Flow and Companions in video games

Adam Victor H. Nørgaard, Anders Møker, Simon J. Petersen & Kasper V. Rasmussen

Adam Victor Hansen Nørgaard

_____

Anders Møker

_____

Kasper Vestergaard Rasmussen

_____

Simon Jacaranda Petersen

_____

Abstract

The project aims towards finding an answer to whether or not there is a difference between two versions, one with a companion with artificial intelligence and one without a companion, in a self made interactive adventure game, where the test participants have to find a way to get away from an island. The report concerns itself with theoretical work regarding flow theory, GameFlow, and game design. These subjects were used in the creation of the game with which tests were conducted, in order to investigate the aforementioned project aim. Furthermore, the design phases ranging from building, designing, and coding of the game itself will be explained in detail throughout the report. The created game was tested on 42 test participants, and they had to answer two questionnaires, one profiling questionnaire and a flow questionnaire to gather data, so statistical analysis could be made. Tests for normal distribution showed that the data from one group was parametric, while the data from the other group was not. The Mann-Whitney U analysis method was used, and it revealed that there was no significant results.

Keywords: Flow, GameFlow, Video Games, Game Design, Companion

Resumé

Projektet sigter mod at finde et svar på om der en forskel mellem to video spil-versioner, en version med en følgesvend med kunstig intelligens og en uden en følgesvend, i et selv produceret interaktiv eventyr video spil, hvor test deltagerne skal finde en måde at undslippe en ø. Rapporten beskæftiger sig med teoretisk arbejde vedrørende flow teori, GameFlow og spil design. Disse emner blev brugt til produktionen af spillet hvor test blev udført, for at undersøge det førnævnte projektmål. Designfasen, rangerende fra opbygning, designet og programmeringen af selve spillet vil ydermere blive forklaret i detaljer i gennem rapporten. Spillet, som blev produceret blev testet på 42 deltagere, som skulle besvare to spørgeskemaer, et profil spørgeskema og et flow spørgeskema for at samle data, så der kunne laves en statistisk analyse. Test for normalfordeling viste, at dataen fra den ene gruppen var parametrisk, imens dataen fra den anden var ikke parametrisk. Mann-Whitney U analysemetoden blev brugt, og den afslørede at der ikke var nogen signifikant resultater.

Nøgleord: Flow, GameFlow, Video Spil, Spil design, Følgesvend

Acknowledgements

We would personally like to thank Andreea Jugănaru for helping us bring Wanda to life through voice acting.

Table of content

1. Introduction

Game development has expanded into a lot of different platforms (Fullerton, T., 2014, p. xv), and the gaming industry is now bringing in more money than the movie industry (Wallop, H., 2009). Additionally, games' abilities to communicate stories are quickly evolving, according to Jim Lee (as cited by Rogers, V., 2010). This is a testament to game development being an ever evolving industry (Fullerton, T., 2014, p. xv). In game development, professionals work in several different key areas, e.g. programming, visual arts, marketing, and sound design (Fullerton, T., 2014, p. xxvi). These areas may be prioritized differently depending on each studio's work processes and ideas. During development of a game, many decisions have to be made, which can make the process somewhat challenging (Schell, J., 2014, p. xxxviii). Games can be a big part of a person's identity, and some people spend hours upon hours playing video games. During gaming, people can reach deep emotional and immersed states of mind, and eventually lose their senses of time, senses of external pressure and other traits (Chen, J., 2007, p. 37). These traits are part of a psychology theory known as flow theory, developed by Mihalyi Csikszentmihalyi. The flow experience refers to a deep but effortless state of immersion, and can help people derive great pleasure from activities, such as games.

This project focuses on investigating GameFlow theory by Penelope Sweetser and Peta Wyeth (2005), and how flow of the player can be affected by an accompanying artificial intelligence also called A.

I. In order to investigate this, a game with two different versions was created. This game is called "Stranded". One version had no player-accompanying AI For the other version a robotic companion with AI was designed, created, and implemented. The game was developed in the Unity3D game engine. All programming was done in C#, and the game assets were created in a selection of 3D and 2D graphical software packages. The study revolved around having two test groups, one for each version of the game. A between-groups experiment was set up to see if the presence of the companion had any influence on the flow of the players.

2. Research Question

How can an interactive adventure game, which can help determine if there is a difference between flow levels with and without the presence of an interactive companion, be developed?

How can it be determined if there is a difference?

3. Hypothesis

For this project, the following hypothesis was set up: “A companion with artificial intelligence will have a positive effect on flow in Stranded”

4. Theory

The different theories used during the course of this project will be discussed in depth in the following section.

4. 1. Flow

The following sections will surround the subject of flow, what it is, the changes made to the flow graph, and how flow has been adapted into games.

4. 1. 1. What is flow?

In the 1960’s, Mihalyi Csikszentmihalyi observed hundreds of painters working on their canvases. What he discovered was how intensively the many painters sometimes worked on their paintings. It seemed as if the painters were in complete harmony with the task in front of them. The painters did not notice what was going on around them, and they lost track of time and place. Furthermore, while the artists were painting they did not experience the need for rest or food (Snyder, C. & Lopez, S. J., 2002, p. 89). The concept of flow originated from Csikszentmihalyi’s observations of the painters in the 1960’s. By conducting interviews with different test participants Csikszentmihalyi discovered that it was the given task at hand that made the artists enter the state of flow. It was not about the outcome, or for the fame, or for the money. The artists were purely in it for the act of creating the painting, and because of the challenge of creating

something that were aesthetically pleasing for them. That was the important part. However, the painters were not always able to bring forth their visions and bring them onto the canvas. Not being able to create the perfect painting served as an even greater challenge, which allowed the artists to sink further into flow. The fact that it was necessary for the artists to have a challenge that was not too easy to complete, and the challenge was not too hard to comprehend inspired Csikszentmihalyi to develop a system. The system developed could explain the different states of the mind in the context of performing a given task, and getting into the state of flow. The system can be seen in figure 1.

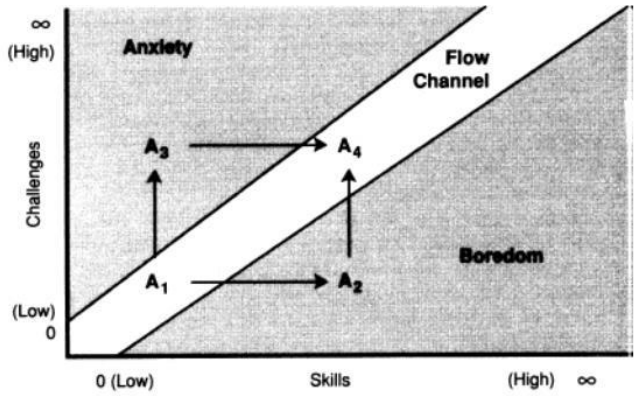

Figure 1. The figure shows the original concept of flow, and the different states (Csikszentmihalyi, M. 2008, p. 74).

Through Csikszentmihalyi's discoveries it was found that there needs to be a balance between the challenge level of the given task which needs to be performed, and the level of skill the person possess. When this balance is achieved the person is able to enter the flow channel, which can be seen in figure 1. If the given task is too difficult compared to one's skill level a feeling of anxiety will be perceived. On the other hand, if the task is too simple, and the person's skill level surpasses the challenge level, a feeling of boredom will be perceived. To make it easier to understand how, and what is needed to enter this flow state, Csikszentmihalyi defined eight different dimensions of the flow experience.

The eight defined dimensions of the flow experience are: clear goals and immediate feedback, equilibrium between the level of challenge and personal skill, merging of action and awareness, focused concentration, sense of potential control, loss of self-

consciousness, time distortion, and autotelic or self-rewarding experience. The eight dimensions are not going to be explained in detail, because this report is working with the concept of GameFlow as well. Within GameFlow the eight dimensions will be explained.

4. 1. 2. Re-iteration of the flow graph

Eventually, changes from the original flow graph were made, which can be seen in figure 2. Studies were done by a Milan based group, which was led by one of Csikszentmihalyi's colleagues, Massimini. The studies showed that the original flow model was not consistent with the mapping. Simply having a balance between skill level and challenge level was not good enough to optimize the quality of experience, i.e. activities providing minimal opportunities for action does not lead to flow (Snyder, C. & Lopez, S. J., 2002, p. 94). As an alternative, the Milan group developed another flow state model, which shows that in order to achieve the flow state; you need a certain level of challenge, and a certain level of skills. This alternative model can be seen in figure 2.

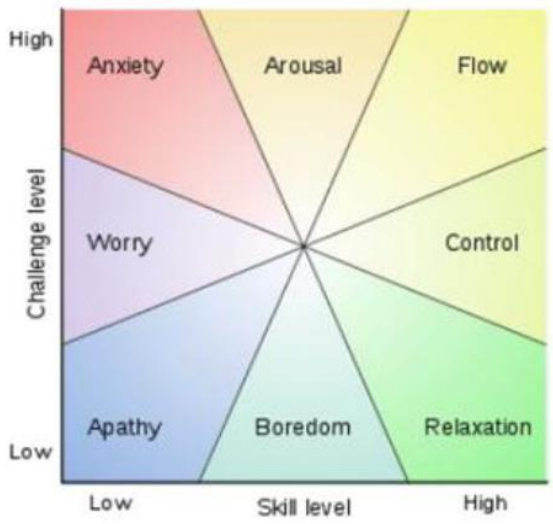

Figure 2. The current model of flow state (Snyder, C., & Lopez, S. J. 2002, p.95).

What figure 2 shows is that a person that could experience flow with relatively low skill level, and a low challenge level is now substituted with the apathy state. Furthermore, flow is experienced if the perceived challenges and skills are higher that the person's average level. On the other hand, according to a speech by Csikszentmihalyi at a TED

Talk in 2004, the ‘arousal’ and ‘control’ states are not the worst states to end up in. The reason for this is that a person can easily move from these states into the flow state (Csikszentmihalyi, M., 2004). If an individual is currently within the ‘control’ state, then adjusting the level of the challenge could have an influence on the individual, which could lead the individual into the ‘flow’ state. On the other hand, if the individual is within the ‘arousal’ state, then acquiring a higher level of skill or competences would in theory direct the individual into the ‘flow’ state.

4. 2. GameFlow

The concept of flow created by Csikszentmihalyi has been altered to better fit within games, and especially video games. The eight dimensions from the flow theory have been slightly altered, and the reasoning behind this is so that it would fit better into the video games genre.

4. 2. 1. Adapting flow to games

Sweetser and Wyeth conducted a broad review, with information gathered from different literature on the usability and the user experience in games. The reasoning behind this was to determine how the eight different elements of flow are noticeable in video games. They assembled a model of enjoyment in games from the findings in the literature, based on the elements of flow and the evidence of flow experiences in games. The result was the GameFlow model, which consists of eight core elements, and they are as follows (Sweetser, P., & Wyeth, P., 2005):

- Concentration
- Challenge
- Skills
- Control
- Clear goals
- Feedback
- Immersion
- Social Interaction

Most of the eight core elements can be related back to Csikszentmihalyi's eight elements of flow, and this can be seen in table 1.

Table 1. Mapping the elements found in the game literature to the elements of flow (Sweetser, P., & Wyeth, P.).

Games Literature	Flow
The Game	A task that can be completed
Concentration	Ability to concentrate on the task
Challenge Players Skills	Perceived skills should match challenges and both must exceed a certain Threshold
Control	Allowed to exercise a sense of control over actions
Clear Goals	The task has clear goals
Feedback	The task provides immediate feedback
Immersion	Deep but effortless involvement, reduced concern for self and sense of time
Social Interaction	n/a

What can be seen in table 1 are the different elements of flow which are categorized, and the subjects will be explained in the paragraphs below. The first element of flow, i.e., a task that can be completed is not directly represented in the GameFlow elements. The explanation for this is that the task, which needs completion, is the game itself. For a game to be enjoyable it is required of the player to use their concentration, and the player must be able to concentrate on the game. The more concentrated the player has to be on the task, meaning their attention and workload, the more absorbing the game will become. When all of a person's relevant skills are needed to deal with the challenges of a certain situation, the person's attention is completely absorbed by the activity (Sweetser, P., & Wyeth, P., 2005).

4. 2. 2. Concentration.

A game should grab the player's attention quickly and it needs to maintain the attention throughout the game. The game should hold the attention for 10 seconds, 10 minutes, 10 hours, or even at 100 hours of play. The way games can capture the player's attention could be by providing something worth attending to, e.g. a detailed game worlds to draw in the player. To increase the player's workload is an in important aspect; however, the workload should be maintained in an appropriate level within the player's limit. Another aspect which is important for the concentration is to not burden the player with tasks that do not seem necessary or important. Lastly; to help the player's keep their concentration there should be a minimization of distractions from major game tasks, or they should be removed completely. This could be done by reducing the non-game-related interactions within the game. Furthermore, reducing the amount of buttons or other distractions from the graphical user interface could help maximizing the amount of screen taken up by the game action (Sweetser, P., & Wyeth, P., 2005).

These different things have been considered when designing the game for this project, because the test participants should use their concentration on the game and not become distracted by the interface of the game.

4. 2. 3. Challenge

When creating favourable game design, challenge is identified as one of the more important aspects. Games should be sufficiently challenging, here meaning that the game needs to match the player's skill level, vary the level of the difficulty, and keep an appropriate pace (Sweetser, P., & Wyeth, P., 2005). What is meant by pace here is the rate at which players experience new challenges, and explore different areas within the game world.

