also be important to investigate how different implementations of visual cues perform against each other, to find the most efficient ways of designing these. Overall, this project should be used as guidelines on how to evaluate the use of visual cues in a multi-screen environment, and inspiration for what areas to research further.

Bibliography

[1] Louay Bassbouss, Max Tritschler, Stephan Steglich, Kiyoshi Tanaka, and Yasuhiko Miyazaki. Towards a Multi-Screen Application Model for the Web. *2013 IEEE 37th Annual Computer Software and Applications Conference Workshops (COMPSACW)*, (July):528–533, 2013.

[2] Biospil. BioSpil – Android-apps på Google Play, 2016.

[3] John Corpuz. 10 Best Game Companion Apps, 2016.

[4] Andrew Duchowski. Eye Tracking Methodology: Theory and Practice - Andrew Duchowski - Google Bøger, 2007.

[5] Tracy Fullerton. *Game Design Workshop: A Playcentric Approach to Creating Innovative Games*. Elsevier Inc., 2nd editio edition, 2008.

[6] Game Swing. Stikbold! - A Dodgeball Adventure, 2016.

[7] Game Swing. STIKBOLD! PRESSKIT, 2016.

[8] Jackbox Games. Quiplash — Jackbox Games, 2015.

[9] Nicholas Katzakis, Masahiro Hori, Kiyoshi Kiyokawa, and Haruo Takemura. Smartphone Game Controller. *Proceedings of the 74th HIS SigVR Workshop*, 2011.

[10] Martin Kraus. SEMESTER LECTURES ON GAME DESIGN AND DEVELOPMENT. 2016.

[11] Microsoft Developer Network. Delegates (C# Programming Guide), 2015.

[12] Rafael Morgan. AirConsole Dev Diary: Our first Local Multiplayer game for AirConsole — AirConsole Blog, 2016.

[13] N-Dream AG. AirConsole - Play multiplayer games together, 2015.

[14] Nintendo. Nintendo 3DS - Official Site - Handheld Video Game System.

[15] Nintendo. WarioWare: D.I.Y. for Nintendo DS - Nintendo Game Details, 2010.

[16] Nintendo WiiU. What is Wii U? - Wii U from Nintendo - Info, Details.

[17] ThoseAwesomeGuys. Move or Die Game — Official Site, 2012.

[18] Ubisoft. ZombiU for Wii U - Nintendo Game Details, 2012.

[19] Anna Van Cauwenberge, Gabi Schaap, and Rob van Roy. "TV no longer commands our full attention": Effects of second-screen viewing and task relevance on cognitive load and learning from news. *Computers in Human Behavior*, 38:100–109, 2014.

[20] Anna Washenko. 8 Great Apps for Second-Screen TV, 2014.

Chapter 10

Portfolio

10.1 Game Concept

Vision statement

Game logline: *Players work together to solve small tasks aboard a train to keep it moving between stations.*

At the moment we call the game Derailed as a placeholder. In Derailed a small group of players (1-4) work together aboard a moving train, to keep the train moving between stations and ensure that nothing goes wrong. As the train moves towards its next destination, minor or major crises will appear, forcing to players to attend to them before a timer runs out. Failure to do so results in penalties either to score or health of the train. Should the health of the train reach 0 before arriving at a new station, the train will derail and the level will be lost. If the players on the other hand manages to keep the train running until the next station the level will be completed, and a score will be given to the team based on their performance. From here, players can advance to the next level with a slightly higher difficulty level.

Mechanics

Derailed has two cores mechanics: *Tasks that makes up the crises that the players has to attend to, and player roles, restricting the players from doing specific tasks.*

The tasks are micro mechanics that the players has to solve within a set timer. In order to solve a task, the player must first move to the relevant work station in the train (i.e. if a window is broken, the player must move there) and interact with the work station to begin the task. After this, the player must complete a simple task on their controller, such as hitting a moving button, pressing several buttons in the right order within a short time span or pressing two buttons fast enough in rapid succession. Some of these tasks can be solved by a single person, whilst others will require the help of several players to solve. The cooperative puzzles requires management within the group, as some of them may require the players to share clues with the other players that only they can see on their controller.

The tasks will appear in different areas of the train, requiring players to move in order to reach the workstation at which the task can be solved.

Each player has a unique role on the team, such as Chef, Conductor, Engineer etc. Each role restricts the player from doing certain tasks, effectively meaning that tasks or part of cooperative tasks can only be done by a certain player. This will force the group to coordinate which member will be solving which tasks, hopefully adding a stress factor to the game.

If the players fail to complete a task within the timer, they will suffer penalties based on the task. Some tasks will subtract from the health of the train, whilst other tasks might subtract from the score of the team or reduce the remaining timer on all other active tasks. Should the players on the other hand succeed to complete a task within its timer, they will avoid the imposed penalties, and in some cases earn additional bonuses as well, such as increased score or extra time for all other active tasks.

Currently the game is using a 4-way movement system, but the group is discussing changing the movement system based on how well it performs. Other suggestions for movement currently include a grid based movement system and a system where you simply click specific rooms to move there.

Setting

The game is set in a western theme, travelling through a classic western scene on a train. Events, characters and environment will be designed to fit the theme, as will sound effects and music. As the train travels from station to station, the environment will go through a day and night cycle, changing the feeling of the scene based on the progress of the players.

Design mock up:

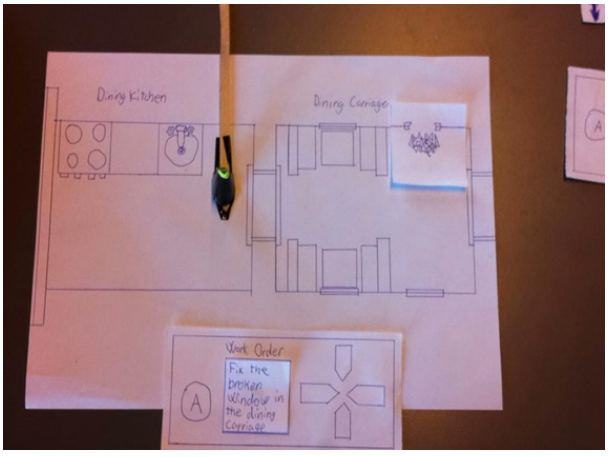


Figure 10.1: A mock up of initial designs: Both level design and controller design.

10.2 Game Art

This chapter is a collection of design samples. The samples are gathered both to show the game art and design, but also to illustrate how the western theme was implemented.

10.2.1 Sketches

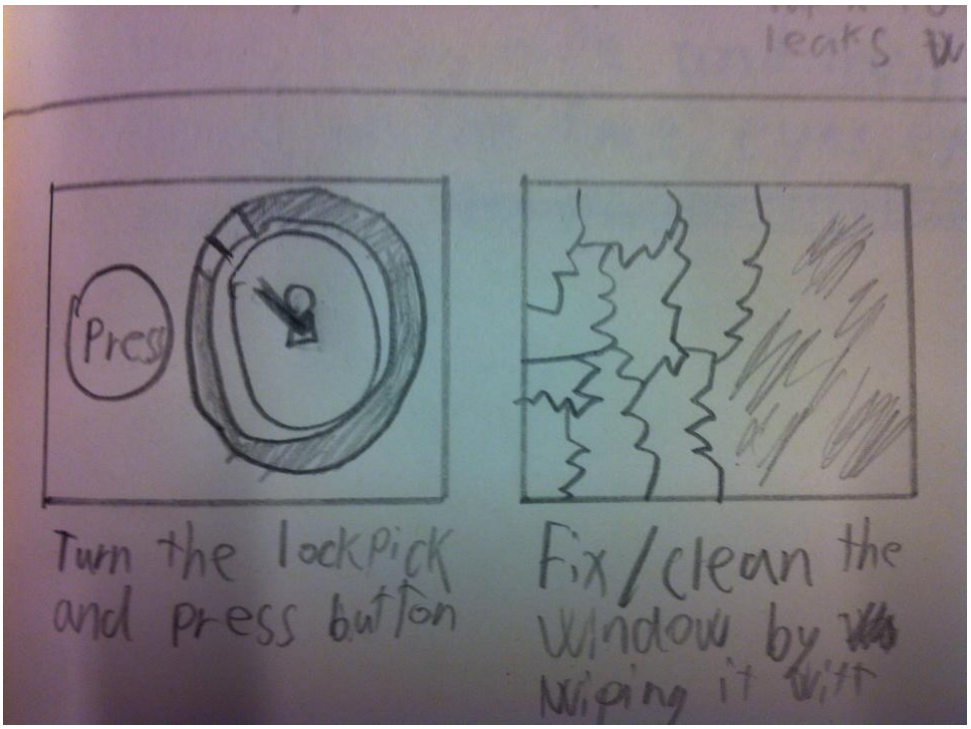


Figure 10.2: Two sketches of early micro mechanic ideas.

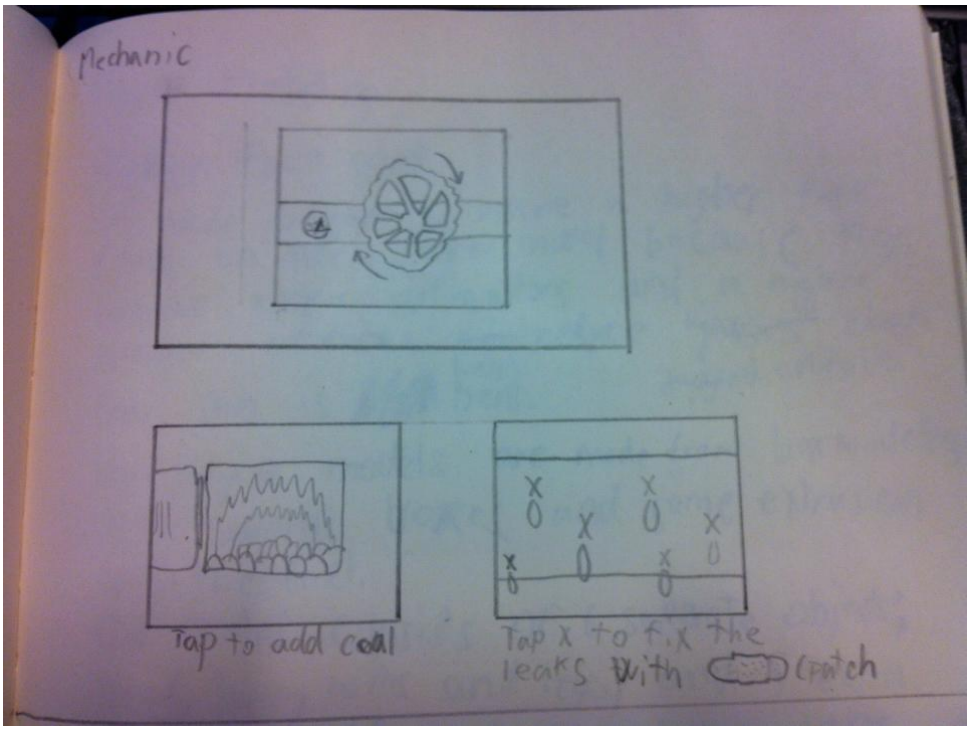


Figure 10.3: Three sketches of early micro mechanic ideas.

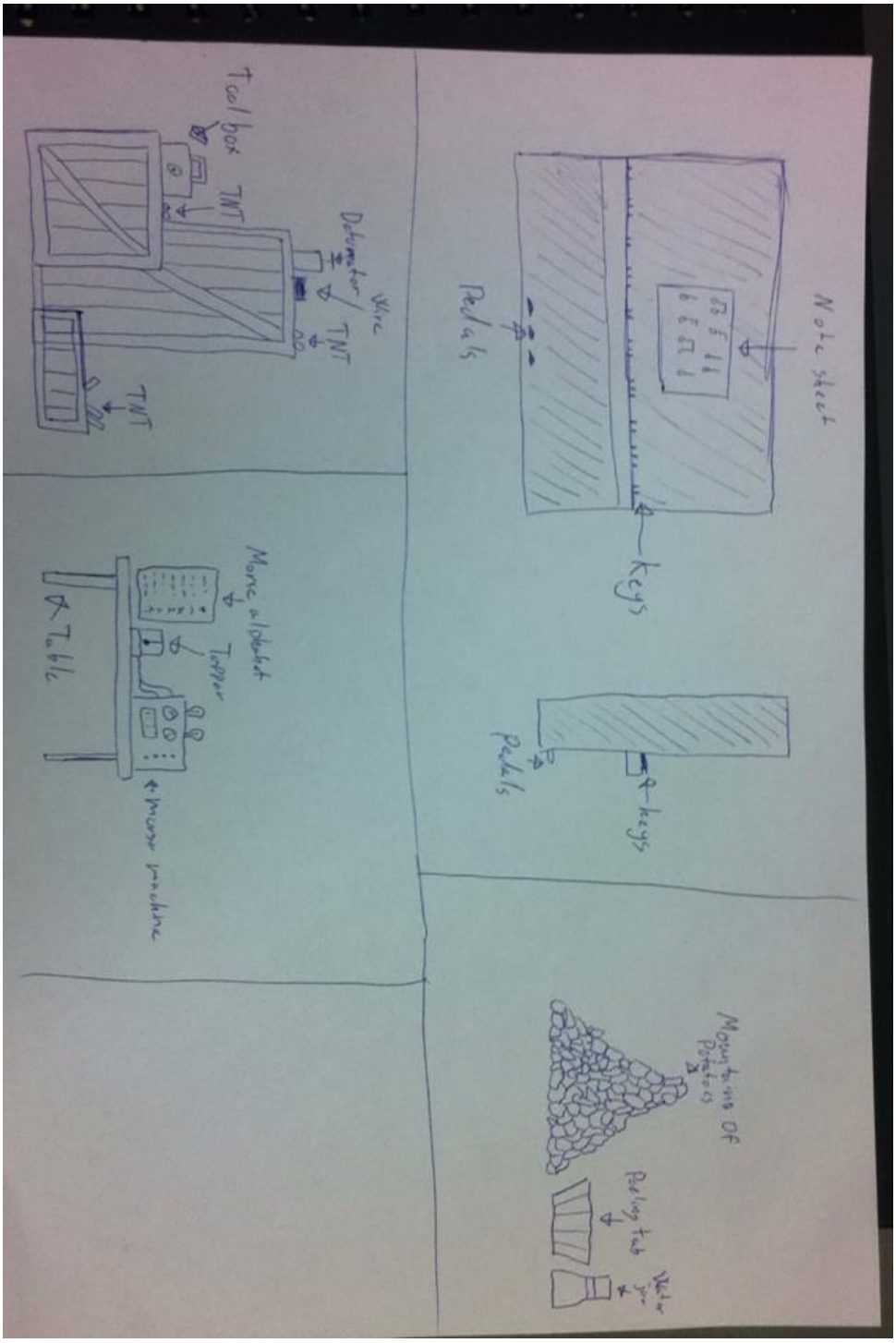


Figure 10.4: Sketches of different elements on the train, which could be made into work stations.