An important precursor of flow is that there is a need for balance between the player's skill and the challenges with the specific activity at hand. Games should be designed to have a level of challenges that is appropriate and not discouragingly hard, or boringly easy (Sweetser, P., & Wyeth, P., 2005).

The way games are creating enjoyment to the players is by challenging them, and this is often by pressuring and pushing the player to the limit of their memory and performance. The games need to present the players with an appropriate sequence of distinct and challenging situations, which are calculated from careful level and obstacle design, and provide a positive game experience, which makes the players continually wanting to play the game. The satisfaction in games comes from finishing these difficult tasks you as a player is presented with, challenges and surpassing the opponents, testing your skill level, mastering new skills, reaching a desired goal, and coping with the different dangerous situations. The rewards from conquering the many challenges the player is presented with are all intrinsic, and the entire process on its own is the reward. This could be something as the feeling of personal triumph (Sweetser, P., & Wyeth, P., 2005). So even though players have different skill levels, the game at hand should still feel do-able and that the players' actions and efforts are paying off. According to Sweetser & Wyeth (2005) most games will allow the players to choose an appropriate level of difficulty, to help regulate their probability of success or failure within a competitive situation according to the player's skill (Sweetser, P., & Wyeth, P., 2005). Furthermore, the pace is an important aspect of the challenge. The pacing here will maintain the appropriate levels of challenge and tension throughout the game. The suitable amount of pace within a game will apply pressure, but it does not frustrate the players (Sweetser, P., & Wyeth, P., 2005).

4. 2. 4. Player Skills

For a game to be enjoyable, the developers must support the players' skill developments and masteries. The way players are introduced to games, and the ways the players are taught to play games are crucial to their skill development and enjoyment of the games. The players should be taught how to play the games through absorbing and interesting tutorials, which will allow the players to become involved fast with the tasks at hand. A way to do this could be by using an in-game tutorial with feedback to learn the basic mechanics, or an alternative to the tutorial could be to let the players learn as they play instead. If the developers go with the 'learn as you play' method, the players get to practice their skills or abilities as a part of completing tasks they need to do. Handing out rewards is also an important part in learning to play a new game. Players must be

rewarded appropriately to continue to play; the effort they invest within a game should be equal to the reward gained from succeeding the tasks or mission (Sweetser, P., & Wyeth, P., 2005).

When the players are trying out the game for the first time there should be enough information given about the game. The players should not need or expect to have to use a manual before starting up the game. If the players have to read through long explanations it could bore the players. Additionally, the game manuals should not be too long, complex, or overemphasise game story at the cost of what can be done and how. Nevertheless, there should not be given help through in-game dialogue, as real-world things attached to the game world characters causes immersion to break. An example of this could be an in-game character telling the player "to open the hatch press X in the control pad", since the in-game character is supposed to live in the game world, that character should not know anything about the controls of the game (Sweetser, P., & Wyeth, P., 2005).

4. 2. 5. Control

In order to experience flow, players need to exercise some sense of control over the actions available for their in-game character. The players should be able to sufficiently convert their intentions into in-game behaviour, and this should give them a sense of control of the actual movements of the character. This should help with the manner in which the players can explore the many environments. Furthermore, it should be easy for a player to interact with the different in-game objects, which is needed to complete the goal of the players. It is also important that the players feel in control of the game's interface and game controls. This means that the in-game controls should not be too advanced, but be kept simple for a faster learning curve where there are some core sets of buttons to help increase the sense of control. There should also be an option to expand to more advanced settings. The core buttons should have an option to be customisable for the player, and the gameplay should fit their learning and playing style. Otherwise the game needs to be designed differently, so it allows for different styles of learning and play style (Sweetser, P., & Wyeth, P., 2005).

The game should be easy to use, and allow the players to start the game whenever they like, meaning it should be easy turning the game on and off, with the ability to save the game at different stages. With these different capabilities added to a game it would give the player a sense of control of the game menu, and give players freedom to explore the game at their own pace. When the player is using the game menu it should be easy to use, it needs to be intuitively organized, and it should not sacrifice readability and functionality for aesthetics. When or if an error occurs in a game it can make the players feel as if they have lost control, especially if the errors are out of the control of the player. The players should not be able to make mistakes that stop the game from working correctly (Sweetser, P., & Wyeth, P., 2005).

4. 2. 6. Clear Goals

When playing a video game the game needs to provide the player with clear goals of what is needed to be done; this should happen at appropriate times. Games must have some sort of objective or goal that needs to be completed, but to achieve flow, the goals must be clear. When you start playing the game the players should be presented with the clear overall goal, or at least after a few minutes of playtime, which could be done through an introductory cinematic that establishes the background story. Within each level, or areas that can be visited, there should be multiple goals and games often use small briefings to give information about a certain mission, to create an outline of the different goals (Sweetser, P., & Wyeth, P., 2005).

4. 2. 7. Feedback

When moving around within a game the player will receive feedback, but it should be the appropriate amount of feedback, and at the right time. During flow concentration is possible because the different tasks are giving the immediate feedback needed. Video games need to provide regular in-game feedback for the players, and this is done to help determine the distance and the progression towards the many objectives. The players need to get feedback on their progress, and when they fail they need the correct feedback about whether they are moving in the right direction. A way to give the players feedback could be by implementing a scoring system, and this system could

provide positive feedback to encourage mastery of the game, as well as always be able to identify the status in the game. In-game interfaces and sounds can be helpful tools to deliver necessary status feedback. A way the sounds could provide feedback could be by changing when entering a new area, or perhaps if enemies are approaching (Sweetser, P., & Wyeth, P., 2005).

4. 2. 8. Immersion

When the players are in-game they should experience a deep, but effortless involvement in the game they are playing. In flow there is an element describing immersion as a deep but effortless involvement, where there is a chance for a loss of concern for self, everyday life, and an altered sense of time. The deep, but effortless involvement is frequently reported by gamers and observers of gameplay, where the players become less aware of their own surroundings and less self-aware than what they were previously. There are many gamers whom are devoting their entire night or weekend into playing a specific game, or many different games, without being concurrently aware of doing so or consciously deciding to do so. When gamers are playing for such a long time they often have many emotions invested in the certain game because of all of the time, effort, and attention they have put into playing. When this is going on the games end up becoming the most important part of their attention, which could make their emotions directly affected by the game (Sweetser, P., & Wyeth, P., 2005).

Some gamers are playing these games to make them think and feel certain emotions, which are not necessarily related to their work. The reasoning behind this could be for the player to calm down after a hard day of work, or perhaps to escape from everyday worries. Games are typically seen as a way of escaping the real world or get away from social norms, or it can be a way for people to do things in which they otherwise lack the skills, resources, or social permission to do. The games should move the players into a level of personal involvement emotionally. The different games played should make the players forget that they are in another medium, so the on-screen interface ends up becoming invisible or they simple do not notice that it is there. A game should entice the player to become immersed in the experience at hand. An immersive game is going to draw the players deeper into the game and it will affect their senses through elements

such as audio and the narrative. The audio can be the sound effects when something is happening, or it could be the soundtrack playing in the background. These audio elements are important for drawing the players into the games and it will help keeping them immersed. A way to immerse the players could be by creating in introduction and a storyline, which is going to explain who they are and what they need to do. This is a way to make the players feel as if they are part of the story (Sweetser, P., & Wyeth, P., 2005).

4. 2. 9. Social Interaction

Games need to support and create different opportunities for social interaction. Social interaction is not an element within the flow theory, and there is a chance that it will interrupt immersion in games, as real people provide a link to the real world, which could end up removing the players from their gaming worlds. However, it is also a strong element for enjoyment in games, as the users might play games for some social interaction, whether or not the players like the specific game they are playing (Sweetser, P., & Wyeth, P., 2005).

For a game to support social interaction it should create different opportunities for player competition, cooperation, and player connection. If the game is a multiplayer game then it should be structured so it enhances the player-to-player interaction and create enjoyment in playing with others in the game world, or outside of the game. Gamers usually enjoy interacting with other gamers, spending time with friends, watch others play, chatting and talking about the game, seeing other gamers' reactions and expressions, gloating when beating a friend, or feeling pride when they win. One of the ways a game could support this social interaction could be through a chat system or creating online score boards, because a large part of the attraction of online games is this new virtual community, the associations, and gradually over time improvements of your character. Another way to support the social interaction could be by creating social competition, because players experience a certain amount of satisfaction from competing against and beating other players (Sweetser, P., & Wyeth, P., 2005).

4. 2. 10. In the zone

As people are attracted to different things and game genres, it can make it difficult to reach the right audience by only using the work of Sweetser, P. and Wyeth, P., (2005) to create the GameFlow elements. Different players have different skills and skill levels, therefore those players are expecting different challenges. In figure 3 a red line is shown, and this line is the single narrow static experience that most games offer (Chen, J., 2007).

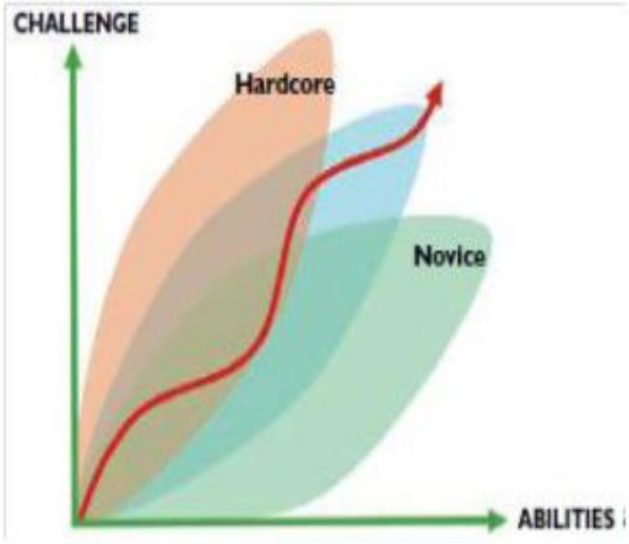

Figure 3. Different users have different flow zones (Chen, J., 2007).

This red line in figure 3 might keep a casual player in flow, but the hardcore or novice gamer is not going to experience the same enjoyment with the game. The reason for this is because of the given challenges are either going to be too easy or too difficult.

One example Chen, J., (2007) gives about this is that if a player is making a simple action of moving the camera view in a 3D environment, then there is a chance that this could end up becoming frustrating for a casual gamer who is used to playing games in 2D. There is a chance that the casual gamer could have enjoyed the rest of the game; however, now they have become too frustrated with how the camera is handled that they will turn away from the game.

So in order to reach a broader audience, the design of the interactive experience cannot be the same for all players. To reach this broader audience the experience needs to offer

many different choices, because it will help with adapting to the different users' personal flow zones, and this can be seen in figure 4.

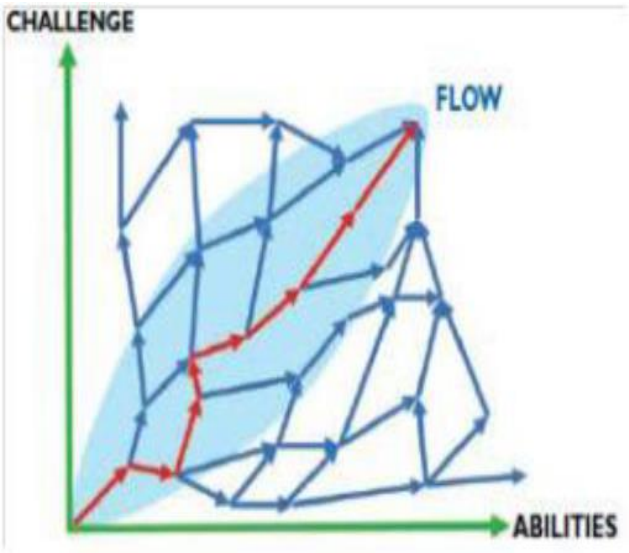

Figure 4. Designers adapt the users' flow experience through the choices they deliberately built into the experience (Chen, J., 2007).

However, it is not enough to "just" populate the experience with many different choices, the designers have to come up with other ways to do this. In video games, just randomly increasing the different number of choices the users have can be expensive. If the players are going to be presented with too many choices they might become overwhelmed, and when the users cannot decide what to choose, they are at a loss. When the players are being bombarded with many different choices there is a chance the players will become annoyed, and it could end up interrupting the gameplay more than it has to. This will interfere with the user's sense of control and concentration as well, which are important parts in both flow and GameFlow (Chen, J., 2007).

5. Related Work

The following section will be covering the different related work found for this project, and explain why it was seen as something that could be useful.

5. 1. Portal 2

Portal 2 is a first-person puzzle-platform video game, where the player is working with a portal gun to solve these puzzles. Portal 2 was developed by Valve Corporation and

released in 2011. Portal 2 was an inspiration for how to design the companion for the project. In Portal 2, the player moves through different levels, called test chambers, where the player has to solve puzzles to move forward. Whilst solving the many test chambers the player is accompanied with a follower, Wheatley, and his job is to bring information and add a funny aspect to the game. This was an inspiration for creating a companion to add into the project. The companion had to be both witty and hand out hints to where the player should go.