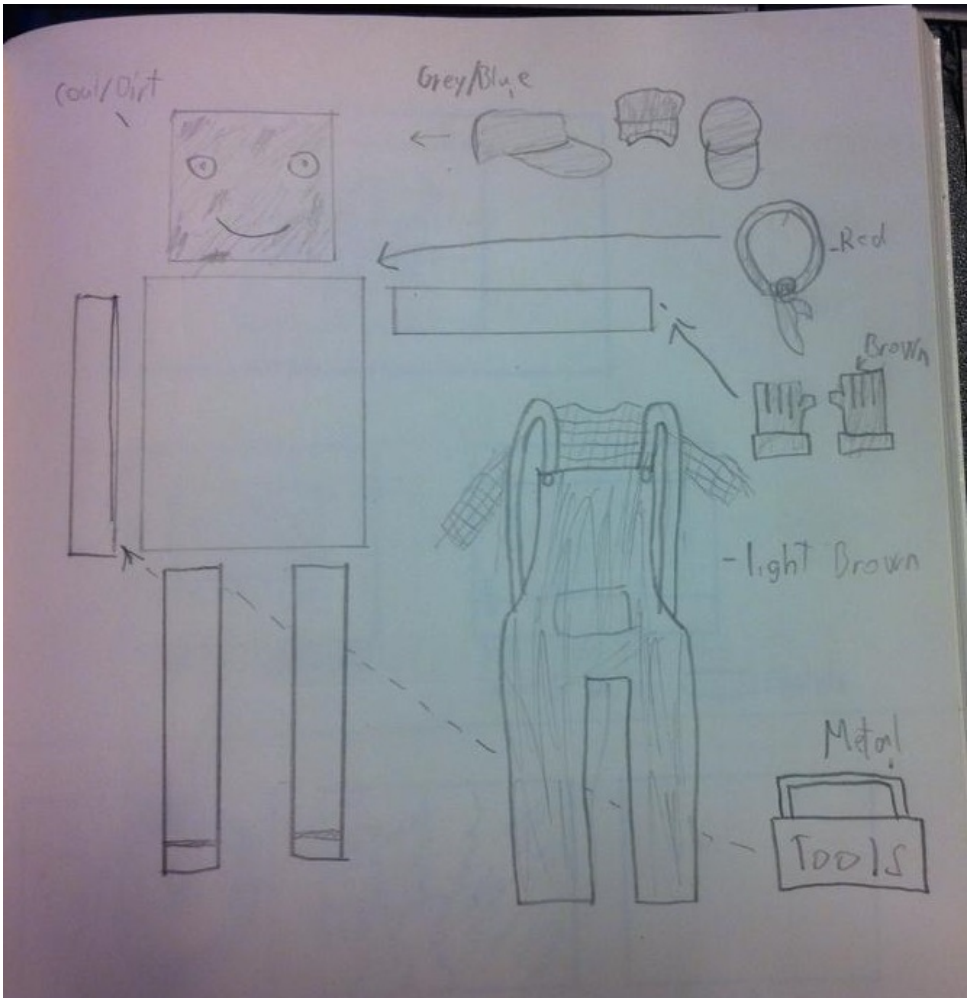


Figure 10.5: A sketch of the mechanic character, with ideas of how he could appear.

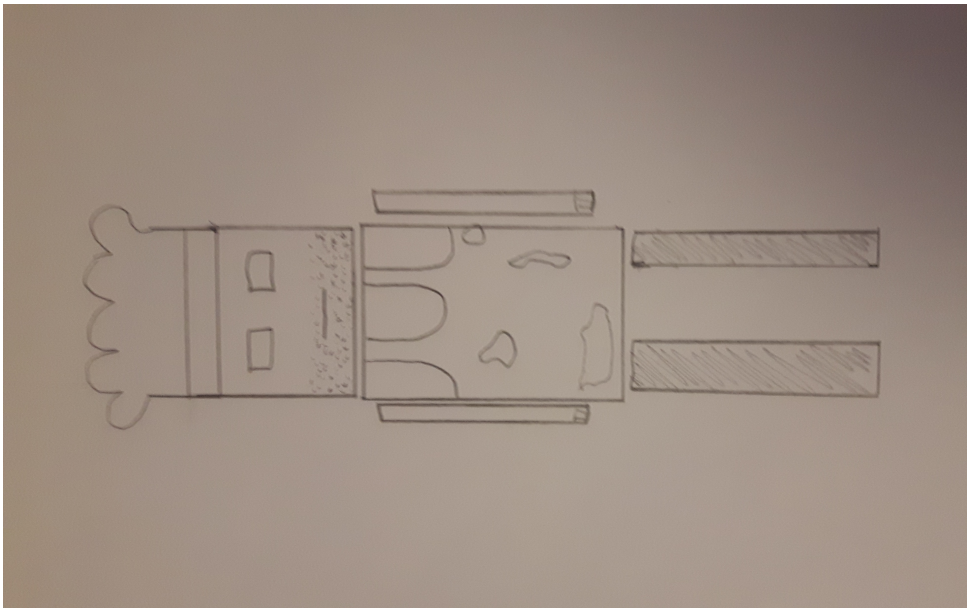


Figure 10.6: A sketch of the chef character, with ideas of how he could appear.

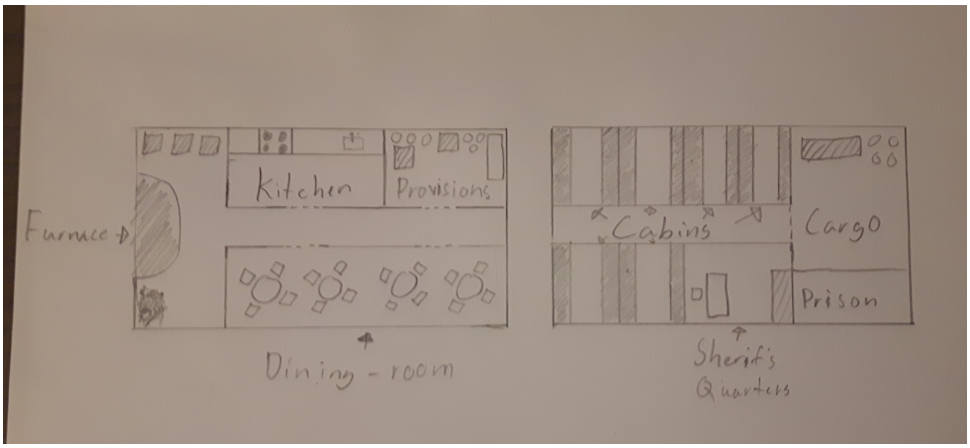


Figure 10.7: A sketch of the layout of the different rooms on the train.

10.2.2 Character Design

The design of the four avatars in the game were planned to ensure two elements. To make them fit into the western themed game, but also for each of them to have their own original appearance. This appearance were created by individual headgear, beards and clothes. In some examples the avatars were also equipped with items. They were all designed from a stereotypical point of view. For example, the sheriff with his sheriff star, mechanic with full beard or the operator with his formal wear. This were a design choice to make their role more iconic, so that they would be easier to recognize. The graphic art were also kept true towards the "box graphics".

Figure 10.8: The design of the train operator, recognizable by his traditional cap and formal wear.

Figure 10.9: The design of the chef, recognizable by his white clothes and iconic chef toque.

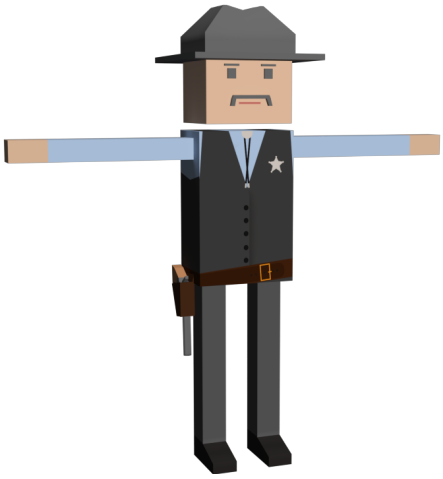


Figure 10.10: The design of the sheriff, recognizable by his cowboy hat, pistol holster, iconic outfit, gunslinger mustache and sheriff star.

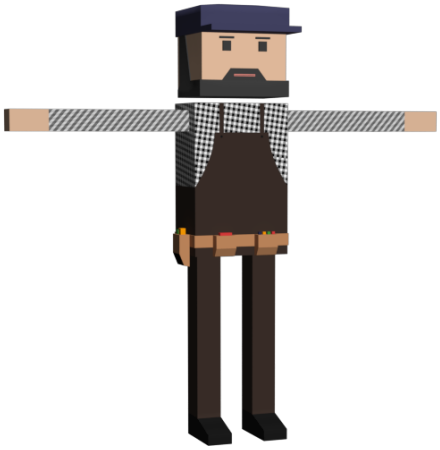


Figure 10.11: The design of the train mechanic, recognizable by his more practical outfit, tool belt and full beard.

10.2.3 Character Icons

The character icons are used in-game and represents the avatar each player has been given. They are placed at the bottom of the main screen inside a colored border. Each of the four icons have been designed with an authentic western style at mind, but with different appearances connected to their individual role. For example, the chef can be recognized by his white clothes and the chef's togue. Another example is the train train operator and his more formal appearance and cap. The sheriff and mechanic are more difficult to recognize by their icon alone, but their characters are equipped with a gun holster and belt with tools.

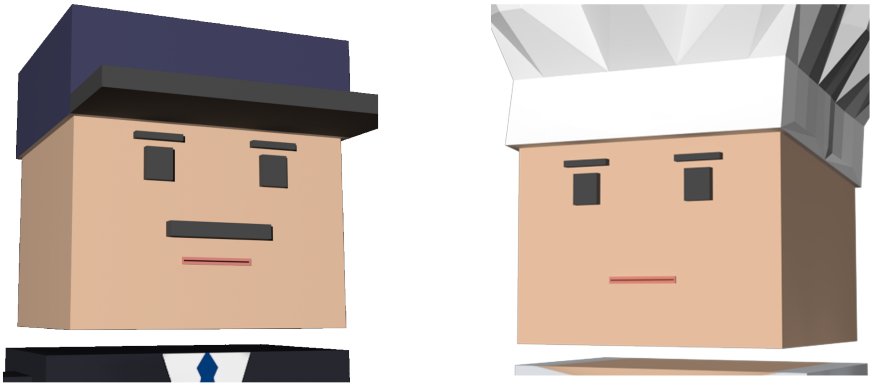


Figure 10.12: (From left to right) Showcases the train operator and the chef.

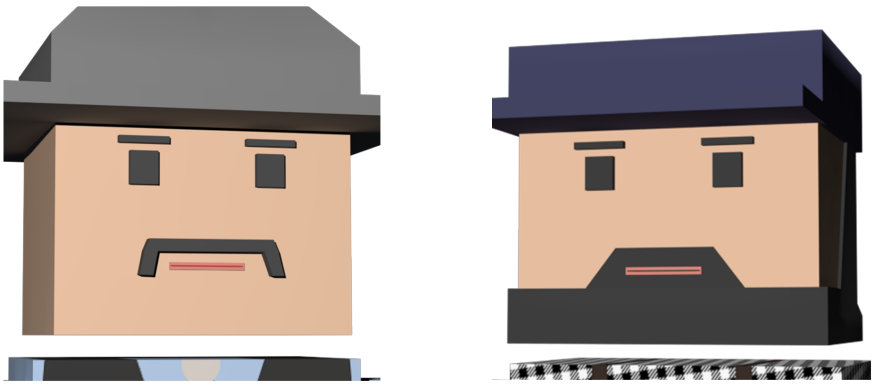


Figure 10.13: (From left to right) Showcases the sheriff and the train mechanic.

10.2.4 Train Design

Designing the train was a mayor step in the initial design phase. This phase mainly focused on how many wagons the train should have, how to design it as a western themed train and from which angle the train itself should be seen from during the game. The angle itself would be a deciding factor in how long the train should be, because all four players had to be able to see the interior level, at all time. Different viewpoints would each give different areas to cover, and with to many wagons to cover the camera would be zoomed out to much for the players to see anything. The chosen angle for the viewpoint would also affect how the interior design should be placed, because larger objects and workstations in the wrong place could block line of sight. This could not entirely be avoided, because a slightly sloping angle was chosen, but the interior was designed to avoid it as much as possible. The outermost walls closed to the camera as well as the interior walls would be invisible for the players, allowing them to see inside the wagons.

Figure 10.14: The wagon at the back houses the kitchen, dining room, passenger seats and sleeping areas.

Figure 10.15: The front wagon is designed to feature a shooting area, prison and multiple maintenance rooms.