In Stranded, whenever the player enters a new area, the dialogue options of the companion changes, offering a dynamic interaction based on the area that the player is in. The player can talk to the companion at any time. This dialogue mechanic is however not available in Portal 2. Instead the companion in Portal 2 will begin talking if the player is standing still for too long. This mechanic have been somewhat replicated for the project; meaning that every now and then the companion will make a comment. There are multiple comments for the companion to choose from, and they will play when entering a new area of the game.

5. 2. The Witness

The Witness is a 3D puzzle video game developed by Thekla, Inc. and released in 2016. In The Witness the player is exploring an abandoned island and solves puzzles, which are spread throughout the island. The Witness' art style was an inspiration in the design process of Stranded. The art style of The Witness can be seen in figure 5.

Figure 5. An example of the art style used within the game The Witness (Retrieved from https://www.playstation.com/en-us/games/the-witness-ps4/).

As shown in figure 5 the game is created with a low polygon art style, and the reason for this is to make it look slightly more cartoonish. This stylized art style was an inspiration for Stranded, and through development a similar style was implemented.

5. 3 Difference in A.I Companion Interactivity

5. 3. 1. Portal 2: GlaDOS & Wheatley.

In terms of interaction, the player does only directly interact with Wheatley twice, and GlaDOS once. Other than that, the player does not interact much with them. When either character is a companion of the player, they behave as plot devices, creating no real relationship for the player, other than a means to further the plot. In figure 6, both GlaDOS and Wheatley can be seen.

Figure 6. Left: GlaDOS. Right: Wheatley. Both are from the Portal 2 video game.

5. 3. 2. BioShock Infinite: Elizabeth.

In the video game BioShock Infinite, released in 2013, developed by Irrational Games and published by 2K Games, the player's objective is to save a woman named Elizabeth Comstock. When Elizabeth have been saved, the player and her travel through the floating city of Columbia, battling their way through various dangers. The level of interactivity is a mixture of Elizabeth being a plot device, and also directly helping and interacting with the player. When battling in the games, Elizabeth will sometimes contact the player, aiding the player by throwing cash, food, medical kits, ammunition and Salts, salts being used to power vigors, a magic type of weapon based on science fiction biologi. Elizabeth also possess the power to make portals to other dimensions using powers of quantum mechanics. These portal can create ammunition caches, gun turrets etc. When playing the games, Elizabeth will also hint to the player what must be done to progress, but also sometimes talk with the players, having small sections of dialogue. The character of Elizabeth can be seen in figure 7.

Figure 7. Elizabeth Comstock from Bioshock Infinite.

5. 3. 3. Mass Effect 1, 2 & 3.

The Mass Effect games were released in 2007, 2010, and 2012 respectively, developed by BioWare, and distributed by Electronic Arts except for the first game which was distributed by Microsoft Game Studios. The main series surround the player character Sheppard who has been tasked with ridding the galaxy of the A.I's. known as Reapers. Throughout the games the players get a large squad of different characters, that the player can fully interact with. When in combat the player can control the characters that are currently with him/her, choose which enemy to attack and which abilities to utilize.

When the player is not on a mission, the characters are collected in a hub, in the game series it is a starship, serving as the main location in the games, as well as the place the player can interact with the other characters, customize them and go on missions. When the player interacts with the different characters, they can build up relationships, both in positive and negative ways. The player can eventually be so connected with a character that they can start a romantic relationship with them. The interaction with the characters carries over in the different video games, so that the narrative changes from the interaction the player's choices. In figure 8, many of the characters from the video game series can be seen.

Figure 8. Many of the characters and companion that appear in the Mass Effect video game series (Retrieved from http://www.unigamesity.com/wp-content/uploads//2013/11/mass-effect-3-1920x1080_2.jpg).

5. 3. 4. Level of Interactivity

From the games discussed, a table of the level of interactivity can be created. The table includes Wanda from Stranded. On the interactivity table, there are 3 different interactions, these will be explained. It is important to note that these points are defined specifically for this project, and are not established in interaction theory:

- **Direct interaction with player.**
  How the companion interacts with the player, in terms or inter-character development, such as dialogue and relations.

- **Interaction effect on gameplay.**
  How the companion affects the game, by either helping with combat or giving hints to progress the story.

- **Interactionary effect on the story.**
  How the companions in the game can affect the overall outcome of the game.

With these definitions in place, a table was created to visualize the difference between each game. The comparisons can be seen in table 2.

Table 2. The difference of interaction with a companion in different games.

Interactivity Table	**Portal 2**	**BioShock Infinite**	**Mass Effect series**	**Stranded**
Direct interaction with player.	Extremely little.	Helps the player by giving aid, ammunition and cash.	Full interaction, both in combat and off-mission.	The player is able to initiate conversation with the companion.
Interaction effect on gameplay.	Extremely little, does come with a hint if player is stuck.	Helps the player during hard combat, and with hints.	Player have to equip companions with weapons and abilities. Will also have to use companions for tactical advancement.	None
Interactionary effect on the story.	None, companions only used as plot devices.	None, companion does not affect the story, is a plot device.	Ultimate relationship status with characters will affect the total outcome of the different games in the series. Also status can create new missions.	Little, the companion can make the game faster to complete if the user forgets the objective and reminds the user.

6. Methods

6. 1. Target Group

The participants used during the testing phase were students at AAU Esbjerg, but also users from various gaming forums. The criteria for the participants were able to use a mouse and keyboard setting or a gamepad setup, and be above the age of 12 for a sufficient level of understanding of the English language.

6. 2. Setup

The experiment was conducted on 12 participants from AAU Esbjerg and 31 from various selected forums. The participants would use a computer with a mouse and keyboard setup. After each test session the participant would have to answer two questionnaires, the flow questionnaire and a profiling questionnaire.

6. 3. Data Gathering

6. 3. 1. Flow Questionnaire

A study supported by the Ministry of Education of the People's Republic of China and conducted by Li, J.S., Xu, H., & Chen, S.S. researched the learning outcome within a game called "Second Life". The way this was researched was by looking into how the user's flow and motivation affected this learning outcome (Li, J.S., Xu, H., & Chen, S.S., 2012). For the experiment Li, Xu, and Chen (2012) had 113 students play the game Second Life, and then having the test participants fill out a questionnaire.

The reason that Li, Xu, and Chen had the students play a videogame is because before creating the study they had collected 239 samples from 50 technology companies in Taiwan. The information given to them by the companies gave them the opportunity to see that the attitude of using computer and immersion experiences had a positive impact on the online learning outcomes. The samples from the companies were however from a study conducted on the regular daily life. What Li, Xu and Chen wanted to find out was whether or not there was a difference in the learning outcome when playing a video game.

What Li, Xu, and Chen did was having their test participants play Second Life, and when they finished the testing the test participants had to answer a questionnaire. The questionnaire is structured to reach different areas, which they wanted to receive information about. The areas in which Li, Xu, and Chen were researching are flow, motivation, and self-efficacy. When the questionnaires were answered Li, Xu, and Chen used Pearson's correlation coefficient on the data to see if there was a relation between the flow, motivation, self-efficacy, and the learning areas covered. The flow part of the questionnaire is used in this project to gather data from the flow levels of the players of Stranded. The questionnaire can be seen in appendix A.

6.3.2 Profiling questionnaire

For the project a profiling questionnaire was created, in order to obtain information about the test participants. The information gained from the questionnaire is about the sex of the test participants, and how old they are. Furthermore, the questionnaire asks

about how often the test participants are playing video games a week, and what types of games they are normally playing. The main reason for using this questionnaire was to make sure test participants met the requirements of the target group. It would also help establish participants' experiences with video games, how long time they play each week, and which genres of video games they like. This questionnaire could also provide basis for further studies related to flow and companions in relation to various video game genres. The questionnaire can be seen in appendix B

6.4 Between-groups Experimental Design

For the experiment, two test groups were set up. One group, the control group, tested the version of Stranded that did not include an accompanying AI. The experimental group tested the one that had. The two versions of the game were designed in such a way that the only variable changed from the experiences was the presence of the companion. The between-groups experimental design was chosen because of the nature of the experiment with only one condition, the effect of a companion on flow within the video game Stranded, therefore e. g. repeated measure would not be useful as the knowledge of how to complete the game could affect the flow of the participants.

7. Game Design

As creating a game has been a big part of this project, it is relevant and necessary to include a section about how to design games. This section explains the methods used for designing Stranded, as well as game design in general.

According to Jesse Schell, game designer, founder of Schell Games, and author of "Game Design: A Book of Lenses", the main goal for a game designer is not to make video games. The video game is merely a means to an end. The end is to create an experience that the player can immerse themselves into (Schell, J., 2014, p. 10). But as Schell points out, experiences are subjective, even when people talk about sharing an experience, they may be similar, and emerging from the same medium. With this in mind, it seems likely that designing games that appeal to a broad audience can be difficult. As experiences are subjective, so might people's' opinions be about what a great game should and should not include. Indeed, according to Hal Barwood, former

LucasArts employee and developer on video game titles such as Indiana Jones And The Fate of Atlantis, design is not always fully conscious (Barwood & Falstein, 2002). Some design choices happen naturally while others require a lot of thought, testing, and readjustments.

Hal Barwood and Noah Falstein are working in collaboration to create a list of 400 rules in the field of game design, called THE 400 PROJECT (Barwood & Falstein, 2002). As of this moment, they have defined 112 of the 400 they believe could exist. All of the 112 now defined rules are too many to go over in this report, so instead this report will focus on the ones that relate to the other theory used to design Stranded, namely GameFlow theory. As previously mentioned, the GameFlow dimensions are concentration, challenge, player skills, control, clear goals, feedback, immersion, and social interaction. Some of the rules described by Barwood and Falstein relate directly to these points. An example of this is rule number one, which is that games should involve some challenge for the player to overcome in order to maintain the player's focus (Barwood & al., FiniteArts, Retrieved April 4th 2016). Rule number three is another example. This rule states that immersion is easily interrupted, and so designers should strive to design the game so that suspension of disbelief is not lost. Rule number six describes that games should provide clear short term goals for the player in order to keep them immersed in the game and feel some sense of progression. Also, although not well defined as a rule in the list, rule 99 states that games should provide a framework for social interaction, as the last of the flow dimensions also describes. In developing Stranded, these rules, as well as the other flow dimensions not accounted for in the list of rules, were given due thought, and most of them were used directly in the design. Given that Stranded is not a multi player game, social interaction was not implemented.

Several of the other rules from the list was also implemented in Stranded. Rule number 22 states that if an AI is present in the game, a certain amount of randomness to its behaviour should be implemented to make it more believable and not seem stupid. The AI in Stranded, called Wanda, has several lines it can say during gameplay, all dependent on which zone of the environment the player and the AI is currently in. These lines are put on a timer, and every 10-20 seconds of game time the AI will say a random line from a selection of available lines. Apart from this, the AI will also say special lines once certain conditions are met, e.g. when the player discovers a new area of the

environment. Rule number 32 states that a developer should put themselves in the player's situation, and think like a player. This is similar to the statement made by Tracy Fullerton in "Game Design Workshop: A Playcentric Approach to Creating Innovative Games". Fullerton states that the first rule for a game designer is to be an advocate for the player (Fullerton, T. 2014). This seems important because the consumers of games are the players. Therefore, it is important to think as a player in order to identify what the players want and create a game that fulfils what the players want from a game. Rule number 59 from THE 400 PROJECT states that games should make the player feel smart. One way Stranded attempts to do this is to make a task that requires players to use their brains to solve a puzzle in the most efficient way. The player eventually finds a stone obelisk that emits three symbols simultaneously; a 3, a 7, and a 10, all presented in roman numbers. These particles give a hint to a code, used for a door later in the same area, which must be passed in order to progress in the game. This allows for the player to accumulate knowledge through the game, and eventually solve the task, using this knowledge. This could eventually lead to the player feel smart about his decisions and efforts. On the other hand, rule number 60 states that designers should provide multiple solutions for challenges. The very same puzzle with the door that requires a code can also be solved by simple trial and error. The user can simply find the right code by inputting various numbers. Still, the most efficient way to solve the puzzle is to utilize the player's knowledge, gained from the stone sculpture and number particles.

7. 1. Typical game elements

Fullerton also describes various typical game elements shared by a vast majority of games, regardless of their genre and play style. Some elements persist, whether the game is a board game, a card game, or a modern video game. Some are important for this project, and some are not. Only the relevant ones will be discussed.

7. 1. 1. Players

The main consumer of video games are the players. This makes video games quite unique, because as Fullerton asks, which other entertainment forms require active participation by the consumer? Examples are music and theatrical plays, but with these

kinds of entertainment, the main consumers are the audience – not the players, as is the case with games (Fullerton, T., 2014, p. 31). Which actions are required by the player differs depending on what game the player is playing, but what is always the same is that to be a player, one must voluntarily accept the rules of the game, put there by the game designers and actively be part of the experience.

7. 1. 2. Objectives

Big parts of video games are often the objectives that they set up for players to complete. This is also referred to by several rules from THE 400 PROJECT, e.g. Rule number six, which requires designers to provide clear short-term objectives (Barwood, H. & Falstein, N., FiniteArts, Retrieved April 4th 2016). This is something very specific to video games, compared to e.g. books and movies. In these types of media, the audience does not have goals during the experience. Obviously the characters in books and movies do have objectives and goals that they work towards, but they are not the audience (Fullerton, T., 2014, p. 31).

In a game of Raptor: Call Of The Shadows, the objective is to, with as many kills as possible, survive until a final boss shows up, and defeat him to earn points to spend on weapons and shield upgrades. In a game of Monopoly the objective is to buy estates in order to earn money, and hopefully, be the player with the most money when the game ends. Fullerton states that “In games, however, the objective is a key element without which the experience loses much of its structure” (Fullerton, T., 2014, p. 32).

Figure 9. Left: Raptor: Call Of The Shadows by Cygnus Studios. (Retrieved from http://www.vogons.org/files/d2.png). Right: Monopoly by Hasbro. (Retrieved from http://xaviesteve.com//wp-content/uploads/2011/12/monopoly-game-with-friends.jpg).

7. 1. 3. Procedures

Procedures covers the “actions or methods of play allowed by the rules”. These can be described as guidelines for which actions that are available for players to do within the game, which allows the player to complete it (Fullerton, T., 2014, p. 32). In Raptor: Call Of The Shadows, the player assumes the role of a fighter pilot, and must maneuver his plane, engage enemies, and collect points by destroying enemy fighters and equipment. These can be seen as instructions that the player can then attempt to follow in order to accomplish some or all of the game’s goals.

7. 1. 4. Rules

Rules in games covers what the player can and cannot do within the game system. In Doom 3 by id Software the player can utilize a variety of different weapons in order to defeat evil demons. The rule related to the usage of these weapons is that any of these weapons will only accept the exact correct type of ammunition. The shotgun uses shotgun shells, the pistol uses clips, and the rocket launcher uses rockets. The rules in video games are typically enforced by programmers, who can directly make it impossible for a weapon to fire if the player does not have the required type of ammunition. In board games the rules are typically enforced by the players themselves. The developers of the game are most likely not present at any game session, and so players have to refer to the game’s rule set if they want to play the game as it was intended. If someone breaks a rule, he is no longer playing the same game as the rest of the players (Fullerton, T., 2014, p. 33).

7. 1. 5. Boundaries

Usually, boundaries in video games refer to the physical boundaries of the game world. In general, the player is only allowed to move around in specific areas defined and designed by the game developers. If a player somehow finds himself in a part of the game world that is technically not part of the game (e.g. places that are not supposed to be accessible), it is most often regarded as a bug or glitch. The physical boundaries of video games are often represented by buildings, fences, or mountains, and as Fullerton describes it, “Players are precluded from moving their characters out of these

boundaries by the underlying code" (Fullerton, T., 2014, p. 35). In Stranded, the physical boundaries of the map are represented by mountains that the player cannot climb or get around.

7. 2. Setting.

Stranded is set on a fictional island in the future where most of mankind have been wiped out by a war. On the island, Wanda, an AI was sent to do reconnaissance work. When the player finds her she has been stuck on the island for over 3 years due to her chasing a butterfly. The player is a plane crash survivor trying to get of the island, with or without the help of Wanda.

7. 2. 1. Characters.

7. 2. 1. 1. The Playable character.

The character the player controls is an airplane crash survivor who needs to find a way to get of the island. The playable character is with no name, gender, age, or voice.

7. 2. 1. 1. Wanda.

As previously mentioned, Wanda is an AI that was sent on a reconnaissance mission. The people behind this orchestration are unknown, and so is their whereabouts. When the player meets her, she is stuck under a small pile of rubble, the player frees her, and their adventure begins. Wanda was designed to be of a curious and childish nature, with her being overly excited by the environment and generally in a very upbeat mood.

8. Level Design

With the initial idea of a deserted island as the game world, the level design process was begun. The idea of the layout of the island was to have different zones, with different puzzles in them for the player, but in the development process of the project, most of the puzzles were removed, but the layout was kept. The first draft of the layout can be seen in figure 10.

Figure 10. Draft of the map used within Stranded.

After the map had been drawn, it was created using Autodesk Mudbox. The actual map looks slightly different. The reason for this was to create a mountain range so that the player would not so obviously discover that the island was square in shape. The map created in mudbox can be seen in figure 11.

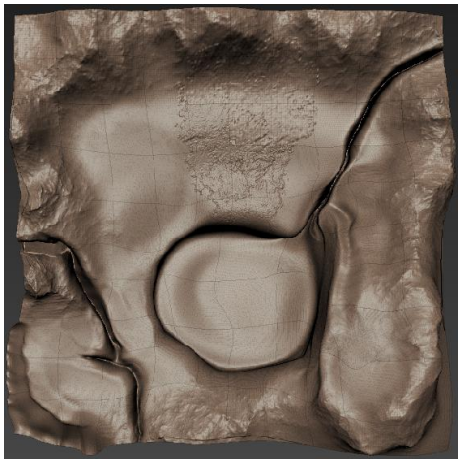

Figure 11. The Stranded game map as seen from above.

After the map was created in mudbox, a heightmap was exported. A heightmap is a black and white picture, with each pixel's grey-scale value being the height of said pixel. The value the pixel can have ranges from 0 to 255, and each pixel has a value in between. The heightmap can be seen in figure 12.

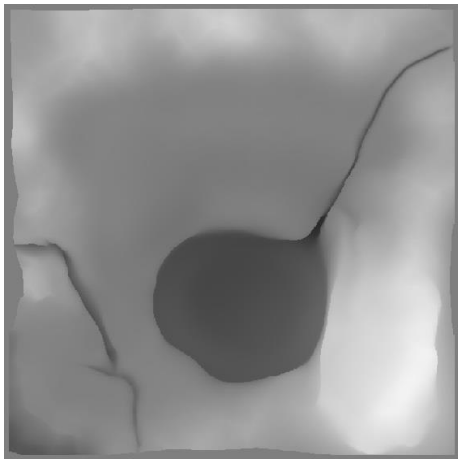

Figure 12. The heightmap exported from Mudbox.

With the heightmap exported, it was imported into Unity3D, so that the data in the picture would create the 3D terrain that was to be used as the game world. In figure 13, the final version of the game map can be seen.

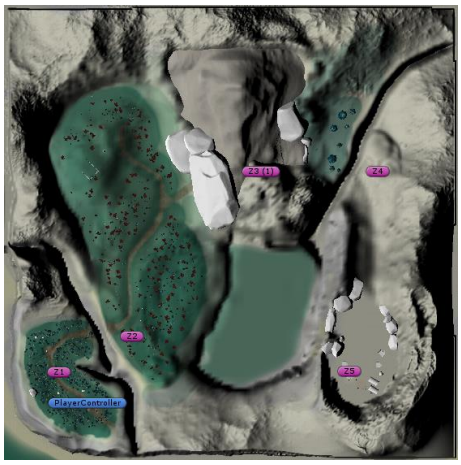

Figure 13. The final map, seen from above.

9. Design and implementation of 3D objects

9. 1. Modelling and sculpting process

Modelling and sculpting encompasses design and implementation of 3D objects. 3D objects used for games and animated movies are, by industry standards, usually made by multiple triangles or squares. The triangles and squares mentioned here are often also referred to as polygons.

According to Autodesk's documentation:

“Polygons are straight-sided shapes (3 or more sides), defined by three-dimensional points (vertices) and the straight lines that connect them (edges). The interior region of the polygon is called the face. Vertices, edges, and faces are the basic components of polygons”. (Autodesk, 2013). Triangular polygons are referred to as tris, and square polygons are referred to as quads. In figure 14 an example of a face of a quad is highlighted. The object in figure 14 consists of several quads. Polygons like these are defined by several components. An example of the vertices of the quad from figure 14 can be seen in figure 15, and the edges can be seen in figure 16.

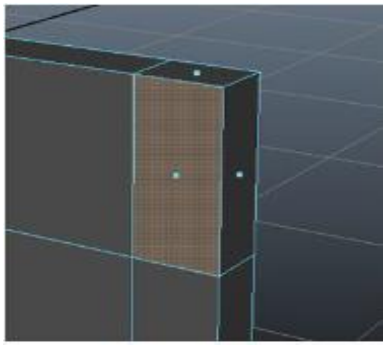

Figure 14. The highlighted area represents the face of a 4-sided polygon.

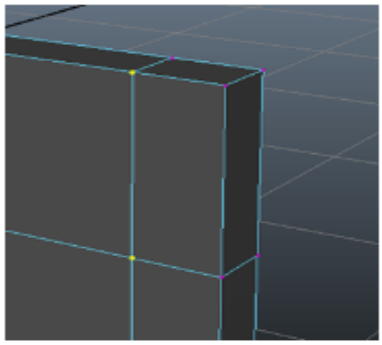

Figure 15. The highlighted points represents two of the vertices related to the polygons of the 3D object.

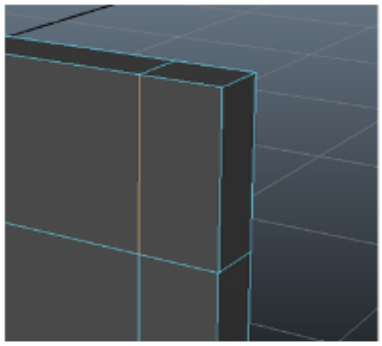

Figure 16. The highlighted line represents one of the edges related to the polygons of the 3D object.

In game development, the poly-count (the amount of polygons) that any given 3D model is made of is very important. With video games, the computer needs to render all the 3D models that are on-screen in real time, while the game is running. The more polygons that need to be rendered, the more computing power is required. On top of that comes rendering the textures (the images applied to 3D models to give them colour), as well as other computations. Therefore, having an environment with a large amount of polygons can potentially largely impact a video game's performance and the user experience.

In light of this, the 3D models that were created for Stranded, were held at as low a poly-count as possible, while still keeping the necessary design outline, and keep them believable and consistent within the art style chosen.

During implementation of the various 3D objects created for this project different approaches were utilized, and several pieces of software were used. The software packages used are:

- Autodesk Maya
- Autodesk Mudbox
- MeshLab by ISTI – CNR
- xNormal by XN3
- Adobe Photoshop

In all cases the base 3D objects were created in Autodesk Maya. Base objects are usually simple 3D objects with a low amount of polygons. In some cases no further editing of the objects were needed, and they were immediately ready for texturing. Texturing was done in Adobe Photoshop. In other cases, further definition of the base objects were created inside Autodesk Mudbox through a process called sculpting. During sculpting, a low polygon object usually undergoes a process called subdividing. Through this process, a 3D model is given more polygons. This is a preparation state before actual sculpting to takes place. The reason for this process is to give the object enough polygons to mould the object during the sculpting process. If this process is not completed, sculpting the object is going to be very difficult, if not impossible.

9. 1. 1. Example: Stone door.

This example shows the modelling process of the implementation of the stone door, used for one of the puzzles in Stranded. Initially, a basic model of the stone door, as well as the stone framing was created inside Autodesk Maya. This stage is shown in figure 17.

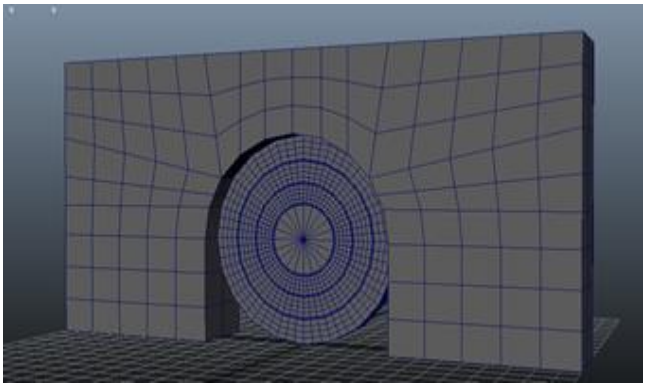

Figure 17. Base model for the stone door and framing.

The base models were then exported from Autodesk Maya, and imported into Autodesk Mudbox. Inside Mudbox, several subdivisions were created, boosting the poly-count from 8560 tris to roughly 4.8 million tris. This made it possible to create a very high-poly detailed version of the stone door and frame, with a lot of deform features that helps define the material used, in this case stone. That high-poly version of the stone door and frame is shown in figure 18. A close up of some of the details is shown in figure 19.

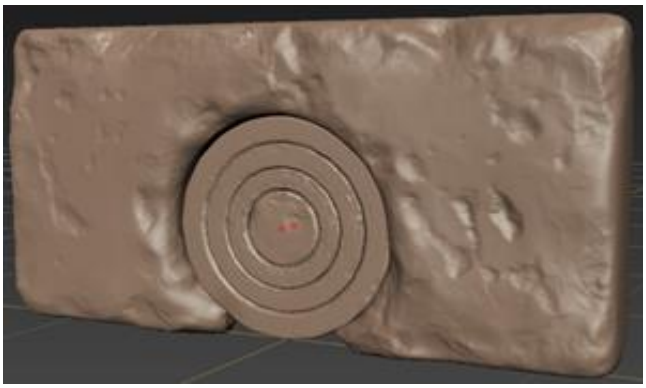

Figure 18. The figure shows the high-poly version of the stone door and frame with 4.8 million tris.