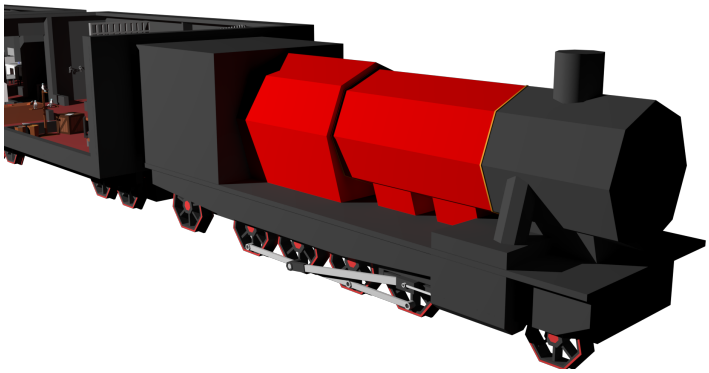


Figure 10.16: The front of the train is designed to look like a steam locomotive, because this is corresponds with the chosen western theme.

10.2.5 Workstation Design

There are four workstations in the game, each implemented with their own micro-game. These micro-games will appear on the smartphone after activating one of the workstations. The stations are placed as realistically as possible inside the train wagons,for example the stove is placed inside the kitchen area and the pipebox close to the distribution board. The workstations are briefly highlighted once an event is ready.

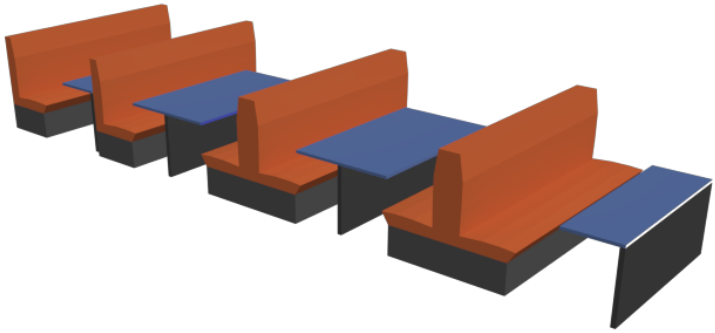


Figure 10.17: The passenger seat workstation.

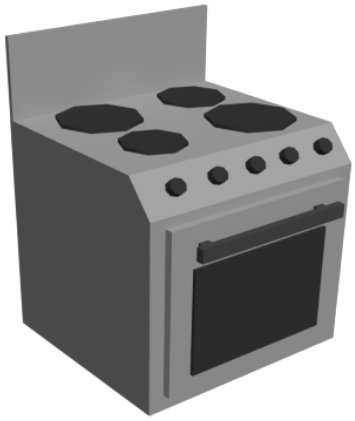


Figure 10.18: The stove workstation.

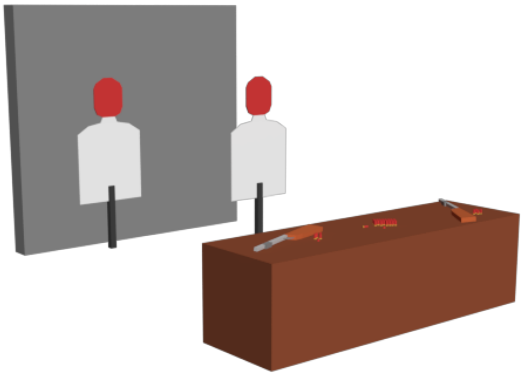


Figure 10.19: The shooting range workstation.

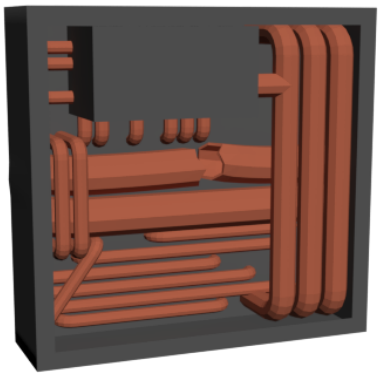


Figure 10.20: The broken pipes workstation.

10.3 Sending and receiving information

Below are three examples of how data is handled and cast into their respective data type.

```
1 if (data["move"] != null) moveDirection = ((string)data["move"]);
2 if (data["isPressed"] != null) movePressed = ((bool)data["isPressed"]);
3 if (data["button_push_A"] != null) a = (bool)data["button_push_A"];
```

As can be seen by the example, the information is accessed in the data package by name, these names are determined and set by the sender, and the reciever will have to know or guess what those data packages are named for it to easily access them, though it is possible to deserialize

that information, at the cost of some extra processing.

The shortcomings of this system is that if a package is recieved and said package is corrupted or contains something other than what is expected, an exception is thrown. Given the custom packages are made specifically for this project, the chance of this event is highly unlikely.

Sending information also allows for great customization. It starts with declaring and initializing a "message" variable, and writing the desired information into said message. In the case of opdating the list of currently active events on the controller devices, three arrays are sent, those three arrays eventNames; which details the names of the active events, eventGroups; which details which group of players are able to solve it, eventStatus; which details the current status of the events(i.e. active or inactive) and list_empty; Which controls whether or not the list of events is supposed to be visible for the user. Creating the package looks as such:

```
var message = new
            {
                info = new { eventNames = ListOfEventNames,
                eventGroups = GroupsAsText,
                eventStatus = ListOfEventStatus,
                list_empty = false }
            };
```

That creates the message and then the message can be send using the AirConsole "Message(int DeviceID, JToken data)" function, at this point the game still handles the players with a integer detailing which active player number they currently have. Thus before sending the package the associated DeviceID must be retrieved, and that is retrieved, using an AirConsole getter function called: "ConvertPlayerNumberToDeviceId(int Playernumber)". Said function is used in the message function call and looks as such:

```
if (player_number == 1) { AirConsole.instance.Message(AirConsole.instance.
    ConvertPlayerNumberToDeviceId(0), message); }
if (player_number == 2) { AirConsole.instance.Message(AirConsole.instance.
    ConvertPlayerNumberToDeviceId(1), message); }
if (player_number == 3) { AirConsole.instance.Message(AirConsole.instance.
    ConvertPlayerNumberToDeviceId(2), message); }
if (player_number == 4) { AirConsole.instance.Message(AirConsole.instance.
    ConvertPlayerNumberToDeviceId(3), message); }
```

10.4 Game Controller

This section will be going through how the game controller is build, the process and design. This section will not cover the basics of HTML, CSS and javascript/jquery, but focus on more important parts of the code. The following code will consist of three different languages, HTML (Hyper text markup language), CSS (cascading style sheets) and javascript.

The overall design with movement and buttons is built after a standard controller design, examples like playstation and x-box, having the movement on left side and the interaction buttons on the right side. Since airconsole gives the possibility of changing the controller design, we then added a third part to the standard controller. The Task list which is used throughout the game to keep track of the game objects, overall controller design is shown in figure 1.

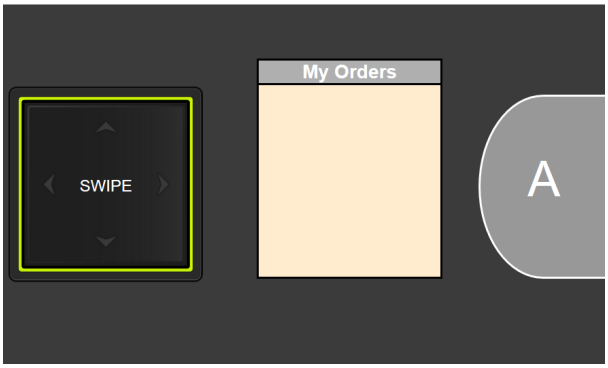


Figure 10.21: The default look of the game controller, containing a movement control, task list and interactive button.

The swipe-d-pad was created with AirConsole own API, only thing changed was the messages send to the game screen. With a limit of 10 packages per device the swipe-d-pad had a problem, since it would send too many when swiping in a direction. Sending a TRUE boolean for the swipe direction and FALSE for all other direction, removing the false Booleans gave more space. Participants would experience delay when moving the swipe-pad to fast around, due to all the messages send from the controller to the screen.

```
1  "directionchange": function(key, pressed) {
2    if (pressed == true) {
3      console.log(key+" : "+ pressed);
4      airconsole.message(AirConsole.SCREEN, {move: key});
5    }
6  },
```

This code above is the part that sends data to the game screen, the *"directionchange"* was AirConsoles own API and was not changed. It has two input a string with the name of the direction and a Boolean if the direction is true, normally the function would send four messages but we cut away the variables containing Booleans with FALSE. This way the amount of messages send is significantly reduced by three fourths.

The task list gets updated by receiving to arrays from the game screen, with the group number which contains numbers, who can do certain task, and what color the task is getting on the list. The list is not interactive but just there for information about the game, the information on the list is split into two columns. First column has a color, each player will see two color, “white” for shared tasks and color for individual tasks. Blue player will only see “white” and “blue” tasks, this is achieved with checking the group number of the task send by the game screen.

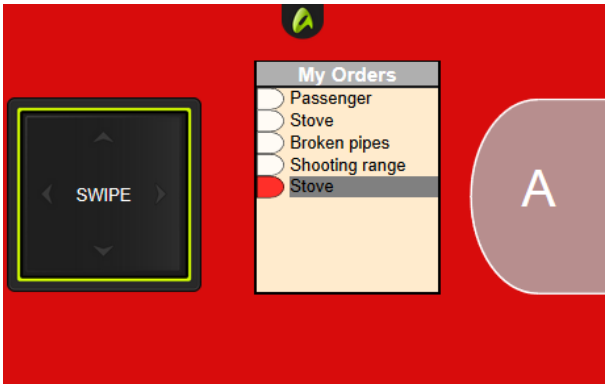


Figure 10.22: Player 1 game controller, showing active task both specific and queued.

Each player gets a different background on their controller, this was done to tell them apart. Figure "red" shows an active task list for player one, as mentioned the first columns only shows colors, and the second column was filled with strings. Player one will not be able to see blue, yellow and green colored tasks. The string gives information about what item in the game was broken. The text would be grayed out if the task was queued, this means there is already a task at the object.
The "A" button is is used to interact with object on the game screen, it simply sends a Boolean to the screen and the controller receives a name of the micro mechanic if the player is close enough to an object.

10.5 Micro mechanics

The micro-games are very simple and designed to only take a couple seconds. They are functioning as a stress-factor and a way of implementing the multi-screen aspect so that the users utilize both screens. This is also why the micro-games can be seen as micro mechanics, implemented as a part of the main game to provide multi-screen gameplay. The users always have to be aware of the micro mechanics, switching the focus from the monitor to their smartphone, but also have to keep communicating where they go. This is due to some micro-games only being available for a specific character, and if one player fails it adds extra pressure on everybody. How many micro-games the players can fail before losing the game can always be seen in the HUD. In the examples below the different micro-games will be shown.

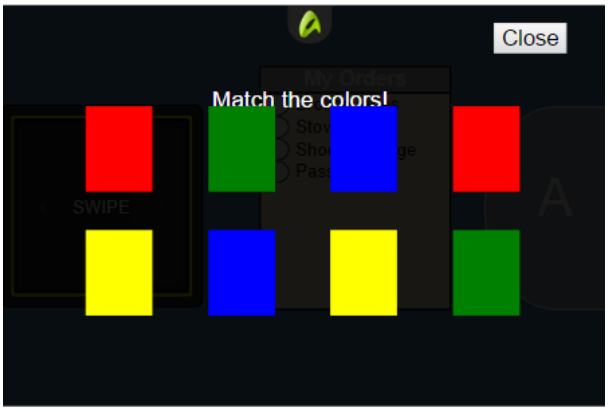


Figure 10.23: This micro-game is activated through the shooting range workstation and is completed by pressing any of the colors, followed up by then pressing the same color again.

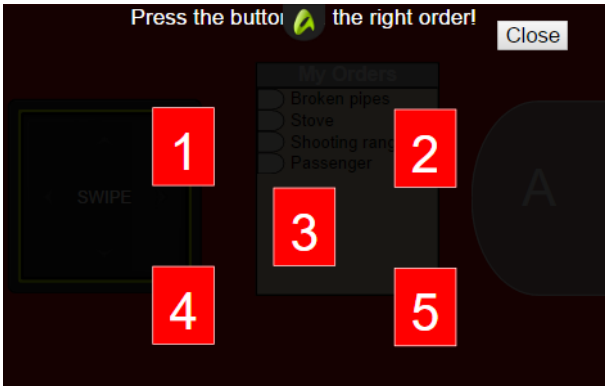


Figure 10.24: This micro-game is connected to the stove workstation and is about pressing the numbers in correct order.

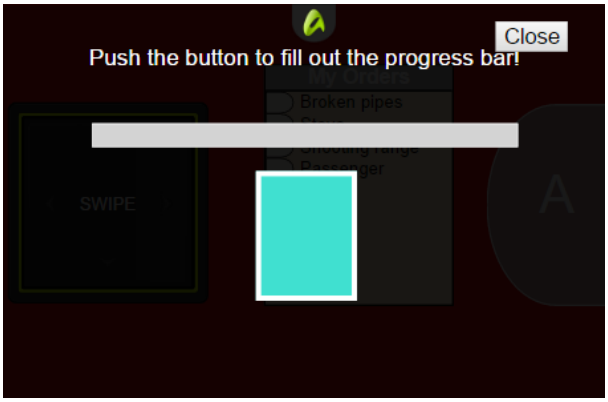


Figure 10.25: This micro-game is launched once the player activates the passenger seat workstation and is completed by pressing button many times in succession.

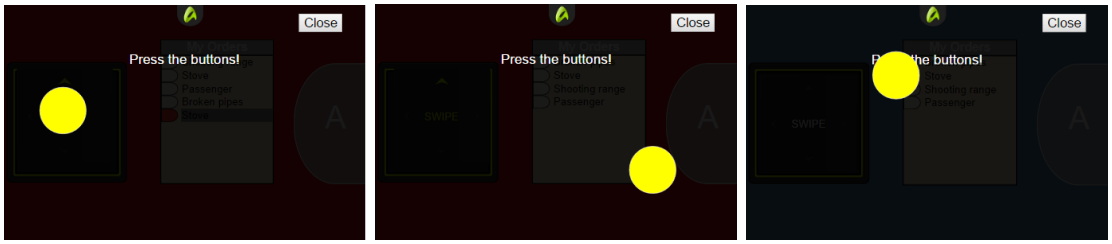


Figure 10.26: This micro-game is activated through the pipebox workstation and is completed by pressing the button eight times. Once pressed, the button will reappear at another point.

10.6 Visual Cues

10.6.1 Pop-Up Animations and Radial Based Timer

Figure 10.27: A collection of the sprites that are animated in the game scene.

Almost all sprite movement are animated using the animator in Unity to keyframe certain transitions and sprite settings. Another method used is providing a script for the sprites so it changes over a specific set time which would usually be difficult to make in the animator. An example is the circular radial based timers in figure 10.27. These timers runs out from being a full circle to nothing, "slicing" the circle from it's center. The script basically gets the time of when the event started, who can do the event and uses the time the event will take to finish and ticks down like a clock from there, slicing the circle until the time is over and the image is completely removed.

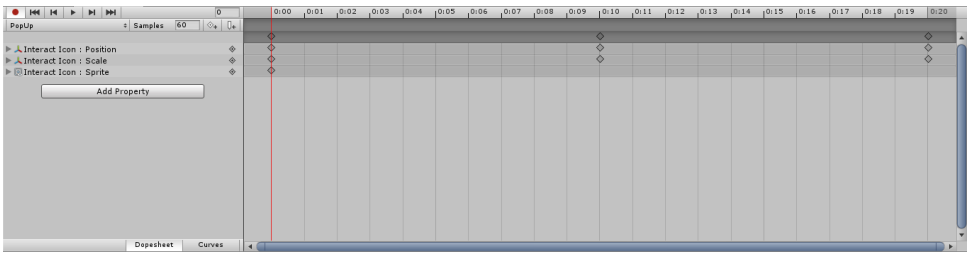


Figure 10.28: How the keyframes for the A button prompt looks.

A very simple animation here for the A button. It is only meant to pop up over the head of the

player and stay there while the player remains in the zone of the workstation, the workstation has an active event and the event can be completed by that players role. The easiest way to trigger animations to happen on sprite is to disable the gameobject until it is needed. Once enabled the animation will play on activation and by setting it to not loop it will remain in the state the last keyframe is set to. This method is used for all the animated sprites that makes use of the Unity animator. The indicator over the head of the player at the start of the game and as previously mentioned, the radial based timer makes use of a script instead.

Figure 10.29: How the keyframes for the A button prompt looks.

The sprites above the head of the player as seen in figure 10.29 or just above the workstations are achieved by using a local canvas on their parent gameobject. What is also noticeable in figure 10.28 is that the sprites in the canvas is always facing the camera. This is also known as billboarding. Billboarding is a method in which you make the object (commonly a 2D object or image) face the camera at all times. This is really simple to implement in Unity as there is a function called transform.LookAt(target); that does just that.

10.6.2 Notification animations

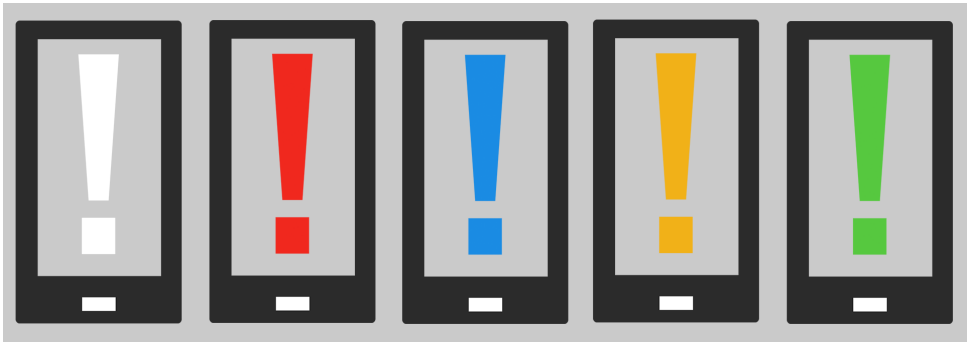


Figure 10.30: The sprites that are going to be animated to act as a visual cue.

This set of images are designed to look like a phone with a mark inside them to act as a cue for the user to look down on the phone to update them self on the list of work orders. These are

simply made up of four shapes made in Photoshop and exported as a PNG individually. They can also be imported as one image and then edited in the sprite editor in Unity to separate them as individual files.

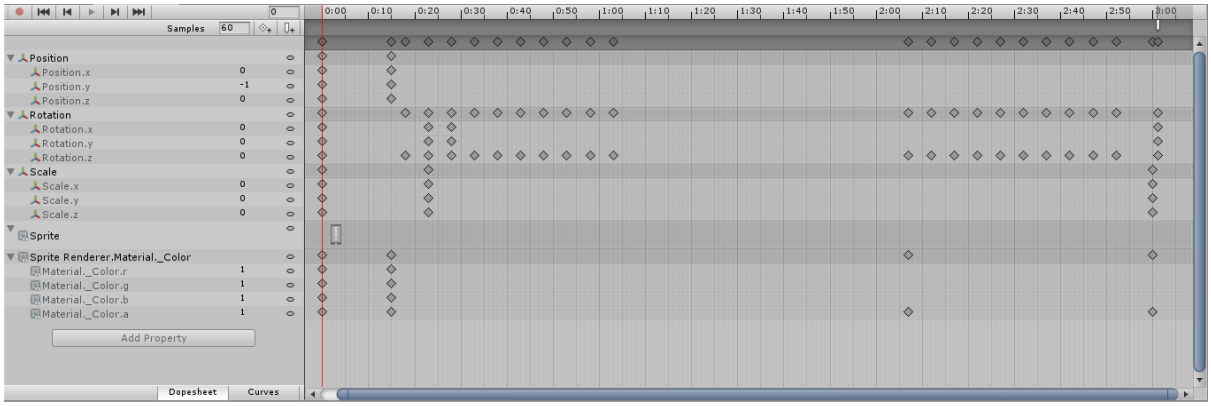


Figure 10.31: A preview of one of the animators with the key frames used for the notification animation.

They are animated very similar to how the other pop-up animation works just a slightly different result. Apart from the pop-up, they disappear after the animation ends. This is by making use of the feature in the animator to call a function when it reaches a certain point of the animation. The function that is called here is to turn off the gameobject once it has faded out completely.

10.6.3 Highlight shader

The shader removes the object and renders only the outline around the object, this way a dublicate of the object can be placed as a child object, and still give the original object an outline, without using excessive amounts of drawcalls.

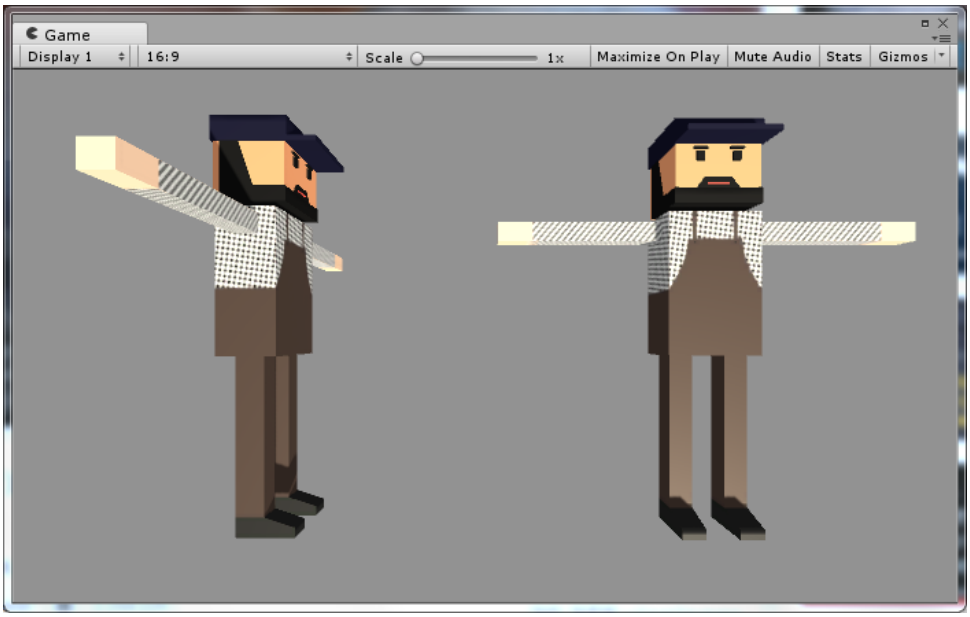


Figure 10.32: A preview of one of the player characters, without the shader applied.

The shader works by rendering in 3 passes, resulting in one output that is only the outline around the object. The shader is applied to an object, for example of original object figure10.32 Above.

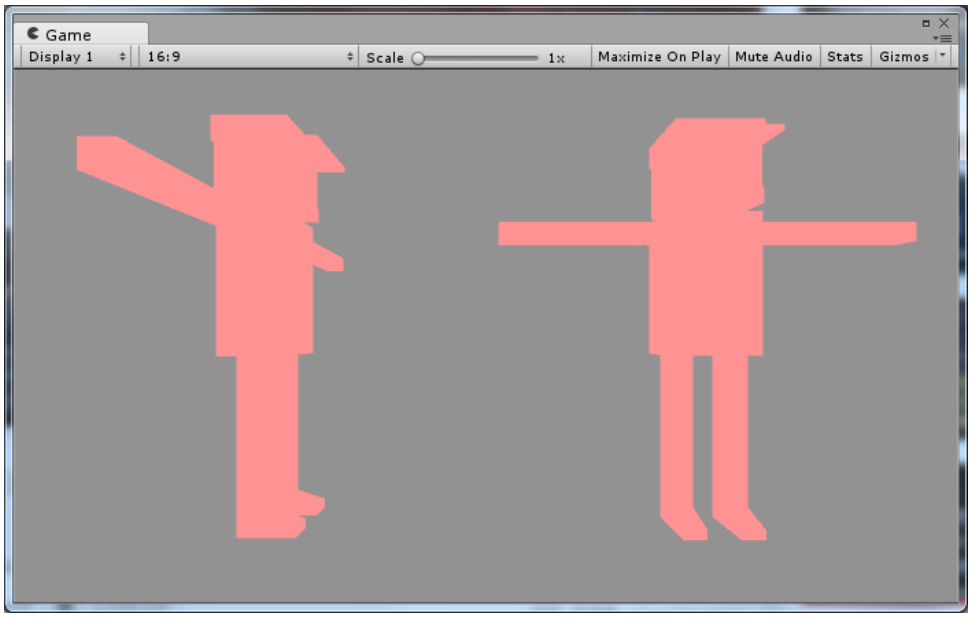


Figure 10.33: A preview of one of the player characters, after the first pass.

After the first pass the entire model is rendered in a solid color, with all of its vertex points extruded by an offset value. The offset value is controlled by a highlighting script, as well as the color of the outline. For the actual render the object has no color and an alpha channel value of 0, but for the sake of demonstration figure 10.33 is colored red.

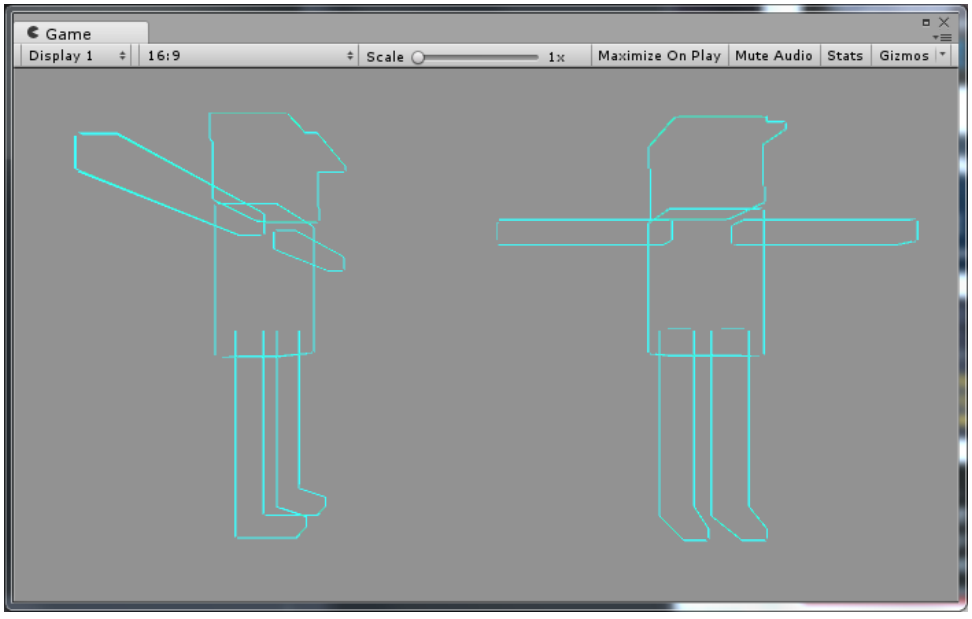


Figure 10.34: A preview of one of the player characters, after the finale pass.

After that, the center of it is removed, but drawn at a separate point in the render queue, if this position is increase it is possible to see a full silhouette of the object through other objects, for this particular implementation it is rendered early so it gets overwritten by the primary model. This is a per vertex operation that calculates the color for each vertex and interpolates the colors in between

After that, the outline is rendered in the color determined by the scripts hard-coded value, as seen on 10.34.