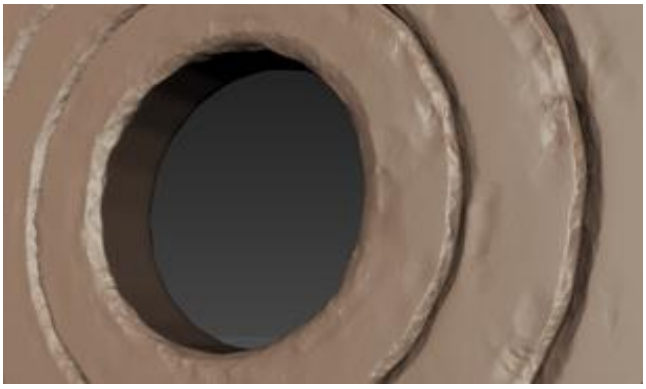

Figure 19. Close-up of some of the details created by the sculpting process.

The sculpting process is done by using brushes inside Mudbox. Sculpting brushes have different uses. The main selection of sculpt brushes are shown in figure 20.

Figure 20. Brushes used for sculpting in Autodesk Mudbox.

The brushes mainly used for the stone door and framing was the scrape brush and grab brush. Additionally, the smooth brush was used to smooth out very sharp edges on the objects. After finalizing the stone door and frame from figure 18 the models were individually imported into the software package MeshLab. This software allows for simplification of 3D objects, reducing the poly-count dependent on the inputs the user provides. Figure 21 shows the frame of the stone door imported into MeshLab, and the navigation to the simplification filter called Quadratic Edge Collapse Decimation used for reducing the poly count. The description of the filter reads “Simplify a mesh using a Quadratic based Edge Collapse Strategy; better than clustering but slower”.

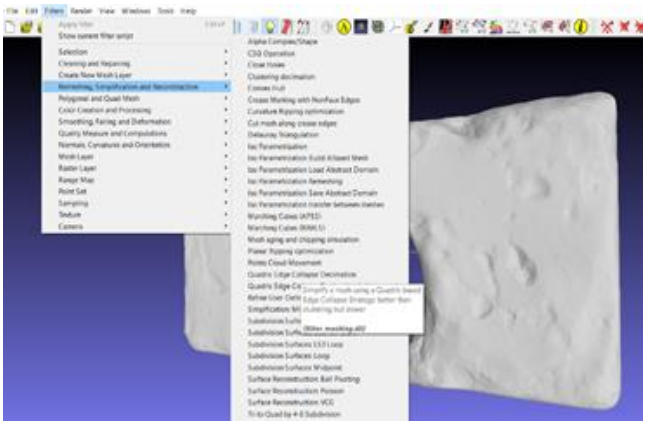

Figure 21. The figure shows the stone door frame imported into MeshLab.

MeshLab presents a window with multiple options for simplifying the 3D object. Only two of them were altered from the default settings; the target number of faces were set to 3000, and the quality threshold was set to 1 for maximum quality output. The full settings are shown in figure 22.

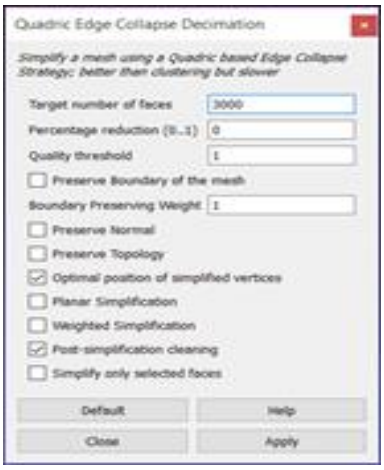

Figure 22. The settings for Quadratic Edge Collapse Strategy used on the stone door frame.

Once the simplification process was done by the program, the output was a 3D object with the same basic shape, but with a heavily reduced polygon count. In the case of the stone door frame, the low polygon count version was reduced to 3000 polygons.

Following the simplification process, the 3D model of the door frame was imported into Autodesk Maya for a process called UV-mapping. During the UV-mapping process a 3D model is essentially stretched out on a 2D plane, which makes it possible to work with in a graphical package such as Adobe Photoshop when making various maps for 3D models. A map in 3D-graphics terms refers to a 2D layout of a 3D model. The UV map of the low polygon version of the stone door frame is shown in figure 23.

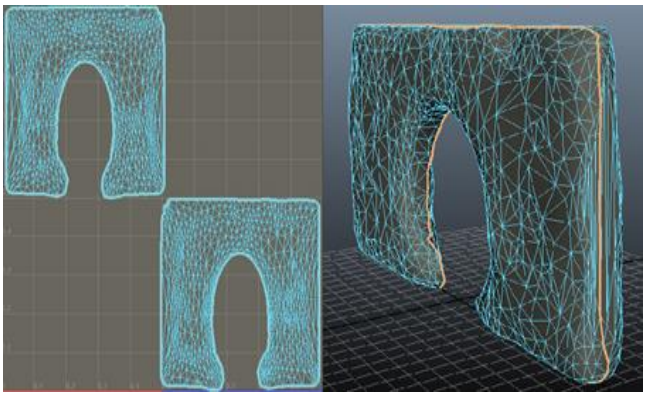

Figure 23. UV-mapped version of the low poly-count stone door frame.

Following the process of UV-mapping the stone door frame, the program xNormal was used to render a normal map from the high-poly version to the low-poly version of the stone door frame. Both the high-poly version and the low-poly version were imported into xNormal and a “ray distance calculator” was used to determine the best numerical inputs for technical calculations for rendering of the normal map. After that, the settings for the normal map render was set, which can be seen in figure 24.

Figure 24. Settings for the normal map render in xNormal.

When the settings are put in, the rest is done by xNormal. It renders out a normal map for the low-poly version of the stone door frame, based on the high-poly version. This process is shown in figure 25.

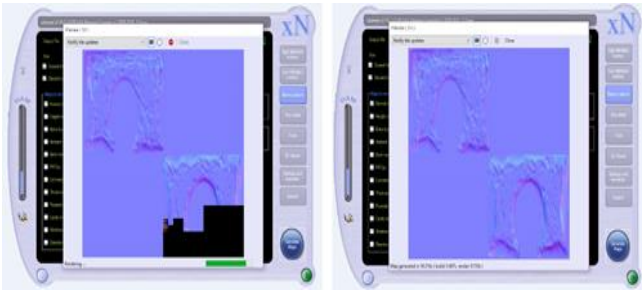

Figure 25. Left: Rendering of normal map in xNormal. Right: Finished rendered normal map.

What the normal map does, is that it essentially makes a low-poly version of a model look like a high-poly version by manipulating how light reacts to the surface of the model, which is dependent on the RGB[1] values of the normal map. The direction that any polygon face is facing is referred to as its direction normal, or simply it’s normal. This normal is a three-dimensional vector with an x, y, and z coordinate. These

[1] Red, Green, and Blue

coordinates can be stored as RGB values, and that is what normal maps are used for. The game engine can then look at the RGB values of a normal map, and determine how light should react to the shader applied to the model in question. This can add a lot of detail to a very simple 3D model. The low-poly version of the full stone door with the normal map rendered from xNormal and with a simple lambert shader with the color (R=129, G=127, B=116) is shown in figure 26.

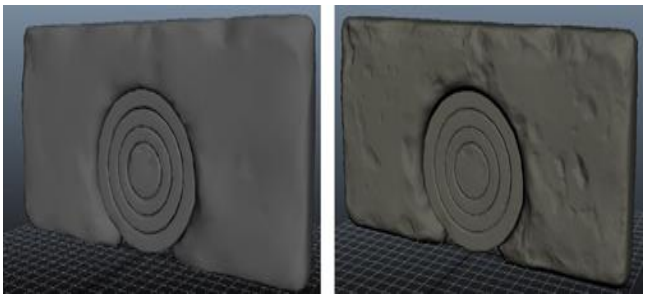

Figure 26. Left: Low-poly version of full door without normal map. Right: Low-poly version with lambert shader and normal map applied.

All models for the stone door, including all modules such as rings and the frame went through the exact same processes, from basic shaping, over sculpting and simplification, to UV-mapping and rendering of normal maps. The final product was a full low-poly version of the stone door and frame, with a heavily reduced polygon count, but with a fine level of detail. Eventually, the full door was reduced from 4.8 million tris to 7000 tris, and final graphical output shown in figure 27.

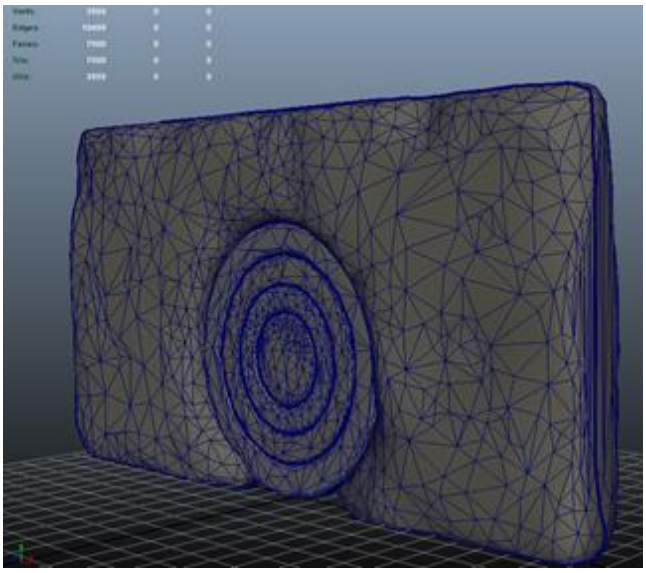

Figure 27. Final 3D model of the stone door and frame, with the wireframe of the model visible.

9. 1. 2. Shading and texturing 3D assets

Shading and texturing refers to processes of applying colour, texture, and other attributes related to looks of a 3D asset. As Derakshani puts it, “Shading is the term for applying colours and textures to create materials, also known in the Autodesk Maya software as shaders. A shader defines an object’s look—its colour, tactile texture, transparency, luminescence, glow, and so forth” (Derakshani, D., 2012., p. 271). In Autodesk Maya, when an object is first created Maya assigns a simple lambert shader to it. A lambert shader deflects and scatters light evenly across the object’s surface (Derakshani, D., 2012., p. 273). Other types of shaders are e.g. the blinn shader and the phong shader. These shaders are often used for metal surfaces, as they are great for creating highlights that look very realistic. These shaders will not be discussed in depth, as they are not used in the Unity game engine. Unity has its own built-in shaders.

In many cases, objects in Stranded have a simple shader assigned, as well as a single colour. These are created by applying a new material to the object and choose a colour for it. Other objects, however, require a little more work. In order to create objects with more colours than one, a UV-map must be created. Figure 23 shows an example of the UV-map for the stone door in Stranded. Once the UV-map is complete, a texture can be applied to the now UV-mapped object’s shader. Textures are image files with colour information or shades of grey, depending on which kind of texture it is.

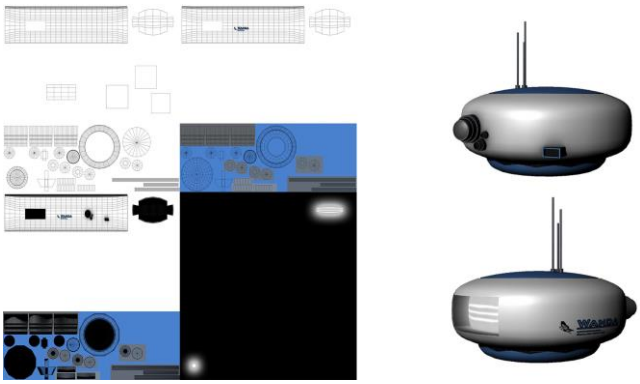

Figure 28. The UV-map, colour map, ambient occlusion map, and emission map for Wanda in Stranded.

The top left of figure 28 shows the UV-map for the AI character Wanda from Stranded. The top right shows the colour map for the same object. The colour map is what gives the object flat colour. The bottom left is called an ambient occlusion map, and provides shadow information in the cracks and crevices of the model. As Derakshani puts it,

"Ambient Occlusion goes by the premise that when two objects or surfaces are close to each other, they reduce the amount of light at the intersection" (Derakshani, D., 2013, pp. 543-544). This principle can be seen in figure 29. The ambient occlusion map was generated by Autodesk Maya.

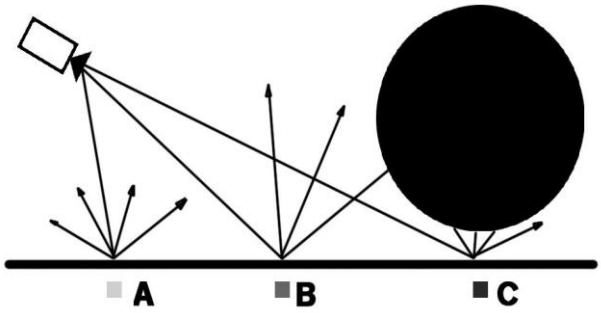

Figure 29. Illustration of the principle of Ambient Occlusion.

The camera in figure 29 casts an initial ray into the world, which is reflected by the floor. If the reflected ray immediately hits another object, the shadow intensity at the original impact point on the floor is very high, which is represented by point C. Point B represents the case where the reflected ray from the floor eventually hits another object in some cases, but some sampled rays hits no other objects. The shadow intensity at this point is somewhere in the middle between completely occluded and completely visible. Point A represents the case where the shadow rays reflected by the floor hits no objects during its travel time, or eventually hit objects that are too far away to be regarded as relevant for occlusion. The result of baking an ambient occlusion map is a grayscale image which can be put on top of the colour map to give it shadow information, adding more detail. The bottom right in figure 28 shows what is called an emission map. An emission map is a texture map which adds lighting information to a model, based on texture coordinates. In Unity's documentation it says: "Controls colour and intensity of light emitted from the surface. When an emissive material is used in your scene, it appears to be a visible source of light itself. The object will appear "self-illuminated" (Unity, 2016b). Black and dark grey areas of an emission map represents areas on the model that emits little to no light. Light colours represents areas that are lit up by lights on the model, and very bright light grey and white colours typically represents sources of light. In the case of the emission map in figure 28, the white parts are the bottom and behind of Wanda, which have light sources. Instead of adding an actual light to the

model in these places in the game engine, we can fake it using the emission map, and thereby save computing power.