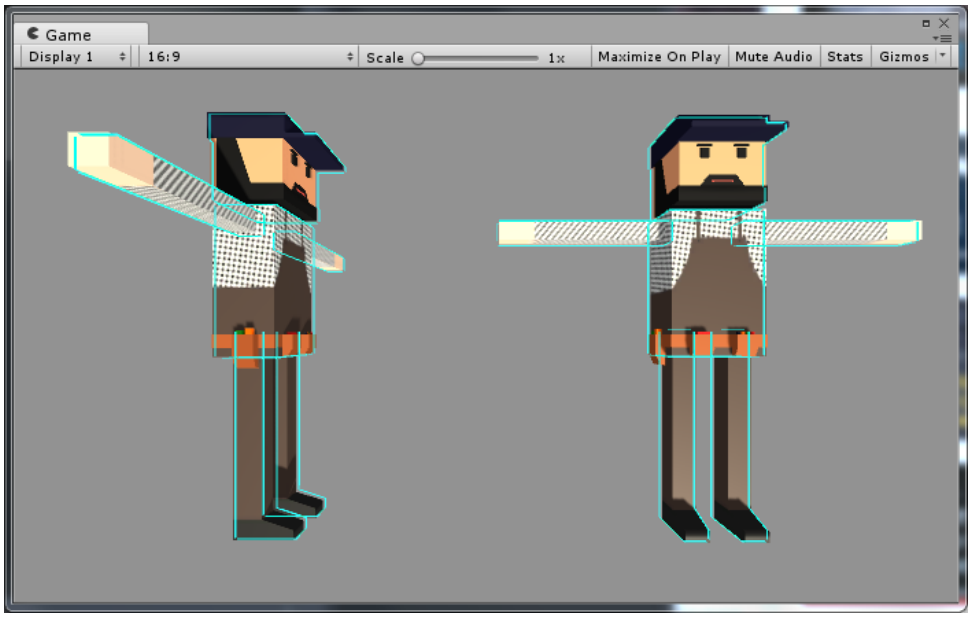


Figure 10.35: A preview of one of the player characters, with the shader and extra model applied.

Figure 10.35 an image showing the model and the overlaid highlight model shows the finale result.

10.7 3D Graphics and Animations

The process of creating all the assets in the game is a bit complicated when it comes to use of software. This chapter is going to explain the process from nothing to a fully implemented and animated model in Unity using motion capture. The process follows the same path as the illustration shows in figure 6.4.

10.7.1 Modeling in 3Ds Max

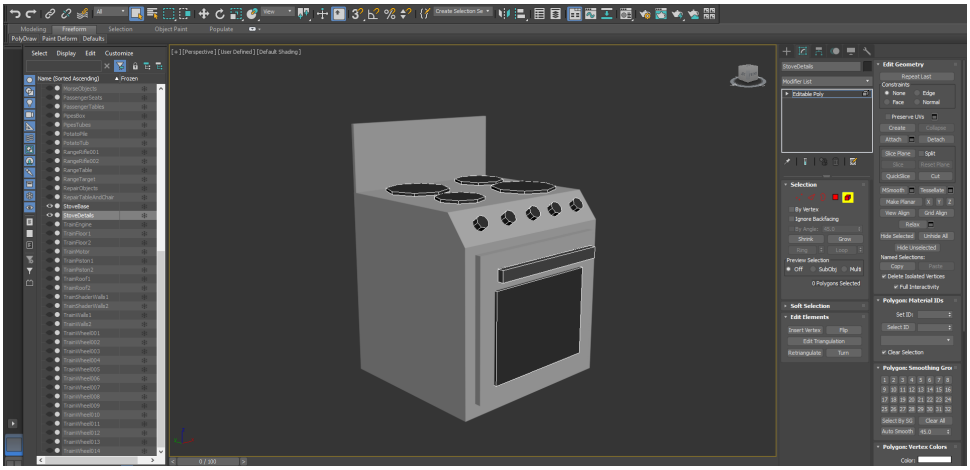


Figure 10.36: An example of one of the box modeled assets in the game made in 3Ds Max.

All of the present assets and character models in the game are created using the method of box modeling. Sometimes the models are made up of several shapes that are box modeled as seen in figure 10.36. The base of that model is made up of a single box extruded and cut to fit the shape of a stove. Several cylinders is added to the object to add details as well as more box shapes for the handle and to make it look like there is an oven in the stove. This is one of the more simple examples of a box shape in the game.

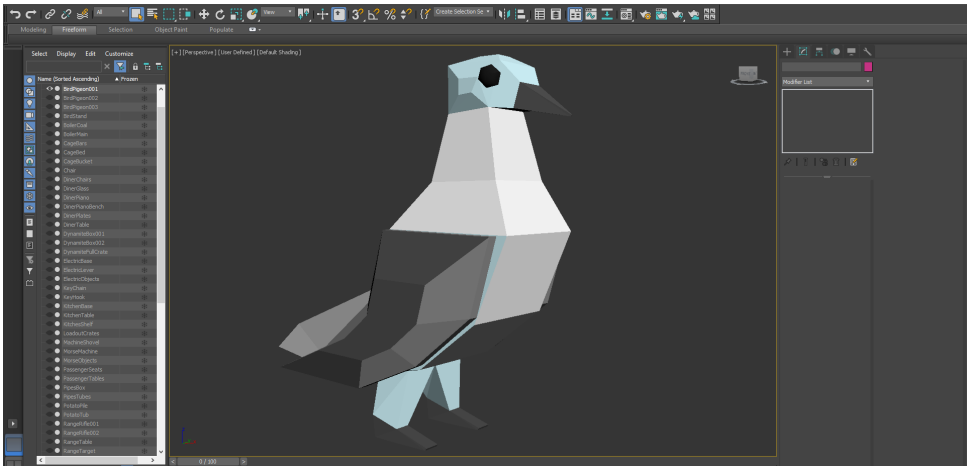


Figure 10.37: An example of one of the more complicated assets to box model in 3Ds max.

A more complicated model made with box modeling as seen in figure 10.37 is a bit harder to create. This model is build up from one box shape. The general approach is to try and make the form of the pigeon before adding details with extrusion that acts as an ease to the hard edges. Another tool that makes the process is a bit simpler is to cut the model to add more vertexes to work with and extrude more specific faces instead of the whole thing. Making the pigeon the same on both sides is a struggle as the bridge tool of edges in 3Ds Max is not always working. At times it will allow the bridge between two edges and draw the face, other times

it just wont even though the procedure on both sides of the pigeon are identical. However the results is pretty similar when looking at both sides.

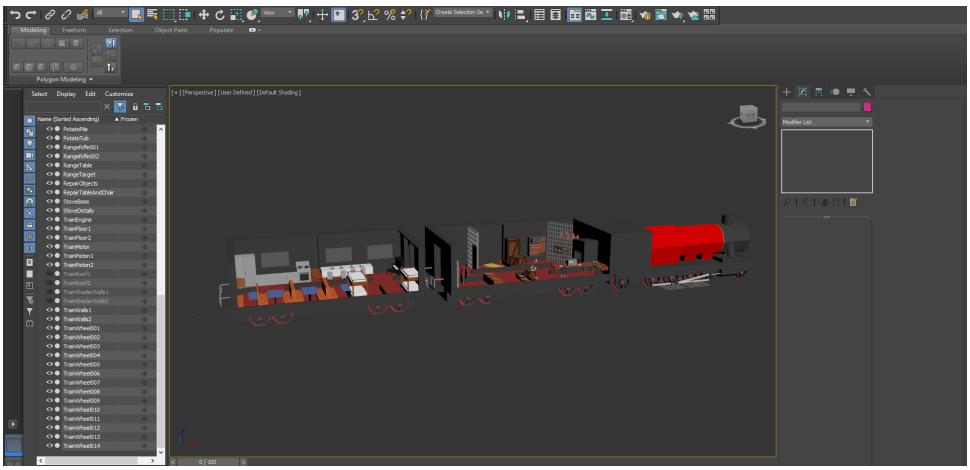


Figure 10.38: An example of the complete set of assets in the train in 3Ds Max.

All the 3D assets that are static is collected in one big file as seen in figure 10.38 to ensure all scales and positions are correct in the train. Placing 3D objects in Unity pixel perfect could be somewhat troublesome after importing them from 3Ds Max. This way it was also easier to figure out the right scale for each object when collected as one. This does however make it somewhat more difficult to add new objects without wiping the model from the current scene in Unity. Thus this was done in the end of the process and stand-in objects were used in the programming process. Exporting assets from 3Ds Max to Unity is pretty straight forward. There is a game exporter build in 3Ds Max which is basically and .fbx file exporter. To get all the required files with the models when exporting, simply tick on embed media in the export settings to include the UVW map or other texture maps if any is used. Afterwards just make sure the .fbx file is put into the assets folder in Unity and it will include all the materials and UVW maps on its own in folders.

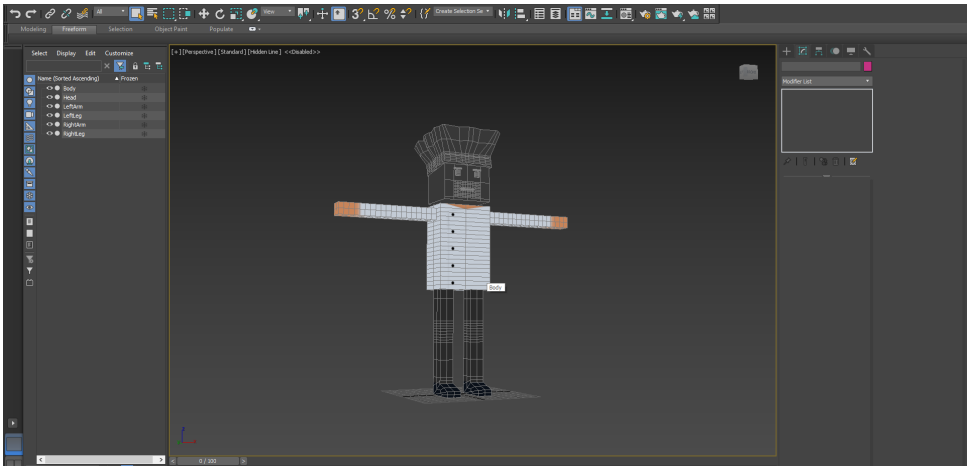


Figure 10.39: An example of the chef character model in 3Ds Max.

The modeling of the game characters follow the same method as previously explained, but with

an addition of UVW maps for the arms, torso and legs of the character. The reason why we used UVW maps were to reduce the amount of problems that comes with adding cloth to a character when rigging. Using 3D objects as the clothes of the characters made a lot of the clothes clip through the body. Applying UVW maps to the characters when they are made up of boxes is very simple as seen figure 10.39. Unfolding the masks and drawing within the rendered margins provided by 3Ds Max gives the result we were looking for. All that was required were to make sure the drawings were simple and without smooth curves to fit the box shaped character and make sure to draw beyond the edges of the margin to reduce the amount of white blur lines on the edges of the character.

10.7.2 Rigging in Maya

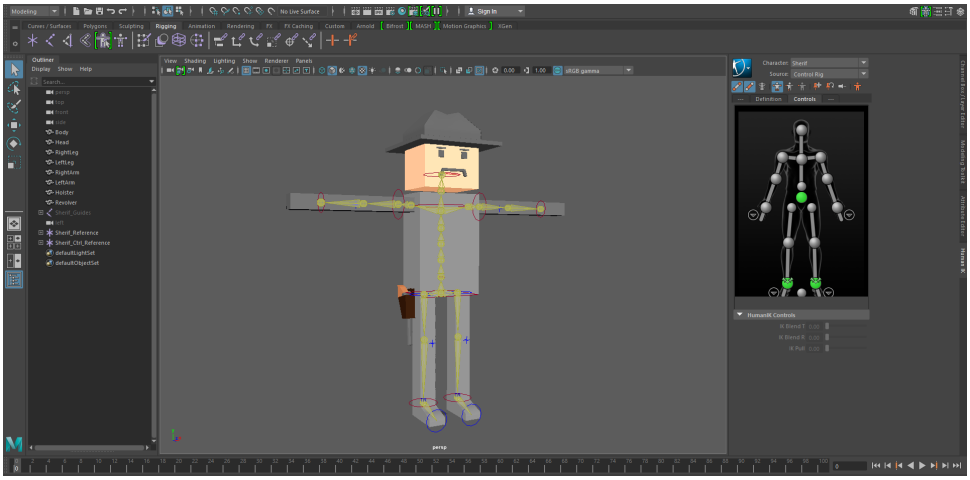


Figure 10.40: An example of the sherif character model rigged in Maya.

The rigging of the character models in Maya is very straight forward when making use of the Quick Rig tool. The rig is made using the tool in the step-by-step mode to ensure all the bones are placed correctly and that the right amount of bones are placed. Once placed and skinning has been applied, a quick check to make sure all the joints and skin moves correctly is acquired. Once everything is in order the model is exported to Unity as an .fbx model just like in 3Ds Max. A naked character model were used in MotionBuilder rigged using the same method as with the other characters to ensure a similar bone structure that is presented in figure 10.40.

10.7.3 Animations in MotionBuilder

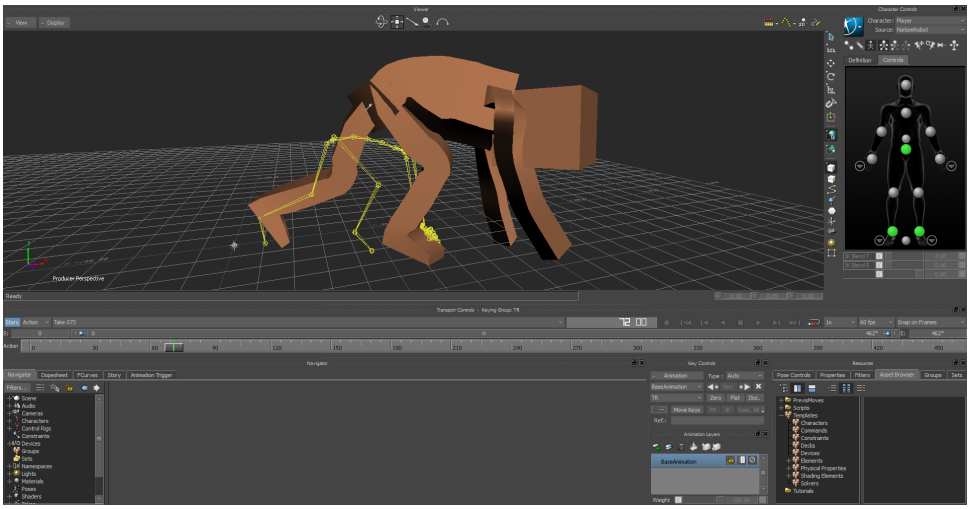


Figure 10.41: A screenshot of our base character model with a motion capture animation.