9.2. Design choices

When designing and implementing assets for video games, it is important to remain consistent in the aesthetics and style of the objects and their colour and textures. This means that the world should look consistent to the player all the way throughout the game. If a decision is made to make a game in a realistic style, it is important not to develop assets that look cartoonish or stylized. They should represent the real world, and should therefore look exactly like real objects. It is important to try and avoid conceptual non-sequiturs (Adams, E., 2010). This also means that a game set to be hyper realistic should avoid the usage of e.g. medieval weaponry. Only objects that make sense in the time period should be accepted, and only if they fit the other aesthetics of the game in general.

The world of Stranded is what is in graphical terms called “stylized”. This means that the world and the objects that exists in the world look realistic to a certain point, but is either simplified to an extreme, or has textures that are not consistent with what objects look like in the real world that we live in today. Figure 30 shows a couple of examples of objects from stylized games, and one from Stranded.

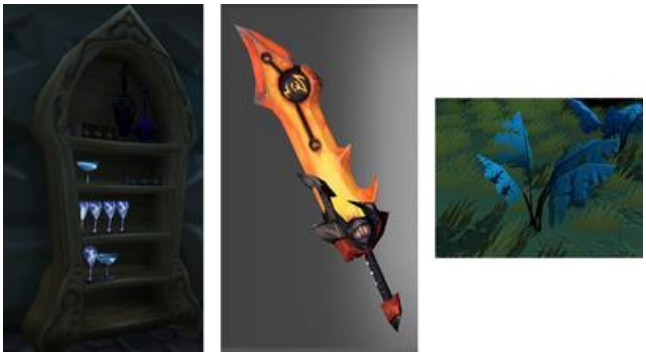

Figure 30. Left: Furniture with assets from World of Warcraft. Middle: Sword from DOTA (Retrieved from: https://hydra-media.cursecdn.com/dota2.gamepedia.com/thumb/4/45/Cosmetic_icon_Sword_of_the_Eleven_Curses.png/256px-Cosmetic_icon_Sword_of_the_Eleven_Curses.png?version=46dd70aee40062a98736f11dc7009b38). Right: Plant from Stranded.

The assets in figure 30 look like the things they are supposed to represent. They look like a piece of furniture, a sword, and a plant. However, they look nothing like how

those kinds of objects would look in the real world. They are heavily stylized. It would appear that utilizing a stylized graphical style allows artists to bypass the rules of the real world, and e.g. create objects that would not exist in the real world. It could be a sword with a blade made of fire, or a ball of energy that could be hurled at enemies. Some aspects of the real world must probably be preserved though, in order for people to still find these objects relatable and, in some sense, realistic. As previously stated, it is important that if one does choose this style for a game, it is best to stick to it for the entire game, in order to not create abruptions of immersion and/or suspension of disbelief.

10. Sound and Music

10. 1. SFX.

The sound effects used in Stranded was chosen to fit with the natural sounds of what the objects would create in real life, e.g. a tree falling would sound realistically as a tree falling, instead of a cartoon inspired *clunk* sound. The sounds effect used in Stranded include footsteps, trees falling, paper being picked up, a flashlight button being pressed and the static sound of a radio transmitter.

10. 2. Music.

The music used within Stranded was the soundtrack of the game Firewatch, developed by Campo Santa and published by Panic. The music in Firewatch was chosen as its quiet nature fitted well with the explorative nature of Stranded.

10. 3. Voice-Recording

Wanda, the companion the player have in the game, was created with the intent of a curious and childlike persona. As Wanda is an AI, with a free roaming robot body, her voice was chosen to be non-human in the sense that her voice does not sound like the average human voice. For the recording itself, the open-source digital audio editor Audacity was used, together with a Presonus external audio card and Presonus microphone. The setup can be seen in figure 31.

Figure 31. The audio recording setup with computer, external audio card and microphone.

The recording took place in a small room that contained both furniture, bookcases, shelves and a table. The reasoning behind this was to have as little as possible of natural reverb interfering with the recording session. After the recording session was done, the best takes were found, the ones that best suited the expression and quality of Wanda, and these were then normalized and exported as WAV files. The files were then imported into another digital audio editor, the pitch modification software Melodyne, created by the software company Celemony. With the files in Melodyne, the pitch modules between each of the polyphonic notes were removed and pitch was increased by 200%. The workspace of Melodyne can be seen in figure 32.

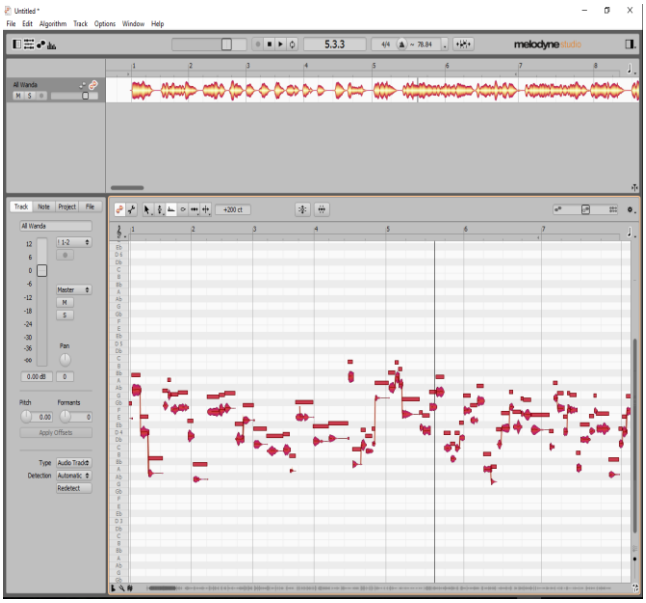

Figure 32. Melodyne workspace.

The results of this manipulation is a voice that sounds robotic and a little child-like which fits the character created for stranded. The sounds effect were taken into Audacity once again to make the last changes, and to insert additional effect such as a drumroll and a computer generated voice.

11. Implementation of Stranded

11.1 Unity3D

Unity3D is a game engine created by Unity Technologies. Unity is a multi-platform engine that supports Windows, Xbox 360, Xbox One, Mac, Linux, Playstation 3, Playstation 4, Wii, and Wii U, as of 2016. They also support the newly released virtual reality technology such as Rift and Vive (Unity, 2016d), with over 4 million registered developers. Unity also have services to help with the creation of applications that include Cloud building, which allows users to build an application on a server which should help decrease build time. There are also analytical tools that helps the developer retrieve user data about their applications (Unity, 2016e), such as play time and when a new user uses the application. There is also the option for custom events to be tracked. An example of this could be how many points the user have earned. The way Unity works is that it has a 3D world which is called the game world; this is where the game can be created. For the world to be displayed on the screen it is needed to set up a

camera in the game world. The camera will look at the world and depending on what it sees, it renders images to what is called the screen space. The screen space is the same as the game world, it is just a 2D reflection of it instead, allowing it to be displayed onto the screen.

11.2 Programming in Unity

Unity comes with Microsoft's Visual Studio as an integrated development environment (IDE). It is primarily designed for C# and .NET languages, and supports Windows, Linux, and Mac OS X. It comes with integrated debugging to help the developer find errors or bugs that might have occurred during run time. There are also plugins to help increase the productivity of the developer, one of these tools is Unitytool VS and this tool allows for further debugging (Microsoft, 2016). Furthermore, it also imports Unity documentation into the IDE so the developer does not have to switch over to an internet browser to look up the online documentation. Instead they have it build into their visual studio.

Unity supports three programming languages; C#, UnityScript, and Boo Script. C# is a C language created by Microsoft. UnityScripts is similar to Javascript, but it is specifically made for Unity. Boo Script is a variant of the Python language.

11.3 Animating in Unity

Unity has an animation tool built in called Mecanim, which allows the user to make animations based on their component. To animate for example movement of an object, one would use the transform component which handles the transformation, rotation, and scaling of said object (Unity, 2016a). In Stranded animations were used both for functionality, such as animating doors opening, but also to create a rotation loop for a windmill in order to give the world more life. The animation tool can be seen in figure 33.

The animation clips are structured in a flowchart called animation controller that works similar to a state machine, which keeps track of which animation is playing, which should play, or if two animations should blend together. Blending of animations is when two or more animations gets put together into a single motion by interpolation, and the

weight, or the values, of the animation get put into the calculation. Meaning that you take some values from one animation and mixes it together with the values of another. This is used in cases where you want to have a smooth transition between two animations. An example of this could be between walk and run, where the blended version would result in a jog animation. For creating a jog animation one could weight both walk and run equally, the program would then take the two animation, enturbulate the difference between the two and create a new motion.

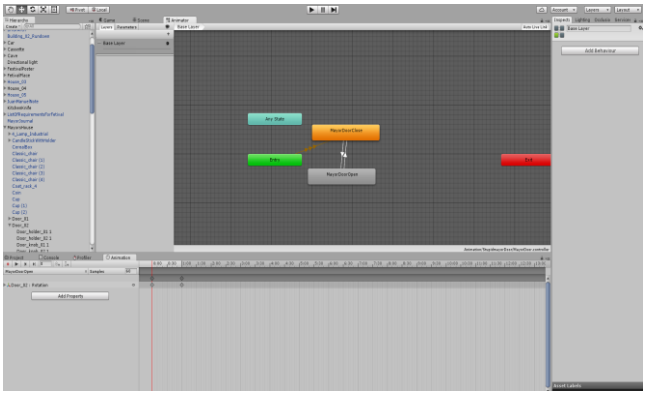

Figure 33. The animation tool in unity.

11.4 Iteration design

The implementations of the game was done through an iterative design method, where a build of the game was developed, tested for bugs, and these bugs were fixed to improve the user experience. This process continued until the game was at a satisfying state.

11.5 Stranded

The goal of Stranded was to create an interactive experience which allows the user to explore the world of Stranded at their own pace. As previously mentioned, one version of the game was with a companion that follows the player around on their journey, and comes with remarks, banter, and clues about the world. Another version of the game was also created, but the difference here is that the companion was absent. However, the players should receive the same amount of information in both of the versions. While developing Stranded two unity-projects were used, and this was done to help decrease the development time of the game. One was for programming the functionality of the game, such as movement of the user, interaction with objects, and creation of the AI.

The other project was for implementing assets, level design, and environment. This was used in order to assure that more people were able to work on the project at the same time, allowing the game to be completed faster.

Stranded is played from a first person view meaning you see the world from the protagonist's point of view. The W, A, S, and D buttons are used to control the direction the character is moving, and the mouse is used for the direction the character is facing. Handling the input of the player is done by using Unity's standard input classes, which changes state if the button has been assigned to a specific value. To make it easier to get an overview of the interaction state for the player, all the inputs are handled by a single script.

```
        InteractionState();
        //Get the input from the keyboard and mouse
        Yaxis = Input.GetAxis("Mouse X");
        Xaxis = Input.GetAxis("Mouse Y");
        horizontal = Input.GetAxis("Horizontal");
        vertical = Input.GetAxis("Vertical");
        Jump = Input.GetButtonDown("Jump");
        Sprint = Input.GetButton("Sprint");
        Interact = Input.GetButtonDown("Interact");
        ComeAI = Input.GetButtonDown("ComeAI");
        GoAI = Input.GetButtonDown("GoAI");


    }
```

//Handle the interaction state which state what can and can't be done in the current state.

```
    private void InteractionState()
    {
        case state.Play:
            FpsScript.Inputs(horizontal, vertical, Xaxis, Yaxis, Jump, Sprint);
            AInteraction.AIControl(GoAI, ComeAI);
        case state.Dialogue:
```

```
 FpsScript.Inputs(0, 0, 0, 0, false, false);
//give inputs to the dialog system
LimitTimer += Time.deltaTime;
if(Interact)
{
   DiaUI.ContinueDialogue();
}
if (LimitTimer > 0.3f && vertical > 0 || LimitTimer > 0.3f && vertical < 0)
{
   DiaUI.SelecteChoice(vertical);
   LimitTimer = 0f;
}
```

The code is about how the input was handled to ensure that the player could interact with the game. It also ensured that the game called the correct method when certain conditions were met, and not only when a button is pressed. This was done to ensure that bugs did not occur due to a method being called when it should not. The way this was done was by having an interaction state, which dictates the current state the player is currently in, and what they can and cannot do in this state. For this application there are three states the user can be in; play, dialogue, and menu. The play state is when the user controls the character and the movement around the world. The dialogue state is when the user is interacting with the AI, and the menu state is when the game is paused and no longer needs to interact with the game world. This was done by using a state machine which ensure that the player can only be in one state at the time.