MotionBuilder was the most fitting application to use for sampling the motion capture animations. It is easy to transfer the objects between the programs as they are all made by AutoDesk and provides a send to program feature. It also made use of the character rig creation made in Maya as they both share the character feature and makes the transition of models in the program seamless. The program records the animations as takes as seen in figure 10.41 and are also exported as an .fbx file which is then send to Unity with the model attached. Otherwise Unity will not be able to track down the animations take without an avatar to link it to.

Figure 10.42: A shot from our motion capture using the Perception Neuron.

The hardware to record the motion capture is called Perception Neuron. It is a set of sensors placed in a similar matter to how bones would be placed in a normal character rig. The sensors are attached to a body harness for the torso and straps for the arms, legs and the head as

seen in figure 10.42. The hardware required a good amount of calibration in between takes to make sure the joints did not rotate or loose sense of position in space. It has a little hub that records the sensors position and sends that information either into the included software or the plugin for MotionBuilder which we used. That hub can be reached through a wired USB connection, recorded on an SD card or with a Wi-Fi connection. The Wi-Fi connection was in a beta and very buggy so that idea was discarded. Since this was a first with motion capture, real time illustration of the character moving was very helpful, thus the SD card was not a good solution either. The final solution resulted in using the cord with a person with a laptop that was recording running behind the person with the motion capture suit on when long distance had to be covered.

10.7.4 3D Models and Animations in Unity

Figure 10.43: A screenshot of the animations applied to the character models.

As mentioned earlier, .fbx files were used as the export file for all the assets and animations in the project. Unity handles these files very well with the embeded media for UVW maps and places those files with the materials in subfolder of where the files are placed. After that the 3D assets can be dragged into the scene from the project window and all the materials and UVW maps will be applied. The character models comes with an avatar file from the export of Mayas own character rig feature. Usually this model can be used right away with the rigging and skinning attached, but sometimes the avatar has to be selected and modified slightly to turn on the rig. The animations have a similar approach as the animation takes has to be rigged to the included base character model with each animation. After that, the animations can be attached to the characters animation controller that has to be created separately with states. Once that is done, the animations can be triggered by having the transitions between animations start on a change of variable as seen in figure 10.43. An example would be when the character is moving, it moves in an axis with a certain speed. If that speed exceeds a certain value, start the running animation.

10.8 Handling events

An event is created as an instance of the struct of the inGameEvents class, using the structs build in set functions, the relevant data for the event is determined. Once the data has been filled in, for this case the events were hard coded for simplicitys sake in testing. the event handler is called to notify the subscribed classes that a new event has started, the event handler procedes to "trigger" the delegate event associated with an event starting.

The subscribed classes are the workstation and the data collection class, note that the event is passed to every workstation regardless of whether the workstation is supposed to use the event or not, this could be optimized for improved performance, but the overhead that it creates is negligable for this project. Once a workstation has recieved the event it will lookup the event type in the event struct and compare it to the works tation type, this will result in the event either being added to the queue if they correspond or ignored.

The workstations operate within four states; Active, Inactive, Failed and interacted. If the current state is inactive, the workstation will pick the oldest event on its queue, and activate it. The workstation does not know what the event is, other than how much time is remaining and where on the list the given event is, the micro mechanic is solved on the controller device, thus there is no reason for the workstation to know anything other than the state of the event.

Once an event runs out of time, another delegate event is triggered through the event handler, telling that the event has failed, and with accompanying boolean the subscribed scripts will know whether or not the event failed or succeeded, and correspondingly either reward or punish the players. At this point the workstation will compare the "finished" event with the event that it currently has going on, if they are the same, the workstation will remove the event from its list.

As can be seen on figure 10.44. The game will retrieve the currently active and queued events while the event is being handled by the workstation. As mentioned in chapter 5.4, the infor-martion each player recieves on their controller device is dependant on the character / role they play. So four generic lists are created and the relevant events coppied into those lists. Then the list is sorted one last time based on whether or not the events in the list are currently active or queued, to sort all of the most relevant events to the top of the list. The information can then be handled on the phone.

While all this is happening the data collecting script is running on the side, noting down data and saving it for future analysis.

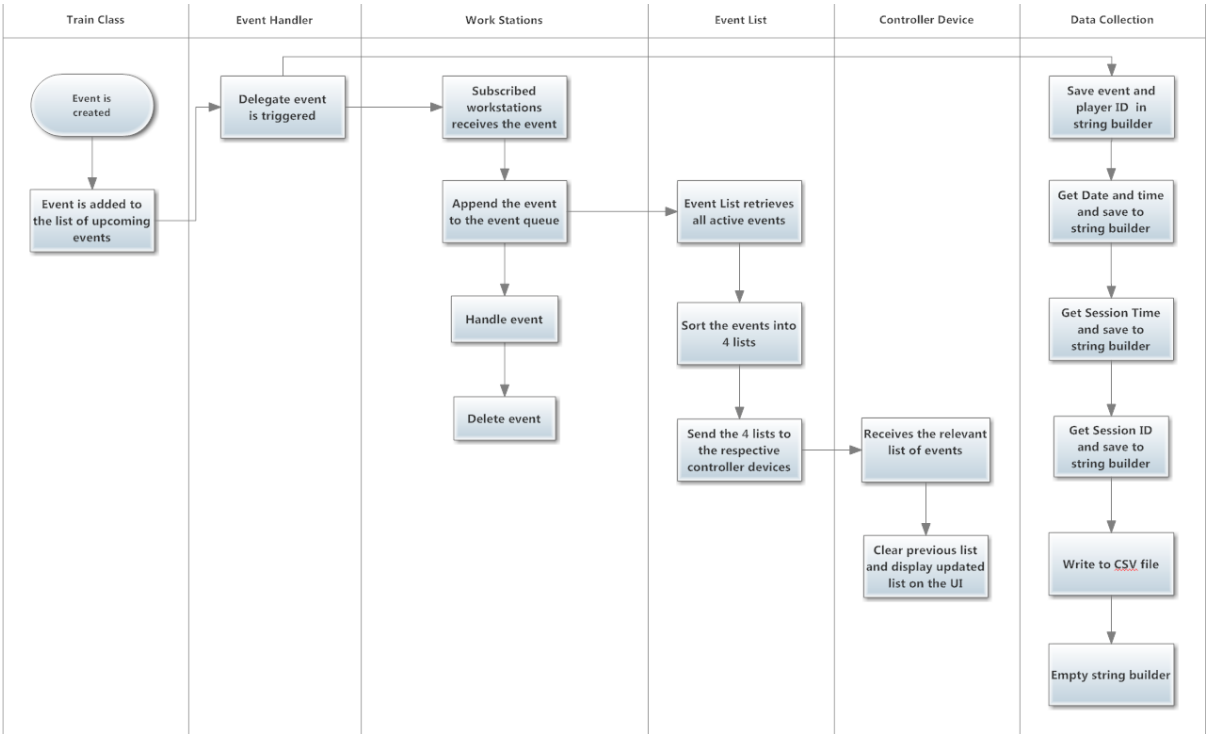


Figure 10.44: A flow chart over the movement between classes for a struct.

10.9 Tests

10.9.1 Consent Forms

The consent form used for testing throughout the project:

MTA16542
Aalborg University

Consent Form

I agree to participate in the study conducted by Group MTA16542 at Aalborg University.
I understand that participation in this study is voluntary and I agree to immediately raise any concerns or areas of discomfort during the session. I can terminate the interview at any time and for any reason I wish.

☐ I hereby give my permission to use all collected information from the interview in anonymised form. All audio and video recordings will be destroyed by the end of the project, unless I give my explicit permission otherwise.

☐ I furthermore give my permission to publish photos and video recordings of me, but only for scientific or teaching purposes.

Name:_____

Smartphone (Make and Model):_____

Signature:_____

Date:_____

Figure 10.45: Consent form used for experiments.

10.9.2 Mental Model Elicitation

In the early design phase a mental model elicitation were made. The mental model elicitation was made as a combined lo-fi test, with a paper mock up model of the game. The test was made to find out if the group's conceptual model was consistent with the users' mental model of the controller. To test this, a perceived affordance scheme was made, involving all elements of the controller. For this test, the controller had two different overlays. A main controller for moving around on the train and reading work orders, and a puzzle controller that would drop down on the smartphone, when a micro game was being solved.

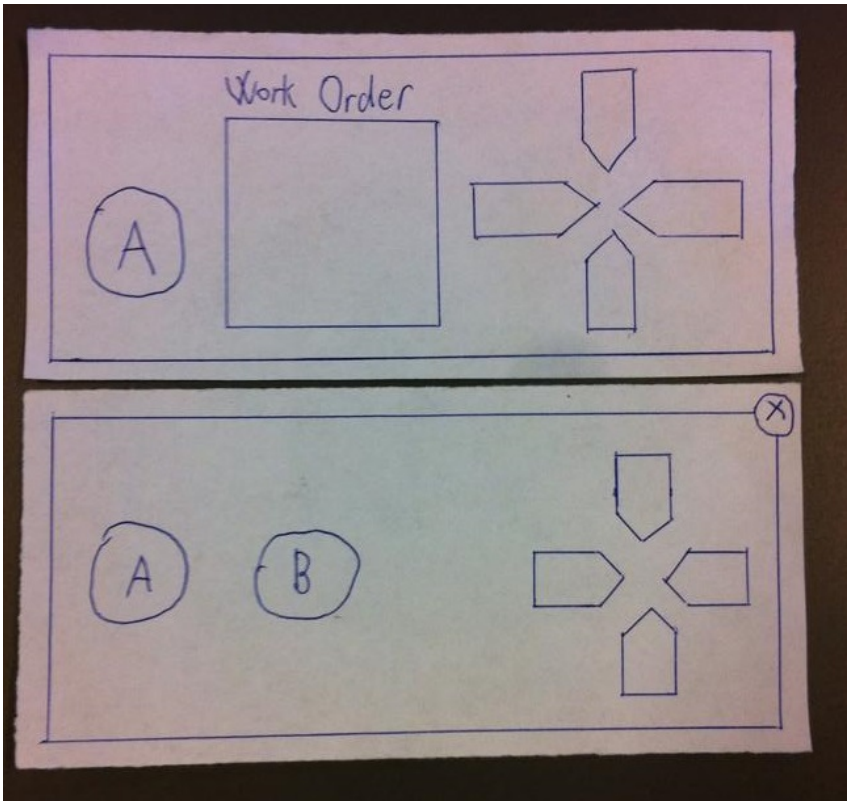


Figure 10.46: A picture of the two controllers, with the main controller in the top, and the micro game controller in the bottom.

A perceived affordance scheme was made for each controller.

Objects	Feedforward	Perceived Affordance	Feedback
Left Button	Pressing the button will move something left	This button can be pressed	Pressing this button moves something left
Right Button	Pressing the button will move something right	This button can be pressed	Pressing this button moves something right
Up Button	Pressing the button will move something up/forward	This button can be pressed	Pressing this button moves something up/forward
Down Button	Pressing the button will move something down/backwards	This button can be pressed	Pressing this button moves something down/backwards
Work Order Screen	This screen can show information	This screen cannot be interacted with	The screen shows information
A "Action" Button	Pressing the button will confirm the presented action	This button can be pressed	Pressing this button confirms/activates the presented action

Figure 10.47: The perceived affordance scheme of the main controller. The scheme states the Feedforward, Feedback, and the perceived affordance of each element of the controller.

Objects	Feedforward	Perceived Affordance	Feedback
Left Button	Pressing the button will move something left	This button can be pressed	Pressing this button moves something left
Right Button	Pressing the button will move something right	This button can be pressed	Pressing this button moves something right
Up Button	Pressing the button will move something up/forward	This button can be pressed	Pressing this button moves something up/forward
Down Button	Pressing the button will move something down/backwards	This button can be pressed	Pressing this button moves something down/backwards
A Button	Pressing the button will activate the main action	This button can be pressed	Pressing this button confirms/activates the main action
B Button	Pressing the button will activate the secondary action	This button can be pressed	Pressing this button confirms/activates the secondary action
X Button	Pressing the button will exit from the screen	This button can be pressed	Pressing this button exits the screen

Figure 10.48: The perceived affordance scheme of the puzzle controller. The scheme states the Feedforward, Feedback, and the perceived affordance of each element of the controller.

The mental model elicitation was tested with 10 participants. The test was split into two, with the first five participants testing the initial procedure, and then changes were made based on the feedback and the remaining five participants tested the new procedure.

Procedure

Before the test begins the participant sign the consent form, allowing for video to be taken of session. After that they are introduced to the project and what it is about. The setting of the game will then be explained, being on a train and repairing stuff. The participant will then be given the first controller for the first time, and asked to describe out loud what he/she sees, and comment on each element and its expected function. They will then be given the second controller overlay and is asked to do the same with that controller. The note taker will in the meantime write down what the participant says into an empty perceived affordance scheme to see if the participants mental model corresponds to the groups conceptual model of the controllers. When they have looked through the controllers the game can start. The participant is asked to think out loud while playing the game, explaining what he thinks is happening when he interacts with the controllers.

The character starts in the bottom right corner of the Kitchen Cart. The first task is to move the character up to the oven. When the character stands at the oven, an alert (The exclamation mark) will appear on the controller and the first work order will be given to them,

and the window will break (place the broken window in the train). The first order is to fix the broken window. When they are at the broken window they have to press the A button to start the puzzle. The puzzle will be laid on the screen and they are given the other controller. They have to solve the puzzle, by choosing a piece with the controller and move it to the correct place. When the puzzle is finished, the puzzle screen will stay until exited by pressing the X button on the controller The exclamation mark will appear again on the controller and the other work order is given. The procedure is the same. The puzzle must be solved by choosing and turning each pipe to make a circuit. When the puzzle is done the test is over.

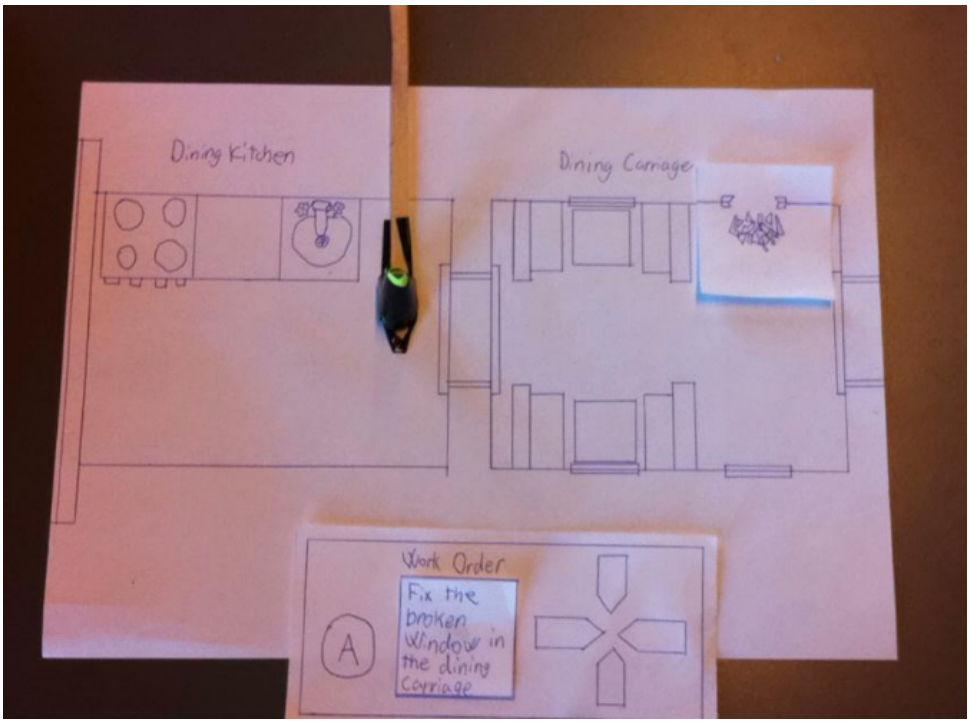


Figure 10.49: The setup of the paper mock-up of the game.

The collected responses from the participants, was put into a perceived affordance scheme, to see if it matches the one made by the group.

Objects	Feedforward	Perceived Affordance	Feedback
Left Button	This can be used to move something.	It is a button you can press, similar to a D-pad	Pressing this button moves something left
Right Button	This can be used to move something.	It is a button you can press, similar to a D-pad	Pressing this button moves something right
Up Button	This can be used to move something.	It is a button you can press, similar to a D-pad	Pressing this button moves something up/forward
Down Button	This can be used to move something.	It is a button you can press, similar to a D-pad	Pressing this button moves something down/backwards
Work Order Screen	Can show objectives, or the order in which something has to be done.	Cannot be pressed. It is a list.	The screen shows information
A "Action" Button	Chooses something, is a "confirm/interact" button.	It is a button you can press.	Pressing this button confirms/activates the presented action

Figure 10.50: The answers about the Main controller overlay collected in a perceived affordance scheme.

Objects	Feedforward	Perceived Affordance	Feedback
Left Button	This can be used to move something.	Something i can control. Similar to a D-pad.	Pressing this button moves something left
Right Button	This can be used to move something.	Something i can control. Similar to a D-pad.	Pressing this button moves something right
Up Button	This can be used to move something.	Something i can control. Similar to a D-pad.	Pressing this button moves something up/forward
Down Button	This can be used to move something.	Something i can control. Similar to a D-pad.	Pressing this button moves something down/backwards
A Button	Initiate, select or confirm something.	Press-able button.	Pressing this button confirms/activates the main action
B Button	Cancel button, or go back	Press-able button.	Pressing this button confirms/activates the secondary action
X Button	Closes the window/ leaves the puzzle.	Press-able button	Pressing this button exits the screen

Figure 10.51: The answers about the Puzzle controller overlay collected in a perceived affordance scheme.

10.9.3 Controller Survey

When the controller was being designed, an online survey was made to find the most intuitive design. The survey revolved around the work order list, and where it should be placed in the game. There are two different types of work orders: The individual work orders, which could only be done by a single player, and the shared work orders, which could be done by all players. The purpose of the survey was to find the most intuitive places to have the work order lists, be it separate or combined into one.

PC & Controller Design Survey

Hello.
We are 5. semester students from Medialogy Aalborg, and we are making a game for the platform AirConsole, which utilizes the use of a smartphone as a controller for your computer.

The game is a local multiplayer game with up to 4 players, who have to work together to solve problems as they occur throughout the game.

In this survey we would like to focus on the placement of the work orders that each player will have, and where on the screens they are placed.

There are both personal work orders for each player and there are shared work orders for all players. in this survey you will be showed different sketches of the different placements. Please answer the questions, and thank you.

*Required

Gender? *

- ○ Male
- ○ Female

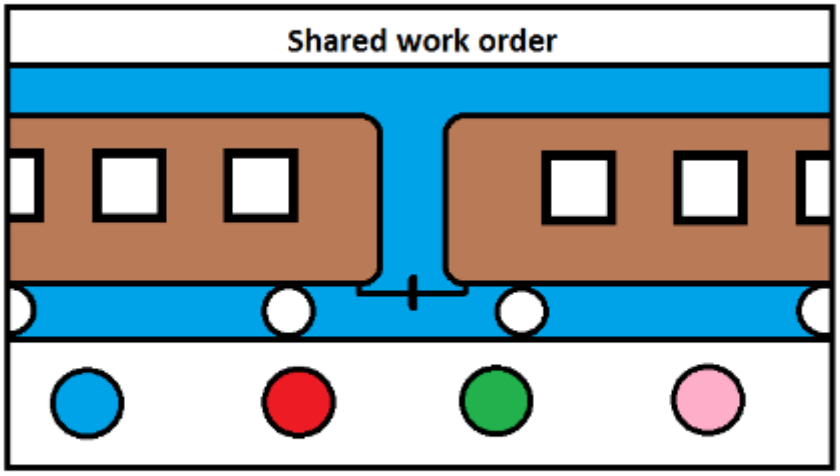
Age? *

Your answer

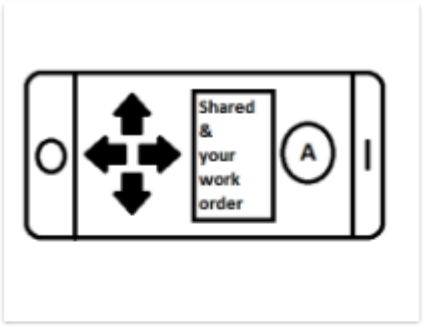
Figure 10.52: The first page of the survey, explaining the purpose and getting general information about the participants.

Here you see the first design of the screen. On this screen the shared work order is shown in the top of the screen.

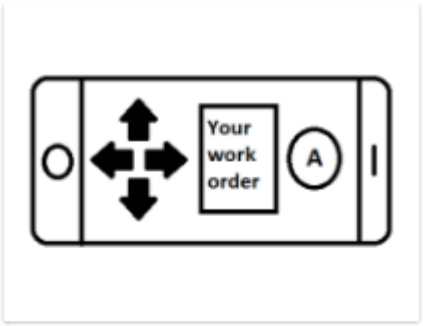
Design 1



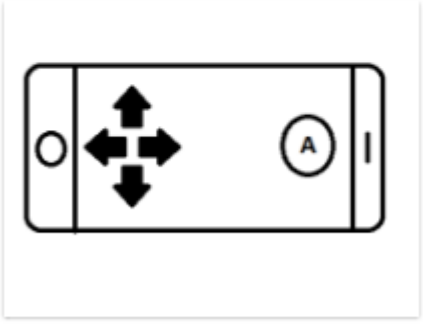
Please choose which controller designs you think would work best with the PC Screen design. *



Option 1



Option 2

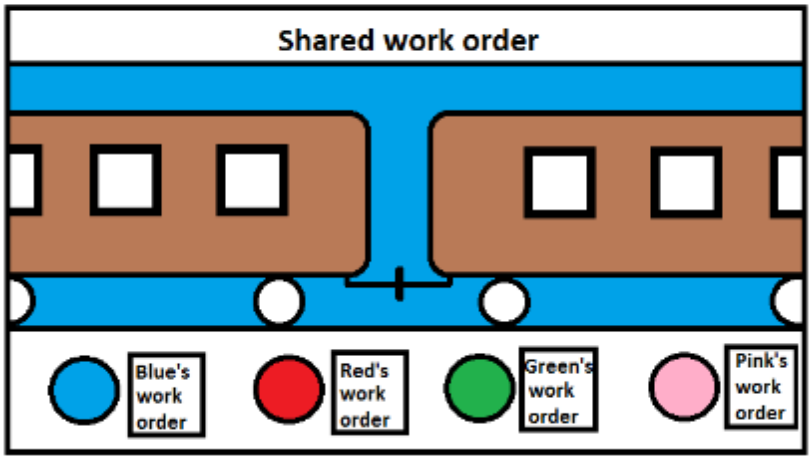


Option 3

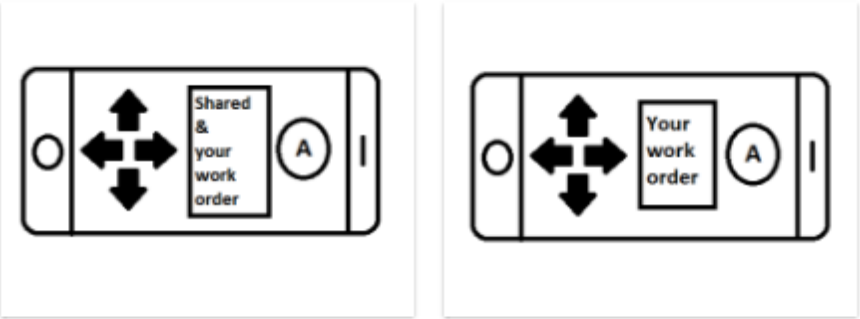
Figure 10.53: The first design of the main screen, where participants can choose which controller design is most intuitive.

Here you see the second design of the screen. On this screen the shared work order is shown in the top of the screen and the personal work orders are shown next to each player.

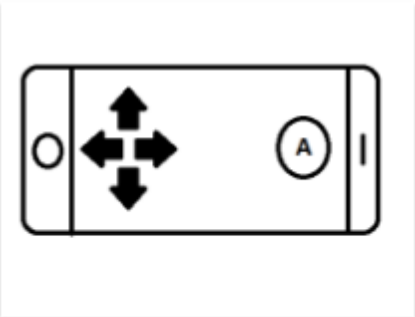
Design 2



Please choose which controller designs you think would work best with the PC Screen design. *



☐ Option 1 ☐ Option 2

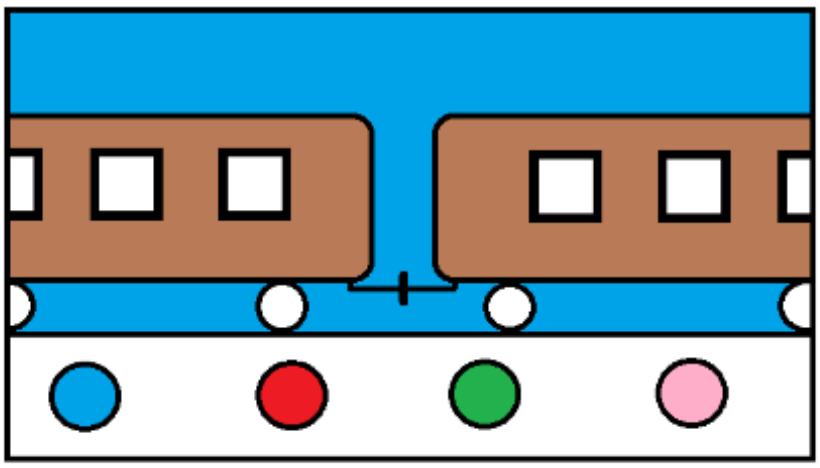


☐ Option 3

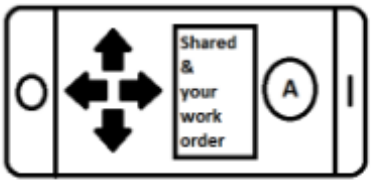
Figure 10.54: The second design of the main screen, where participants can choose which controller design is most intuitive.

Here you see the third design of the screen. On this screen no work orders are shown.

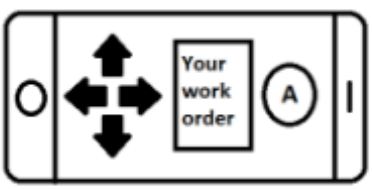
Design 3



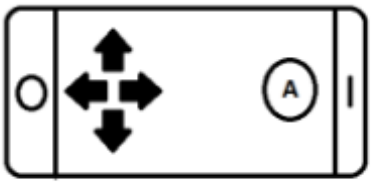
Please choose which controller designs you think would work best with the PC Screen design. *



Option 1



Option 2



Option 3

Figure 10.55: The third design of the main screen, where participants can choose which controller design is most intuitive.