The user has the option to interact with certain objects in the game environment. This is done by being relatively close to the object while looking at it and pressing the interact button, which in Stranded is “E”. Interaction is created by having a ray cast directly forward from the player camera, and if this ray hits an object, it will then assign that object to the current target object, which allows the user to interact with the current target object, when pressing the interaction key. Furthermore, it also looks at the object’s tag[2] and changes the crosshair icon. The crosshair is, per default, a single dot in the middle of the screen. If the object that the player is currently looking at is

[2] Tag: typically a single word, which can be used to identify the type or use of certain objects in Unity

intractable, the dot changes to a square-like icon. This is done to communicate that the object is intractable.

11.6 Tools created for project

During the creation of the project there was a lot of models that was placed throughout the scene of the game. With that many models it would take a significant amount of time if models would have to be changed, either because of changes in visual design or because of optimization with a model that have fewer polygons. Therefore, an extension to unity was created that could replace models placed in the scene with another one. This could either be done by manually selecting the models, or typing in a name and all models sharing this name would be replaced.

```
Public class ReplaceGameObjects : ScriptableWizard
{
   public GameObject Prefab;
   public GameObject[] ObjectsToReplace;
   public bool KeepOriginalNames = false;
   public string ObjectName = "";
   public List<GameObject> ObjectToReplaceList = new List<GameObject>();
   [MenuItem("Custom/Replace GameObjects")]
   static void CreateWizard() {
var replaceGameObjects = DisplayWizard<ReplaceGameObjects>("Replace GameObjects", "Replace");
      replaceGameObjects.ObjectsToReplace = Selection.gameObjects;
   }
   void OnWizardCreate() {
      if (!string.IsNullOrEmpty(ObjectName)) {
    foreach (GameObject objs in FindObjectsOfType(typeof(GameObject)) as GameObject[]) {
            if (objs.name == ObjectName){
```

```
                ObjectToReplaceList.Add(objs);
            }
        }
        foreach(GameObject go in ObjectToReplaceList)
        {
            GameObject newObject;
            newObject = (GameObject)PrefabUtility.InstantiatePrefab(Prefab);
            newObject.transform.SetParent(go.transform.parent, true);
            newObject.transform.localPosition = go.transform.localPosition;
            newObject.transform.localRotation = go.transform.localRotation;
            newObject.transform.localScale = go.transform.localScale;
            if (KeepOriginalNames)
                newObject.transform.name = go.transform.name;
            DestroyImmediate(go);
        }
    }else {
        foreach (GameObject go in ObjectsToReplace)
        {
            GameObject newObject;
            newObject = (GameObject)PrefabUtility.InstantiatePrefab(Prefab);
            newObject.transform.SetParent(go.transform.parent, true);
            newObject.transform.localPosition = go.transform.localPosition;
            newObject.transform.localRotation = go.transform.localRotation;
            newObject.transform.localScale = go.transform.localScale;
            if (KeepOriginalNames)
                newObject.transform.name = go.transform.name;
            DestroyImmediate(go);
```

11.7 UI

Stranded has very few user interface elements. First off there is the crosshair, which changes based on what the player character has in front of them. Then there is also the dialogue interface, which consists of questions that the player can ask the AI, and what is currently being said by the AI. The design of the dialogue UI can be seen in figure 34.

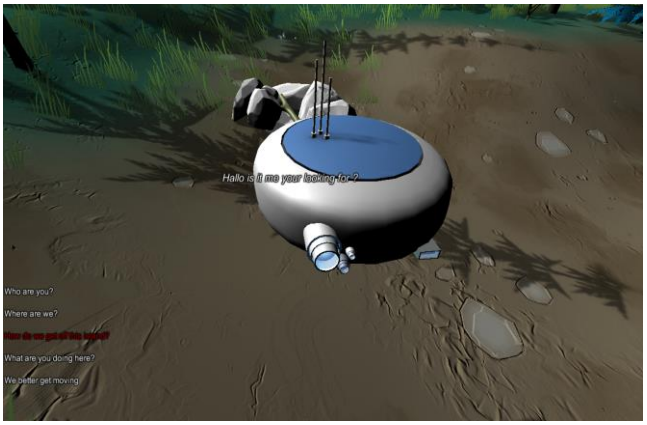

Figure 34. UI design for dialogue.

The design of the UI is kept very simple to ensure that the players still felt like they were in the game world, and not having this feeling obstructed by the UI

11.8 AI

In the game version with a companion, which follows the players on their journey throughout the game. The AI navigate around the world using a Navigation Mesh (navmesh), which is data that describes the walkable areas of the game world (Unity, 2016c). By using this information it will calculate a path to the desired location. The navmesh data is baked, meaning it cannot be changed at runtime. This navmesh is what the AI uses to calculate a route to transverse the terrain of the game world in a intelligent way. The way the navmesh works is that it observes the terrain of the game and sees where the AI can stand, the locations are then connected together into convex polygons, which consists of multiple points that is converted into a single mesh, the navmesh. The navmesh is then used to find a path between two points, starting with start points which ends up visiting all the neighbouring points until it reach the designated end point. By tracking all points visited to reach the designated goal will allow the program to find a sequence of point from the start and the end, and this is what called a path. This method of pathfinding is called A*.

11.9 AI dialogue

The user needed to be able to talk with the AI, so a plugin, called Dialoguer, was used to help in the creation of the dialogue for the AI. The way the plugin works is it initiates a state machine which defines what dialogue is currently active, and what should be active next, based on different conditions and how the user interacts with the system. In figure 35 a flowchart of one of the dialogue tree the AI has can be seen.

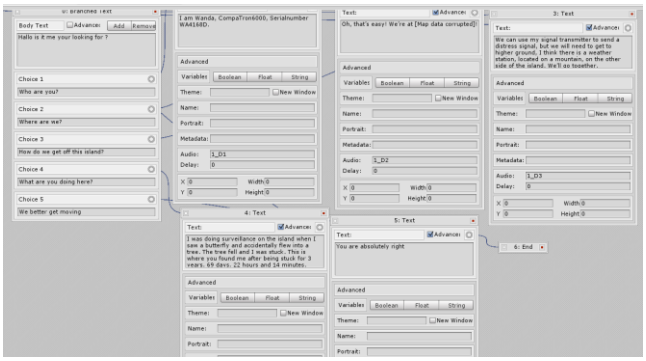

Figure 35. Dialoguer flowchart for creating dialogue.

The flowchart helps visualize the state machine being used, where each node has the function to pass additional information. This information could be the name of an audio clip, which is used to tell the program what audio clip should be played to the corresponding node. In figure 35 the UI of the dialogue system is shown, and how the different nodes are connected. When dialogue first gets called, an event is taking place, and this event is called onDialogueTextPhaseHandler. This eventhandler first fills out the dialogue text with what is being said, and afterwards controls if it is a branching text, meaning the player have the option to select an answer. If this is the case then it will fill out the dialogue options for the player to choose from.

11.10 AI behavior

For the project the AI needed to change what it could say based on what area of the world it was currently in, and the way this was done was by implementing different zones. These zones changes what the AI is going to say, and the dialogue the player will get when interacting with the AI. In figure 36 the different in-game zones can be seen.

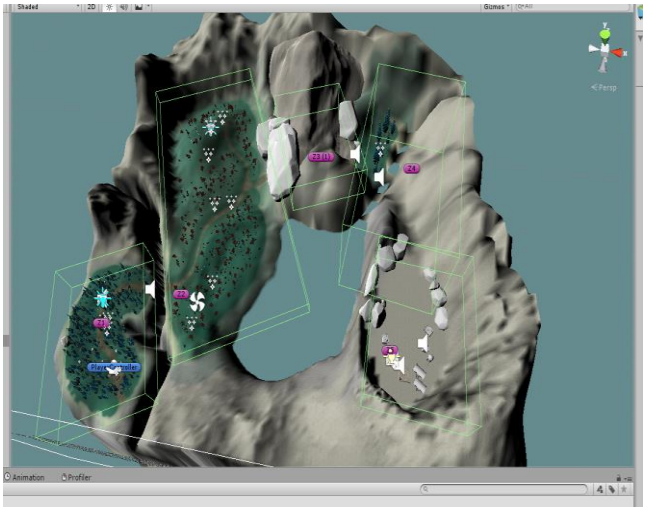

Figure 36. Topdown view of the map showing the different zones.

The zones affect the AI when the AI is entering a zone it will initiate AI banter , and if the AI is staying in the current zone for a longer period it will initiate random banter based on the zone. The way this was done was by using eventhandler. When the AI enters a new zone events are called, and these events are giving information about the AI and that it has changed its current zone.

```
//Delegate for ZoneChange
public delegate void ZoneChangeEventHandler(object source, NameEventArgs args);
//event
public event ZoneChangeEventHandler ZoneChanged;

protected virtual void OnZoneChanged()
{

    if (ZoneChanged != null)
        ZoneChanged(this, new NameEventArgs() { Name = zone.ToString() });
}
```

What can be seen is the event handler for the change zone event. First, the event is defined and what parameter this event has, in this case a custom event called NameEventArgs is used. Once the AI has figured out what zone it has entered the method OnZoneChanged is then called, which first checks if there are any listeners for this event. If it does, it will instantiate a new NameEventArgs class, and give it the name of the zone. The listener then gets the new data from the event.

```
ai.ZoneChanged += OnZoneChanged;
```

```
public void OnZoneChanged(object source, NameEventArgs e)
   {
      //Stop the Coroutine, change the zonename and start the coroutine again from anew
      ZoneName = e.Name;
      Debug.Log("Change zone to " + e.Name);

      if(CheckIfVistedZone(e.Name) == false)
      {
         //Now that the zone have not been visted before play audioclip that correspond to the
zone you have entered
         FindAndPlayEnterZoneTalk();
      }
      if (IEZoneTalk != null)
      {
         StopCoroutine(IEZoneTalk);
         IEZoneTalk = null;
         IEZoneTalk = ChangedZone();
      }
      StartCoroutine(IEZoneTalk);
```

The new data gathered then has it own OnZoneChanged event, which defines what is going to handle the information that the event gives. In this case it is a simple string that is a sequence of characters. This event looks at the string, and sees if the AI has been in the zone before. If it has not been in this zone it will play the method for entering a new zone, however, if the zone has been visited before it will restart the coroutine[3] for the random banter the AI can come up with in the given zone.

12. Results

The quantitative data gathered from the Flow questionnaires were analysed through different methods, these methods will be presented and explained here

For the experiment the following null-hypothesis was established.

"A companion with artificial intelligence will have a negative effect on flow in Stranded"

[3] Coroutine is a independently executing computation

During the analysis, the two data sets are referred to as No_Wanda and Wanda, the former being the results of the experiment without an artificially intelligent companion and the latter with an artificially intelligent companion. The experiment gave a total of 42 samples, 21 for No_Wanda and 21 for Wanda. The results can be seen on appendix C. These results were analysed statistically. First the data was analysed using the Shapiro-Wilk test of normality, to which a value of $p>0.05$ is this is a chosen critical value (Field, A. & Hole, G., 2003, p. 160). The reason behind using Shapiro-Wilk and not Kolmogorov-Smirnov, is because of the small amount of samples (Razali, N. M. & Wee, Y. B., 2011, p. 5). The results of the Shapiro-Wilk can be seen in table 3.

Table 3.The normal distribution for the experiment resulted in ($p<0.05$) for No_Wanda and ($p>0.05$) for Wanda. Thus the data from No_Wanda is not normally distributed, but Wanda is.

Tests of Normality

	Shapiro-Wilk		
	Statistic	df	Sig.
No_Wanda	.888	21	.020
Wanda	.917	21	.077

In figure 37 a histograms, visual representation of normal distribution can be seen, together with the distribution curve.

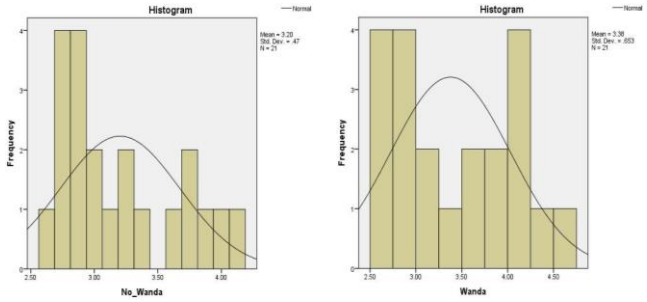

Figure 37. The histogram for both the No_Wanda and the Wanda. As mentioned, the curve visually describes a normal distribution.

As seen in figure 37, No_Wanda was not normally distributed and Wanda was, therefore the data would have to be treated with non-parametric methods. The data told that a Mann-Whitney U was used (Field, A. & Hole, G., 2003, p. 235-239). In table 4 the output of the test can be seen.

Table 4. Output from SPSS. As it can be seen, the output is not statistically significant (p>0.05).

Test Statistics

	Flow
Mann-Whitney U	187.500
Z	-.832
Asymp. Sig. (2-tailed)	.405
Exact Sig. (2-tailed)	.413
Exact Sig. (1-tailed)	.206

The exact data on the table that is of interest is the exact sig. (1-tailed) as the null-hypothesis states that a companion will have a negative effect, which means only one direction is of interest. The output of exact sig. (1-tailed) is (p= .223), which is not significant, which means that the null-hypothesis cannot be rejected based on the results from this experiment.