This survey was posted on facebook in a group for testing, and it got 77 responses. the collected data was put into different charts.

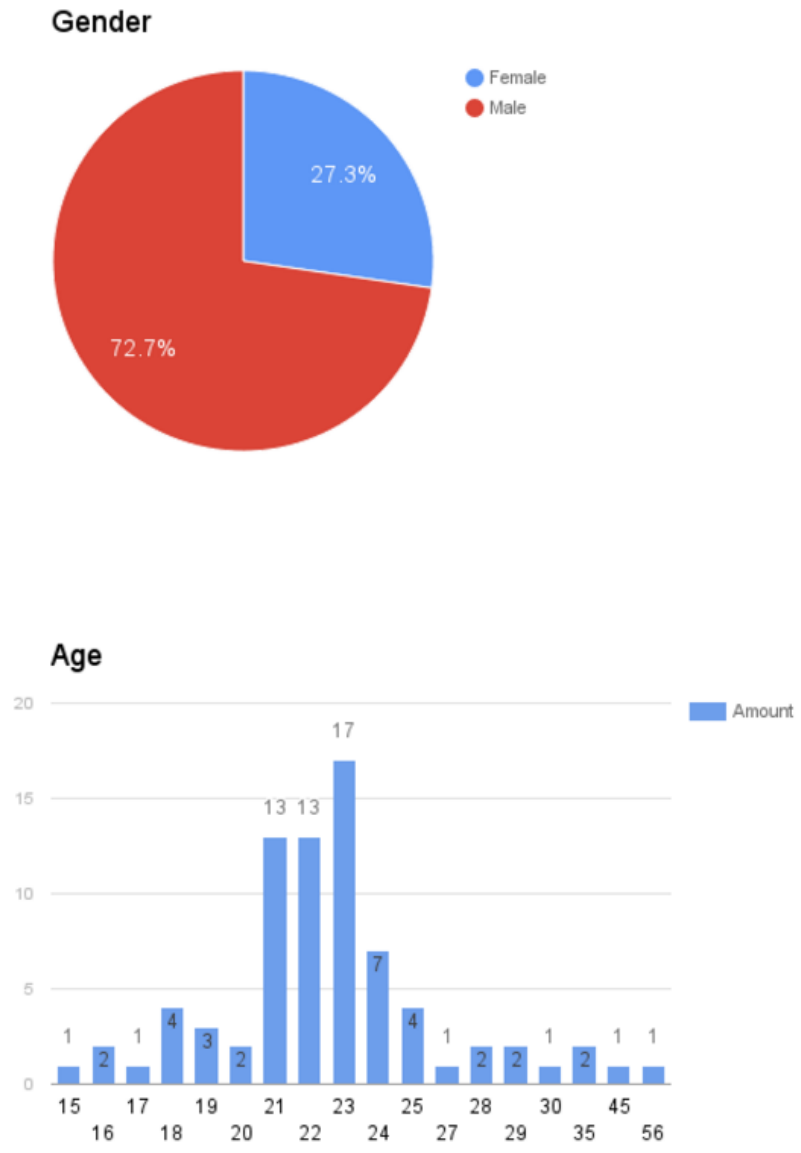
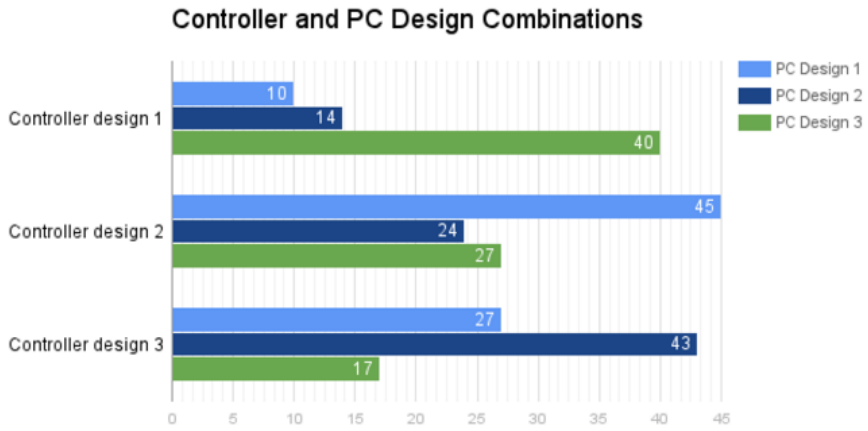


Figure 10.56: The gender and age ratio of the participants.



PC Design 1
Controller Design 1: 12.2%
Controller Design 2: 54.9%
Controller Design 3: 32.9%

PC Design 2
Controller Design 1: 17.3%
Controller Design 2: 29.6%
Controller Design 3: 53.1%

PC Design 3
Controller Design 1: 47.6%
Controller Design 2: 32.2%
Controller Design 3: 20.2%

Figure 10.57: The collected answers and preferred controller/screen combinations.

10.9.4 Internal Testings

An internal test was made in the when the first controller design was implemented. The purpose of the test was to test the optimization of the controller screen on different smartphones. The test was done with all the group-members' smartphones. When the controller was designed it was tested on a HTC One Mini2 2 smartphone, which for this test was used as a measuring point. For this test the controller was opened on all smartphones, and every element of the controller, including the size of the screen, was measured with a caliper. All the measurements were collected in a spreadsheet.

			Coordinates						
HTX One Mini 2	**Daniel**	Size (X*Y)	x	y		Daniels data is used as a measuring point			
	Up	12*13	11	34		Up size x:	100%	Workorder size x:	100%
	Left	12*13	3	20		Up size y:	100%	Workorder size y:	100%
	Right	12*13	28	20		Left size x:	100%	A Button size x:	100%
	Down	12*13	11	6		Left size y:	100%	A Button size y:	100%
	Workorder	20*34	43	11		Right size x:	100%	Screen size x:	100%
	A Button	23*23	71	16		Right size y:	100%	Screen size y:	100%
	Screen Size	99*56				Down size x:	100%		
						Down size y:	100%		
			Coordinates						
One Plus 3	**Bruhno**	Size (X*Y)	x	y					
	Up	15*16	19	42		Up size x:	125%	Workorder size x:	125%
	Left	15*16	4	25		Up size y:	123%	Workorder size y:	124%
	Right	15*16	34	25		Left size x:	125%	A Button size x:	126%
	Down	15*16	19	8		Left size y:	123%	A Button size y:	126%
	Workorder	25*42	53	13		Right size x:	125%	Screen size x:	123%
	A Button	29*29	81	20		Right size y:	123%	Screen size y:	121%
	Screen Size	122*68				Down size x:	125%		
						Down size y:	123%		
			Coordinates						
Iphone 4	**Mikkel**	Size (X*Y)	x	y					
	Up	9*14	12	31		Up size x:	75%	Workorder size x:	75%
	Left	9*14	3	17		Up size y:	108%	Workorder size y:	88%
	Right	9*14	20	17		Left size x:	75%	A Button size x:	104%
	Down	9*14	12	3		Left size y:	108%	A Button size y:	104%
	Workorder	15*30	32	10		Right size x:	75%	Screen size x:	69%
	A Button	24*24?	48	13		Right size y:	108%	Screen size y:	89%
	Screen Size	68*50				Down size x:	75%		
						Down size y:	108%		

Figure 10.58: The measurements of the controller elements in different phones, collected in a spreadsheet.

One Plus 1	**Anders**	Size (X*Y)	x	y		Up size x:	125%	Workorder size x:	125%
	Up	15*16	19	42		Up size y:	123%	Workorder size y:	124%
	Left	15*16	4	25		Left size x:	125%	A Button size x:	126%
	Right	15*16	34	25		Left size y:	123%	A Button size y:	126%
	Down	15*16	19	8		Right size x:	125%	Screen size x:	123%
	Workorder	25*42	53	13		Right size y:	123%	Screen size y:	121%
	A Button	29*29	81	20		Down size x:	125%		
	Screen Size	122*68				Down size y:	123%		
			Coordinates						
Iphone 5	**Thor**	Size (X*Y)	x	y		Up size x:	92%	Workorder size x:	90%
	Up	11*13	15	31		Up size y:	100%	Workorder size y:	88%
	Left	11*13	3	17		Left size x:	92%	A Button size x:	100%
	Right	11*13	26	17		Left size y:	100%	A Button size y:	100%
	Down	11*13	15	3		Right size x:	92%	Screen size x:	90%
	Workorder	18*30	40	10		Right size y:	100%	Screen size y:	89%
	A Button	23*23	61	13		Down size x:	92%		
	Screen Size	89*50				Down size y:	100%		
			Coordinates						
Samsung	**Motty**	Size (X*Y)	x	y		Up size x:	117%	Workorder size x:	115%
S5 Neo	Up	14*15	18	39		Up size y:	115%	Workorder size y:	115%
	Left	14*15	4	23		Left size x:	117%	A Button size x:	117%
	Right	14*15	32	23		Left size y:	115%	A Button size y:	117%
	Down	14*15	18	7		Right size x:	117%	Screen size x:	114%
	Workorder	23*39	50	12		Right size y:	115%	Screen size y:	113%
	A Button	27*27	75	18		Down size x:	117%		
	Screen Size	113*63				Down size y:	115%		

Figure 10.59: The measurements of the controller elements in different phones, collected in a spreadsheet.

10.9.5 Final Evaluation

The evaluation was made as a control group experiment, meaning that the participants were divided into groups where half of the groups played the game with visual ques and the other half played without. It was important to decide what to explain to the participants before the test, to keep up the validity. A list was made that states which things were to be explained.

What to explain to the participant before the evaluation?
The setting.
You are on a train, and you have to work together to keep the train running for 5 minutes.
During the trip, things will start to break down, and it is your job to walk to the broken things and fix them.
You have to stand in front of the thing to fix it.
You have five lives and if you fail to repair five things then you lose.
You are each assigned a character along with a color.
Some work-orders are shared by all and other are individual, and described with your color.
The controller. Each controller have a D-pad on the left of your controller.
You move by placing your finger and sliding it in the direction you want to move.
In the middle you have a work order list.
There you can see what tasks you can fix.
You can see both the shared tasks and your own individual task.
This means you have to work together and communicate between all players, to decide who does what.

You can see what element is broken.
On the right you have an interaction button.
You use this to interact with the workstations when you are in range.
You have to play two rounds of the game.

What not to explain?
Do not mention the position of the workstations, the participants need to find these themselves.
Do not mention or explain the visual cues in the game, as participants needs to discover and understand these themselves.
Don't mention what we are testing (visual cues) and don't mention anything about treatment/control groups.

Results

The collected results were put into separate tables for each session.

index1	Game1	gVar1	inputs
1	2.67	3.67	15
2	2.92	4.27	12
3	3	6.71	14
4	2.45	4.07	11
5	2.83	1.37	6
6	2	0	3
7	2.17	0.57	6
8	3.25	2.92	4
9	2.5	4.5	2
10	2	3	3
11	1.33	0.33	3
12	3	8	2
13	1.67	0.33	3
14	1		1

index2	Game2	gVar2	inputs
1	1.35	0.37	17
2	2.29	1.91	14
3	2	1.85	15
4	1.71	1.3	14
5	1.64	0.45	11
6	2	2.92	14
7	2.1	3.09	11
8	1.92	1.41	13
9	2.31	6.56	13
10	2.33	2.06	12
11	2.21	2.95	14
12	1.5	0.5	10
13	1.64	0.85	11
14	2.17	2.17	6
15	2	2.33	7
16	2	1	3
17	2.83	4.57	6
18	2.75	4.92	4
19	3.5	1.67	4
20	1	0	2
21	3	1	3
22	2	2	2

Figure 10.60: The collected results of the treatment groups. Index is the order, Game is the Mean, gVar is the variance, inputs is the number of inputs.

index1	Game1	gVar1	inputs
1	3.18	7.53	17
2	2.69	3.06	13
3	3.27	3.64	15
4	3.57	6.11	14
5	2.7	2.46	10
6	3.09	4.49	11
7	3.57	11.29	7
8	3	4	9
9	4.43	7.29	7
10	3.13	4.41	8
11	5.8	7.2	5
12	1.33	0.33	3
13	2.5	0.5	2
14	3	0	2
15	2	0	1
16	2	3	3
17	1	0	2
18	2	0	1
19	2	0	2
20	4	12	3
21	3.5	12.5	2
22	4	8	2

index2	Game2	gVar2	inputs
1	1.69	1.16	16
2	1.79	1.41	14
3	2.42	1.15	19
4	2.5	1.35	14
5	2.5	2.45	12
6	3.5	7.71	8
7	1.6	1.82	10
8	2.06	0.93	17
9	2.45	3.07	11
10	2.31	2.23	13
11	1.6	0.8	5
12	3	6	7
13	3.2	3.96	10
14	2.25	2.79	8
15	1.75	1.07	8
16	2	1.6	5
17	2.25	1.64	7
18	2.5	1.9	5
19	1.48	1.86	6
20	3.67	11	8
21	3.56	4.03	8
22	1.5	0.33	4

Figure 10.61: The collected results of the control groups. Index is the order, Game is the Mean, gVar is the variance, inputs is the number of inputs.

When the participants were done playing the game, they were given a questionnaire about the game.

Questionnaire:

Mark with an X, in the slot with your answer, where the numbers are:

Disagree	Somewhat disagree	Indifferent	Somewhat agree	Agree
1	2	3	4	5

	1	2	3	4	5
It was easy to find and identify my character:					
It was easy to navigate and move around the train:					
I understood the tasks window on my controller:					
It was easy to keep track of the currently active tasks:					
It was easy to find the workstations in the train:					
It was easy to know when new tasks were active:					

Figure 10.62: The questionnaire given to the participants at the end of the evaluation.