13. Discussion

13. 1. Game design and Flow

As flow theory states, for a person to be so absorbed into an experience that he can rightfully be said to be in flow, there must be an equilibrium between the difficulty level of the activity, and the skill level that the person possess. Therefore, if a game designer aims to make a game that makes this experience possible, he should design the game so that this balance is in place. This can be rather difficult, as players possesses very different individual skill levels, and these skill levels may be unevenly distributed over several game genres. A player who is very good at first-person shooter games, could be worse off if challenged by a real-time strategy game. Game designers typically combat this by implementing different difficulty levels that the player can choose from. Another option is to implement a dynamic difficulty system, which automatically adjusts the game's difficulty level based on the player's performance. In the case of Stranded, no systems were implemented to balance the game for all players. Instead, the challenges of the game were tested several times on players with different skill levels and tweaked if major difficulties was reported by any of the player bases. A dynamic difficulty system would require more time than was scheduled to implement, and given that the game in general focuses more on the gameplay related to the companion and not the puzzles, it was discussed whether the difficulty of the puzzles even needed to be very

high. It was agreed that given the focus of the study, the puzzles needed not to be difficult, and so it was attempted to make them as easy as possible, while still providing simple tasks for the players to complete. As a result of this decision it is possible that only few people found the game very challenging, and thus players of high skill level in this particular genre could have leaned to the "bored" category of the flow graph. It is possible, however, that these players were still capable of reaching the flow state despite slight feelings of boredom, e.g. through the autotelic personality. It is also debatable whether feelings of flow, boredom, and anxiety is even possible to experience if the test session is very short. DeMarco and Lister states that e.g. when working with single minded tasks, in their context programming, flow cannot be turned on and off, but require at least 15 minutes of concentration to manifest. They further state that during this period of initial concentration, even the smallest interruption can disrupt the process and make it either difficult or impossible to descent into the flow state. (DeMarco, T. & Lister, T., 2013, p. 62). To further argue this case, more studies would need to be performed.

13. 2. Work process

After the initial brainstorm period of this project, an iterative work method was utilized to always be able to further define and redefine the aspects of Stranded. Several puzzles were created that was to be implemented, along with several resources found on the game map. Many of these were scrapped in order to focus more on the core study of the project. The iterative method worked very well for developing the game, as it made it possible to constantly see new assets being brought into the game, as well as assess the usability and quality of the game. This work method also allowed for work on multiple aspects of the game at the same time, compared to e.g. the waterfall methodology. In-group showcase sessions were held every second week, which helped keeping track of the development state of Stranded as well as making sure that every team member was actively working towards the end product. When a working prototype was ready, small test sessions were held to test for major bugs as well as whether the game was sufficiently challenging. A number of changes were then made, based on the feedback from these tests.

Development of Stranded was mostly based on the theory found in Fullerton and Shells books on game design, GameFlow theory by Sweetser and Wyeth, as well as accumulated experience creating several games during semesters at the university. It would have been interesting, given a narrower target group, to try and use the playcentric approach discussed in "Game Design Workshop: A Playcentric Approach to Creating Innovative Games" by Fullerton. This method incorporates test sessions in the extremely early phase of game creation, all throughout the development, in order to capture what it is that the key audience really wants from the game, and then implementing it, keeping the player experience at the central focus. The playcentric approach is also an iterative method (Fullerton, T., 2014, p. 3), and so would allow for many of the same development traits as the method used for this project.

13. 3. Analysis

As the output from the Mann-Whitney U was not significant, another approach was tried. As previously mentioned, the participants also answered profiling questionnaires, see appendix D, this data was used to see if there was a difference when analysing participants who choose adventure games as one of their preferred genres of video games. The results for the adventure flow scores can be seen in appendix E. First, the Shapiro-Wilk test is shown in table 5.

Table 5. Adventure_No_Wanda shows (p<.05) thus a nonparametric test will have to utilised.

Tests of Normality

		Shapiro-Wilk	
	Statistic	df	Sig.
Adventure_No_Wanda	.249	14	.012
Adventure_Wanda	.187	14	.100

As it can be seen, the normal distribution yielded close-to the same results as in the original analysis. The data, mixed between nonparametric and parametric, was thereafter used with a Mann-Whitney U, which can be seen in table 6.

Table 6. The Mann-Whitney U for the Adventure data set.

Test Statistics

	Adventure_Flow
Mann-Whitney U	81.500
Z	-.763
Asymp. Sig. (2-tailed)	.445
Exact Sig. (2-tailed)	.458
Exact Sig. (1-tailed)	.229

The results is ($p<.229$) which means that the results is not significant, thus these results would not be able to reject the null hypothesis either.

13. 4. Source of Error

Uploading both of the game versions to a few selected forums on the internet could possibly have created some distortions in the data was collected. Several issues come to mind in using this kind of method to reach more test subjects. One is that the identities of the test subjects are unknown. That is partially what the profiling questionnaire is there for. With the use of the profiling questionnaire it is possible to filter out outliers, but more direct control over who test the games could have been preferred. This is due to another issue, which is the limited control over the physical settings of the test area of the subjects who tested after having downloaded the game themselves. In theory, these subjects could have tested the game in a highly disruptive environment, full of other people or loud sounds or music. These are all possible flow disrupters and could have had an effect on the outcomes of those tests. Still, in order to achieve a satisfying amount of test subjects the method of sending out the game to selected forums was chosen. Yet another issue with this test method is that if test subjects had any questions before or after the test, these could not be answered by the developers, except on the different forums or by email. This may have discouraged subjects from either taking the test due to lack of motivation because of their questions, or limited their understanding of what they had to do during the test. Attempts were made to give very clear information as to how the test should be performed, and what the subjects had to do, but given that the developers were not physically present, any questions could not immediately be answered.

14. Conclusion

As the games industry continue to grow, more research and studies into games' effects on people needs to be conducted. The research question for this project was "How can an interactive adventure game, which can help determine if there is a difference between flow levels with and without the presence of an interactive companion, be developed? How can it be determined if there is a difference?" The hypothesis for this project was "A companion with artificial intelligence will have a positive effect on flow in Stranded". In order to attempt to provide an answer for this hypothesis, a video game called Stranded with two different versions was successfully designed and implemented. One version had an AI accompanying the player throughout the game, and the other did not. Game assets were created in several software packages, including 2D and 3D packages like Adobe Photoshop and Autodesk Maya, as well as audio packages such as Audacity and Melodyne. The assets were imported into the game engine Unity in which the actual game was assembled and created. Programming for the game was done in Microsoft Visual Studio and written in C#. The two versions of Stranded were used to gather quantitative data about test participants' levels of flow during test sessions. Given that the only difference between the game versions was the presence of the companion, the data was assumed to help reject the null-hypothesis. The data was collected using quantitative methods, via test participants answering a questionnaire on a likert scale, ranging from 1 to 5. The Shapiro-Wilk test was used to test for normal distribution of the data, in order to determine which analysis method would be used to analyse for difference between the game versions, and in extension, the effect of the companion on the players' flow levels. The data showed that one set was parametric while the other was not. In order to further analyse on the data, the nonparametric analysis method Mann-Whitney U was used. The Mann-Whitney U revealed no significant result, however. Thus, the null-hypothesis was not rejected. Additionally, normal distribution analysis, as well as a Mann-Whitney U test were performed on participant results, for which the participants had expressed interest in adventure games. These were analysed because the genre that best describes Stranded is adventure. These tests also revealed no significant result.

References

Adams, E. (2010). *Fundamentals of Game Design.* Berkeley, CA: Pearson Education.

Autodesk. (2013, September 4). *Introduction to polygons*. Retrieved from Autodesk: https://knowledge.autodesk.com/support/maya/learn-explore/caas/mne-help/global/docs/maya2014/en_us/files/Polygons-overview-Introduction-to-polygons-htm.html

Barwood, H., & Falstein, N. (2002). More Of The 400. San José, CA, United States of America.

Barwood, H., & Falstein, N. (2016, April 4). *THE 400 PROJECT*. Retrieved from www.finitearts.com: http://www.finitearts.com/pages/400page.html

Chen, J. (2007). Flow in games (and everything else). *Communication of the ACM, 60(4)*, 31-34.

Coulson, M., Barnett, J., Ferguson, C. J., & Gould, R. L. (2012). Real feelings for virtual people: Emotional Attachments and interpersonal attraction in video games. *Psychology of Popular Media Culture, 1(3)*, 176-201.

Csikszentmihalyi, M. (2004, February). *Mihalyi Csikszentmihalyi: Flow, the secret to happiness*. Retrieved from TED: https://www.ted.com/talks/mihaly_csikszentmihalyi_on_flow#t-285479

Csikszentmihalyi, M. (2008). *Flow: The Psyhology of Optimal Experience.* New York, NY: HarperCollins e-books, p. 74.

DeMarco, T., & Lister, T. (2013). *Peopleware: Productive Projects and Teams.* Crawfordsville, CA: Pearson Education.

Derakshani, D. (2012). *Introducing Autodesk Maya 2013.* Indianapolis, IN: John Wiley & Sons, Inc.

Field, A., & Hole, G. (2003). Choosing a Statistical Test. In A. Field, & G. Hole, *How to Design and Report Experiments* (pp. 274-275). London, UK: SAGE Publications LCD.

Field, A., & Hole, G. (2003). Nonparametric Statistics. In A. Field, & G. Hole, *How to Design and Report Experiments,* (pp. 235-239). London, UK: SAGE Publications LCD.

Field, A., & Hole, G. (2003). Parametric Statistics . In A. Field, & G. Hole, *How to Design and Report Experiments*, (p. 160). London, UK: SAGE Publications LCD.

Fullerton, T. (2014). An Advocate For The Player. In T. Fullerton, *Game Design Workshop: A Playcentric Approach To Create Innovative Games, Third Edition* (p. 3). Boca Raton, FL: CRC Press.

Fullerton, T. (2014). Introduction. In T. Fullerton, *Game Design Workshop: A Playcentric Aprroach To Creating Innovative Games, Third Edition* (p. xxvi). Boca Raton, FL: CRC Press.

Fullerton, T. (2014). Playtesting and Iterative Design. In T. Fullerton, *Game Design Workshop: A Playcentric Aprroach To Creating Innovative Games, Third Edition* (pp. 249, 272). Boca Raton, FL: CRC Press.

Li, J. S., Xu, H., & Chen, S. S. (2012). The Effet of Flow and Motivation on Users' Learning Outcomes in Second Life. *Journal of Education Technology Development and Exchange 5(1)*, 95-108.

Microsoft. (2016, April 18). *Build Unity Games with Visual Studio*. Retrieved from Visual Studio: https://www.visualstudio.com/en-us/features/unitytools-vs.aspx

Razali, N. M. (2011). Power comparisons of shapiro-wilk, kolmogorow-smirnow, lilliefors and anderson-darling tests. *Journal of Statistical Modeling and Analytics, 2(1)*, 21-33.

Rogers, V. (2010, July 09). JIM LEE Reflects On His Job As DC Co-Publisher, Part 2. *Newsarama.com*, p. n/a.

Schell, J. (2014). *The Art of Game Design: A Book of Lenses (2nd edition).* Boca Raton, FL.: CRC Press.

Snyder, C., & Lopez, S. J. (2002). *The Handbook of Positive Psychology*. New York, NY: Oxford University Press.

Sweetser, P. W. (2005). GameFlow: A Model for Evaluationg Player Enjoyment in Games. *Computers in Entertainment (CIE), 3(3)*, 3-11.

Unity. (2016a, April 18). *Animation System Overview*. Retrieved from Unity Documentation: http://docs.unity3d.com/Manual/AnimationOverview.html

Unity. (2016b, April 2). *Emission*. Retrieved from Unity3D: http://docs.unity3d.com/Manual/StandardShaderMaterialParameterEmission.html

Unity. (2016c, April 18). *Inner Workings of the Navigation System*. Retrieved from Unity Documentation: http://docs.unity3d.com/Manual/nav-InnerWorkings.html

Unity. (2016d, April 18). *THE LEADING GLOBAL GAME INDUSTRY SOFTWARE*. Retrieved from Unity3D: http://unity3D.com/public-relations

Unity. (2016e, April 18). *WORLD-CLASS SERVICES FOR MONITIZING, MAKING AND IMPROVING YOUR GAME*. Retrieved from Unity3D: http://unity3d.com/services

Wallop, H. (2009, December 26). Video games bigger than film. *The Telegraph*, p. n/a.

List of figures

Figure 26:
Left: Low-poly version of full door without normal map. Right: Low-poly version with lambert shader and normal map applied.

Figure 27:
Final model of the stone door.

Figure 28:
The UV-map, colour map, ambient occlusion map, and emission map for Wanda in Stranded.

Figure 29.
Illustration of the principle of Ambient Occlusion

Figure 30.
Left: Furniture with assets from World of Warcraft. Middle: Sword from DOTA (Retrieved from: https://hydra-media.cursecdn.com/dota2.gamepedia.com/thumb/4/45/Cosmetic_icon_Sword_of_the_Eleven_Curses.png/256px-Cosmetic_icon_Sword_of_the_Eleven_Curses.png?version=46dd70aee40062a98736f11dc7009b38). Right: Plant from Stranded.

Figure 31.
Audio recording setup

Figure 32.
Melodyne workspace.

Figure 33.
The animation tool in unity.

Figure 34.
UI design for dialogue.

Figure 35.
Dialoguer flowchart for creating dialogue.

Figure 36.
Topdown view of the Stranded map showing the different zones

List of tables: