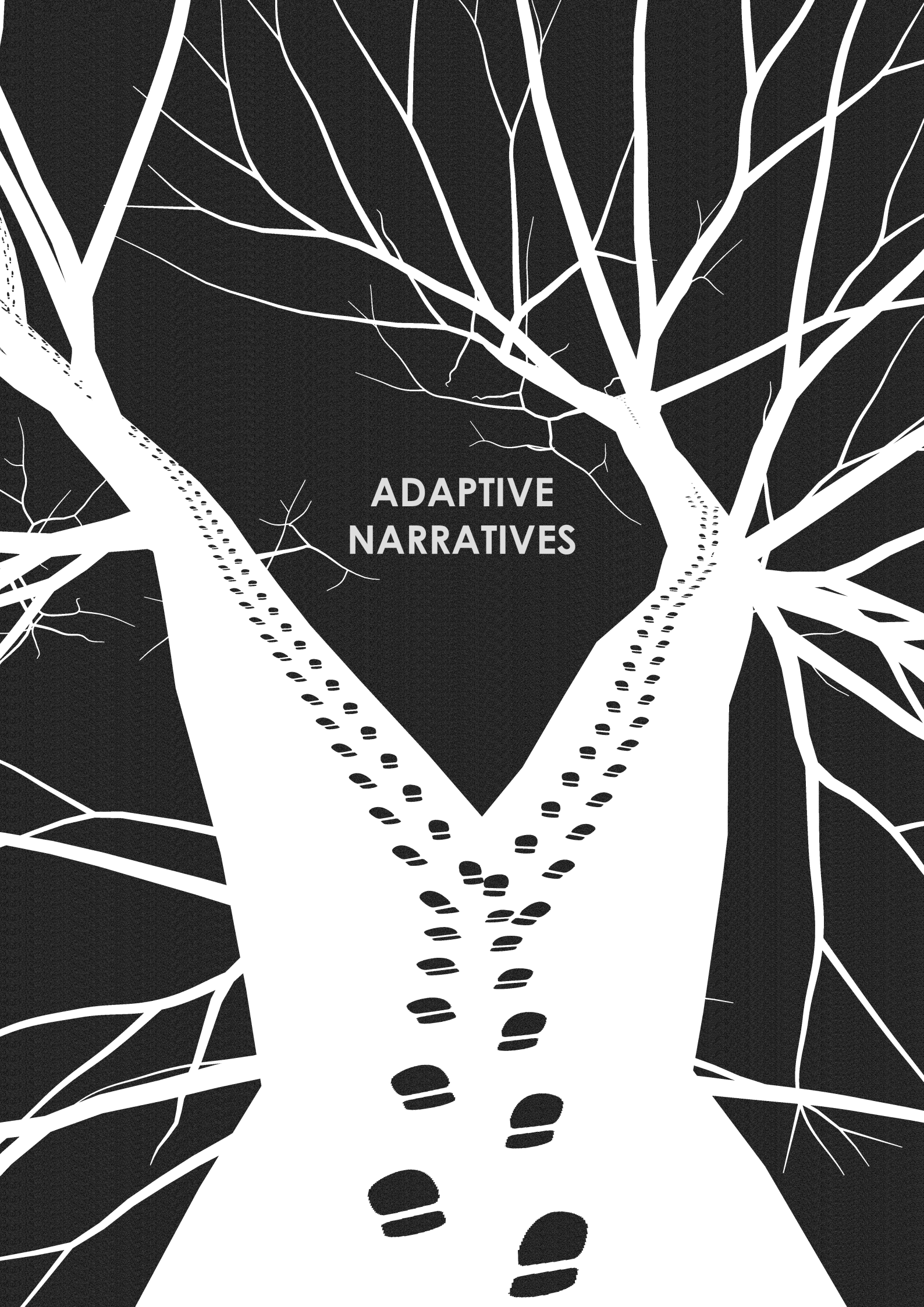
ADAPTIVE
NARRATIVES

Aalborg University Copenhagen

Semester:
MED 3

Title:
Adaptive Narratives

Project period:
Fall 2019

Semester theme:
Visual Computing - Human Perception

Supervisor(s):
Lars Reng

Project group no.:
308

Members:
Asmus Eg Nielson Andresen
Daniel Claes Thiesen
Emil Linddahl Christensen
Marc Bisgaard Petersen
Scott James Naylor

Aalborg University Copenhagen
A.C. Meyers Vænge 15, 2450
Copenhagen SV, Denmark

Secretary: Lisbeth Nykjær
Phone: 9940 2470
lny@create.aau.dk

Abstract:
The development of realtime photorealism in 3D imagery paired with an increasing interest in both TV-series and video games, this project explores ways of adapting photorealistic narratives in realtime, depending on the viewers facial expression. Through an analysis, image processing, branching narratives and realtime rendering technology was explored. A product was created through an iterative development as a proof of concept. This product would detect a user's smile and change the weather in a virtual environment. The product was tested on 34 participants, who were intrigued and generally liked the concept. It was further evaluated by an expert, who confirmed that it served as a proof of concept. In the future the product should be able to detect more facial expressions and adapt the narrative unbeknownst to the viewer. However, further research, development and testing would be needed.

1 Reader's Guide

This report has a bit different structure due to there will be shown multiple production instead. The report will start with an Introduction, Analysis, and Method. After Method, there will be a chapter about the production overview which will tell how the flow is being determine through a figure. This will results in four productions where each of them is covering design,implementation and Reflection of each productions. After it go to a Discussion, Evaluation, conclusion and Future works.

Contents

2 Introduction

In the world of interactive cinema, the fact that the user has to make a conscious decision is a problem as it breaks immersion. The fact that viewers have to consider their options when a branch in the narrative arises and explicitly inform the system of their choice can break immersion. This project will explore this area and provide a possible solution [1].

Traditional interaction design focuses on explicit interactions. This is to clearly express without room for confusion to the computer, what the computer has to do. These explicit interactions are not unidirectional. Computers mostly respond to explicit input with explicit output. For example, a pop-up on your computer notifying you of a software update that is scheduled for later. This pop-op may have an option for the user, to postpone the update. Clicking this update is what constitutes an explicit instruction. Two hours later the computer will remind you of the update. We also all know how annoying this can be at times, when the computer demands your attention during a presentation or when you are driving. How can we make interfaces less demanding of attention [2]?

Implicit interaction is not as well understood in interaction design. This is the process of the interface understanding what you want without any explicit instruction. Humans are innately talented at this form of communication. We can express a multitude of moods and ideas through tone of voice and body language such as facial expressions. What if an interface could detect these emotions, analyse them and respond to them in an appropriate way [3]?

Companies such as sightcorp have created facial expression recognition software and employ it to obtain analytics on audiences and customers from a wide range of industries. For example, to foresee how an audience will react to a certain advertisement before it is released. This is a relatively new area of implicit interaction design. It may have a lot of potential to bolster other industries that are less explored regarding facial expression detection and analysis [4].

Implementing an interface that responds accurately to a person's emotion can make for a highly personalised experience. Personalised content through interactive digital narratives is becoming increasingly popular with releases such as "Bandersnatch" - an interactive movie with a branching story line released on Netflix. An interactive narrative can be explained as *"a story in which the reader has the opportunities to decide the direction of the narrative, often at a key plot point"* [5]. It transcends a passive action into an engaging experience. However, in the case of most releases like Bandersnatch, this is still an explicit interface as the viewer must explicitly choose and click on a direction for the story. If such an experience could be combined with an interface that facilitates implicit input such as facial expressions - could this create a more seamless and immersive experience of this interactive content?

There is a slight hindrance on the feasibility of these interactive movies; that is the enormous amount of planning and production time involved. Bandersnatch is five hours of content, which most viewers will see less than half of. An interesting area of research at present is the area of computational creativity. Humans use stories to give meaning to experiences, this is uniquely human and is termed narrative intelligence. Recent developments in artificial intelligence have proven that we can program computational creativity. DeepMind (an artificial intelligence company) has been creating artificial neural networks to mimic the human visual brain. This has made it possible for a computer to compose artwork that can be indistinguishable in creativity from human art. However, computational creativity does not stop with images, it can also compose narratives. If an artificial neural network could create a narrative structure or just add details to an existing story; that story could be in theory, endless [6].

We have the story. But, how do we create the actual movie? Realtime photorealism is fast approaching with new technology being released constantly. Such as NVIDIA's new ray tracing graphics cards. These new technologies and with future advancements mean that an astonishing

level of detail can be interacted with at speeds upwards of 60 frames per second [7].

Figure 1: Example of a realtime photorealistic scene. [8]

What if the aforementioned points were combined? To create an interactive branching narrative that was determined from the viewers facial expressions and governed by artificial intelligence. Could this make for a more personalised, seamless and interactive experience? With the introduction of 5G networks, the internet can be accessed at very high speeds. 4G today has a theoretical maximum speed of 125 megabytes per second. 5G has a theoretical maximum speed of 1.25 gigabytes per second. In theory, these experiences could be streamed online using popular platforms such as Netflix as a medium [9].

2.1 Initial Problem Statement

"How can you detect smiles in response to a musical or video stimuli?"

3 Analysis

3.1 Introduction

In the following chapter, the initial problem statement will be explored and researched and by the end of the analysis, a final problem statement (FPS) and a list of design requirements will be presented. This will be achieved by researching different aspects of the initial problem statement. This will lead to a more specific problem statement and provide information that can be used as a foundation to the project and a solution to the FPS.

3.2 Physical emotion

The following section will look at the physical form of human emotion. This knowledge will assist the creation and understanding of software that is able to recognise these emotions. A vital part of human communication is through facial expressions, understanding the physiological features of these should increase the efficiency and effectiveness of the software.

Humans show emotions in various ways, these emotions can in some cases be measured and quantified for research or design purposes. To mention a few, there is facial expressions, the conductivity of skin, heart rate, pupil dilation, body-expression and they can also be measured to an extent by doing brain scans like MRI [10].

These emotions are controlled by the nervous system which changes chemicals in reaction to the world around us and how we experience it. A vital part of emotions is the human cognition, making it possible for us to understand the environment we are in and observe its changes and how those affect us [11].

Being able to read these emotions could help design a better experience for the user or consumer of any given technology or application. Being able to precisely read a user's facial expression in real time can assist the designer on how to customise an application so that each user will have a better experience with it as opposed to the one-size-fits-all approach that has previously been used [12].

Major facial expressions like the widening of the mouth, raised eyebrows and showing of teeth often last from a half second to four seconds on average. With a camera recording four frames per second, the fastest of these major expressions can be captured reliably [13].

Microexpressions are facial expressions that are shown for very short periods of time. The analysis of microexpressions aims to study the voluntary and involuntary emotional responses to various stimuli. Charles Darwin hypothesised these in 1872 when he proposed that humans are not able to fully control their facial expressions [14]. The human face will generally show one of the seven universal emotions which are anger, sadness, fear, disgust, surprise, happiness and contempt. These can be shown either simultaneously or conflicting with each other. They are controlled by the amygdala and is basically a bio-psycho-social reaction that lasts for a one thirtieth of a second up to four seconds on average depending on whether it is a microexpression or a major facial expression [15].

Studying these facial expressions using a camera with a high frame rate would facilitate the reading of the 'true' emotion of a person in relation to what they are experiencing. Like with the major or macroexpressions, microexpressions show the same seven types of emotions, but like mentioned above they happen much faster and can be easily overlooked, if not studied carefully. It is also interesting that microexpressions happen across culture, personal background or language. This expands the possibilities for the use of microexpression detection in product development [13].

As mentioned above, humans have evolved vision that is capable of recognising facial expressions. Therefore, the next chapter will look into how human vision works.

3.3 Human vision

Human vision is being looked into as it works in a comparable way to a camera. This should aid in the understanding of how camera's and computer vision works overall. Being able to recognise certain features through visual stimuli is something that animals and humans have done for millennia, recently however humans have taught computers to do the same.

The concept of computers detecting objects can then be used in the design of various applications and media. But, before dwelling into how computers could do this, lets take a look at how humans do it. When humans see an object and memorise it, the visual systems creates a representation of the object and that is what enables us to identify objects of the same category, even if we have not necessarily seen the object before. For example, one might easily identify a tree, but exactly what kind of tree and its exact properties might still be unknown. For humans there are 3 primary issues concerning object recognition: image clutter, object variety and variable view. Image clutter meaning that, usually, there are many objects in view at all times, some on top of each other and therefore might not be easy to recognise. Object variety is the same as the tree example given above, where we are faced with many types of the visual input representing the same object category, trees, buildings and cars for example. Variable views refers to the fact that objects can look very different, depending on which angle it is being perceived from. So how do human perceive objects? Familiarity and experience with certain objects in various conditions stored in the memory and categorised as being one and the same [16].

3.3.1 Perceptual organisation

"Abrupt discontinuation in color and/or brightness in the scene usually correspond to a boundary or corner" [16] Humans see objects that are bound by edges, meaning that the color contrast that is seen by our visual system, recognises these variances in contrast as edges. However, the visual system must also, apart from seeing edges and regions that those edges encapsulates, be able to differentiate between what is part of an object and what are parts of the background. This is called the figure/ground theory. Figure refers to the region of an object which is perceived to be part of that same object, and ground being what is perceived to be the background for the given object. This is how the visual system is capable of recognising objects in complex scenes [16].

3.3.2 The human eye

The eye is an important organ to analyse when researching human senses. Since this project relies on computer vision, it is interesting to compare how the eye works different to a camera. The human eye has different components that work together to convert light into neural signals that the brain can understand. Firstly, the light enters through the cornea and toward the pupil. The iris (the coloured part of the eye) is controlling the amount light coming through the pupil by contracting and expanding, which means the pupil will become larger or smaller depending on the amount of light. When the light has passed through the pupil, the light is met by a transparent structure called the lens which refracts the light. The lens can be controlled by muscles in the eye depending on how far away the light originates from. This way the light can be refracted and focused properly onto the retina. One point worth mentioning is that when the light is refracted it flips the image upside down.

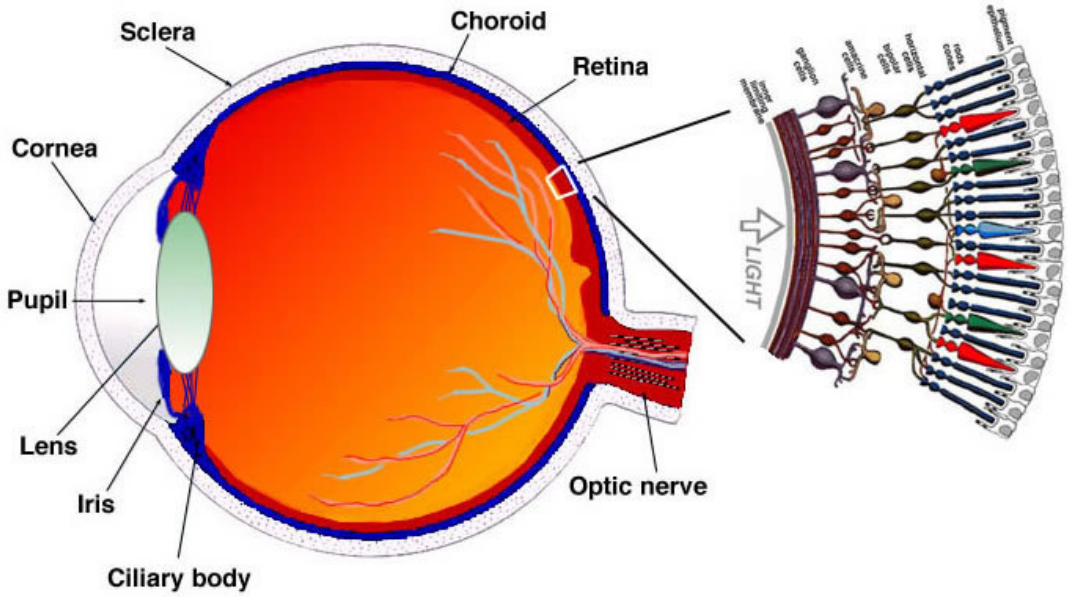

Figure 2: Inside the human eye [17].

The retina is located inside the eye and is collecting the light through photoreceptor cells which translate the light into electric signals called neural signals. These photoreceptors come in two kinds, they are called rod receptors and cone receptors, which have different functions. Primarily, rods are used in low light environments and cone cells are used in light environments and are capable of perceiving colours. The information from the photoreceptors will then be sent to the optic nerve and decoded in the primary visual cortex, here the image is flipped due to it being upside down from refraction. With this, it is time to get an understanding of how a computer, through similar methods, can have vision too [16].

3.3.3 Similarities between eye and camera

A digital camera, like the eye, receives light and converts it into data that can be interpreted, for instance, by a computer. The way the camera obtains this information is similar to how the eye does. The camera uses a lens which has the same functionality as the lens in the eye and, as seen in the eye, the image is flipped. The amount of light traveling through the lens is controlled by the aperture. The aperture works in a similar fashion to the iris, by covering the lens and controlling the amount of light coming through [18].

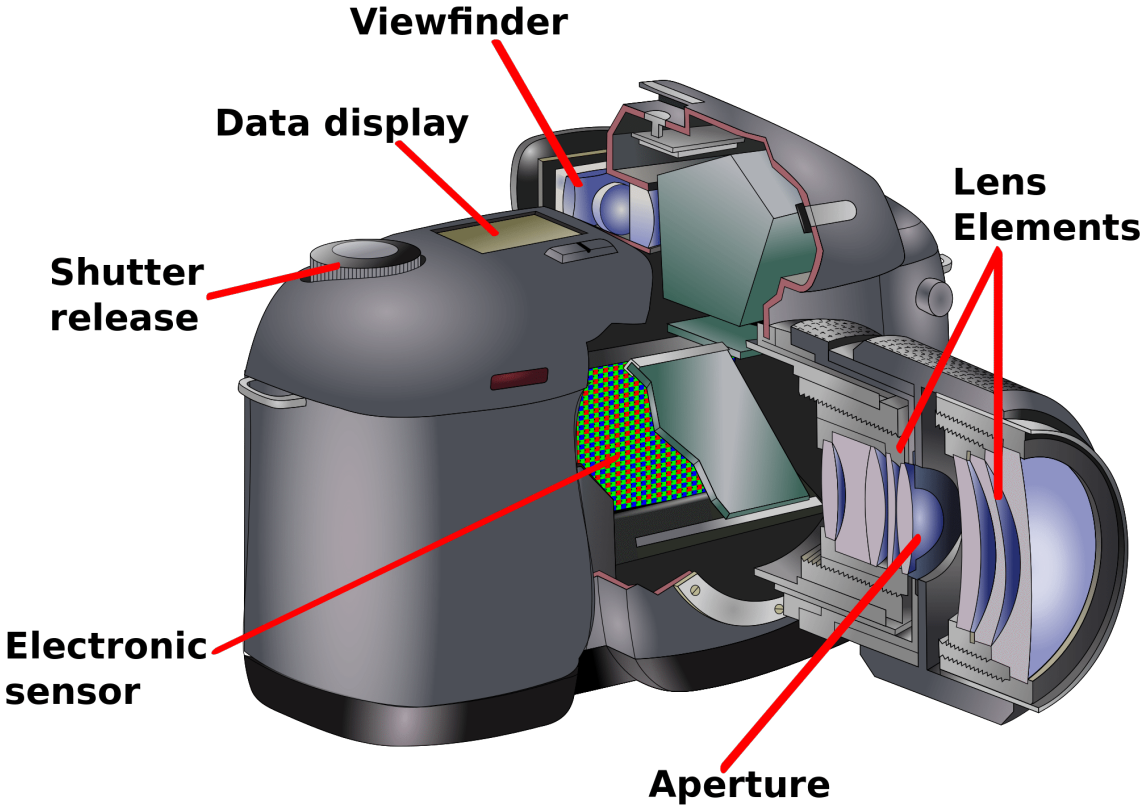

Figure 3: Inside the digital camera [19].

The equivalent of the eye's retina is the camera's sensor. The light is translated into an electronic signal which is measured by its wavelength, and that wavelength correspond to a certain colour. This colour is then assigned to the corresponding pixel [20].

The resulting data which makes up the final image can then be passed into a computer, which concludes the process known as image acquisition. Subsequently, the image is ready for manipulation and analysis, this is known as image processing.

3.4 Image Processing

As stated in the IPS, one of the goals of this project is to detect a smile (2.1). In order to accomplish this goal, image processing must be researched. Specifically, the elements of image processing that relate to computer vision will be explored. Relevant filters, operations and algorithms will be presented in this section. These include; point processing operations, such as greyscale conversion, neighbourhood processing, such as median filter, morphological operations, such as reduction, dilation and segmentation, such as edge detection.

3.4.1 Point Processing

Point processing is a form of image processing where a specific pixel is changed based on its own value, not the values of the pixels surrounding it. Two point processing examples will be presented here: greyscale conversion and thresholding.

In RGB colour images, each pixel has three different values corresponding to the amount of red,

green and blue it contains, these range from 0 to 255 bits. Each of these values correspond to the intensity of that colour. Occasionally, when colour data is not required, it is beneficial to convert an image into a greyscale image for optimisation purposes. Greyscale conversion is done by reading each pixel's RGB values, and then calculating the average of all three, setting the new grey color to that of the calculated value [21]. The equation would look like this:

$$(R + G + B)/3$$

Thresholding is a simple segmentation operation and a form of point processing that can be used in skin detection, colour detection and for converting images to binary. It works by detecting the values of individual pixels and assigning them a binary value according to whether they are within or outside a given threshold. For example, if we are trying to detect skin colour, we know that there is more red than blue or green colour in the skin. Each pixel's RGB values will be read and compared to the RGB values of skin colour. If the pixel is within the given skin color threshold, it will be changed to white, if not, it will be changed to black. This will be repeated until all of the pixels have been converted to binary and only the skin coloured pixels will be shown in white. The RGB values of the threshold can be altered to detect any colour [21].

3.4.2 Kernel Correlation

One form of pixel manipulation uses neighbourhood processing to change one specific pixel as seen, for example, in mean filters which are normally used for blurring images. An important concept to understand in neighbourhood processing is the kernel. A kernel in image processing is usually a two dimensional matrix with coefficient values, this kernel then reads the value of the pixels it is applied to (see figure 5) and changes the origin pixel's value depending on the coefficients, before moving to the next pixel. The size of the matrices and the coefficients can be altered which can lead to many different results. The bigger the kernel, the more neighbors are included resulting in stronger blur. The size is usually an odd number, as the kernel is often centered around one specific pixel. In the case of the mean filter one specific pixel is changed to the average of both the pixel itself and the neighbouring pixels that are within the kernel. For example, if we are taking the average of nine pixels, the value of these pixels are summed up and divided by nine. The new value is then applied to the pixel in the centre of the kernel (As seen in figure 4) [21].

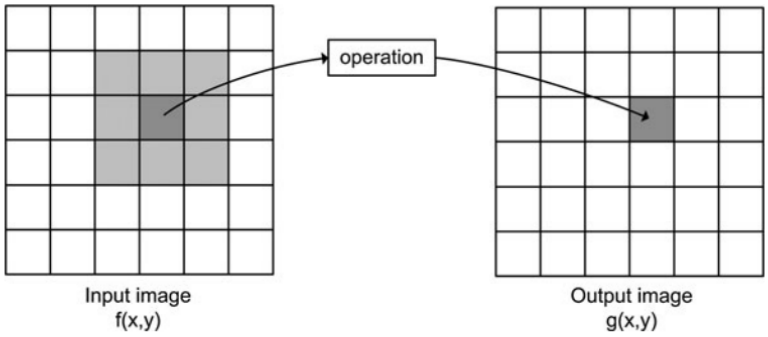

Fig. 5.2 The principle of neighborhood processing. To calculate a pixel in the output image, a pixel from the input image and its neighbors are processed

Figure 4: Neighbourhood processing concept [21].

To make sure that all of the noise is removed, the kernel needs to be applied to every pixel in the image and each pixel within the kernel is multiplied by the kernel coefficients. This is known as kernel correlation [21].

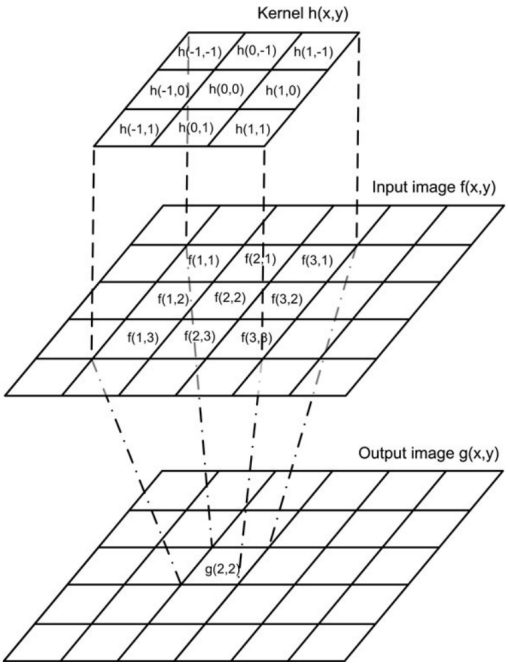

Figure 5: kernel Correlation [21].

They are then divided by the sum of the kernel coefficients, and the average is calculated. Kernel coefficients can be changed to anything, and different numbers create different outcomes. For example, the Gaussian blur filter has a specific kernel that holds unique coefficients as seen in figure (6).

$$\frac{1}{16}\begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix}$$

Figure 6: 3x3 Gaussian kernel[21].

These coefficients puts more weight to pixels that are closer to the center of the kernel. Here, the Gaussian distribution is used for calculating the kernel coefficients. It should be noted that the values specific to Gaussian blur change with the size of the kernel, the example in figure (6) presents a 3x3 kernel [21].

Edge detection is similar to threshold in the sense that it works by looking for significant changes in greyscale. This significant greyscale difference is, where the edge detection method calculates

the edge to be. Within edge detection there are several filters, one of these is the Sobel filter.

The Sobel filter, works by calculating the gradient of a given image then emphasising the edges of the major spikes in the gradient. The Sobel filter is integer based and points out differences in the horizontal and vertical lines of an image. This helps the computer to easier process the information and save computational power. The Sobel filter uses two 3x3 kernels which when combined with the source image consists of vertical and horizontal line detection and are able to enhance those lines so that the output is just the edges of the image. In order to make the edge detection more accurate a threshold can be added which removes noise to prevent unimportant edges from being detected [21].

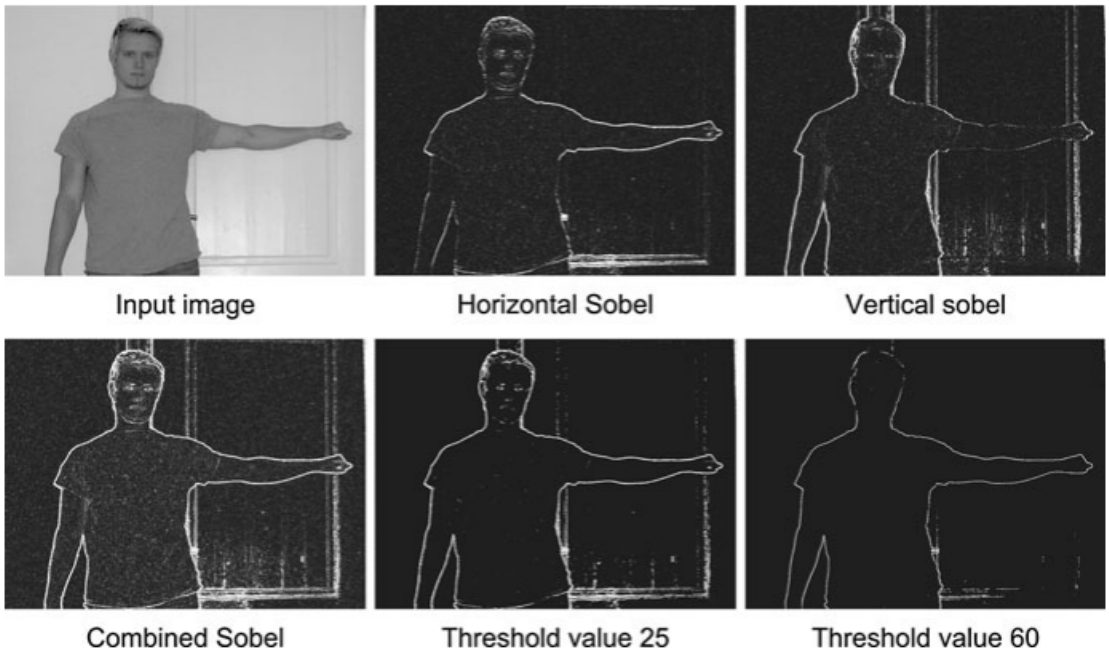

Figure 7: An example of how a sobel filter works [21].

3.4.3 Morphological Operations

Erosion and dilation are the fundamental morphological operations in image processing. The term morphological relates to the shape or morphology of features in an image. These operations are usually applied to binary images. However, they can be applied to greyscale images using slightly different implementations. Erosion and dilation relate to, in simple terms, the shrinking or growing of image features respectively [21].

Erosion is the shrinking of image features and can be useful in many situations. For example, if you have two image features that are connected by a thin bridge, erosion could erode that bridge away, separating the features. It works by implementing neighbourhood processing, with a structural element (or kernel) which has an origin which can be any index of the two dimensional array [21].

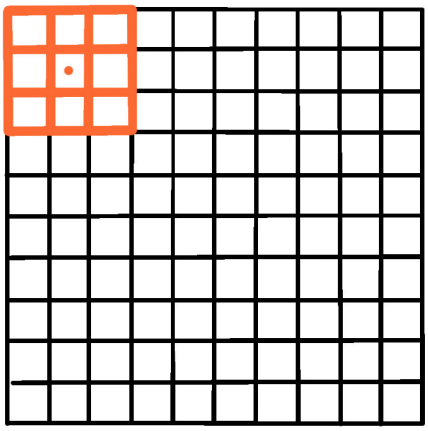

Figure 8: Visualisation of a 3x3 kernel on a 10x10 pixel image.

In the case of erosion, the kernel reads the binary values of the pixels within itself. If the sum of these pixels are equivalent to the size of the kernel (for a 3x3 kernel this would be 9), or in other words if every pixel within the kernel is white, the origin pixel remains white. However, if one pixel is black, the sum of the kernel would therefore be 8 and the origin of the kernel will be changed to black.

Dilation works in a similar way but produces the opposite effect. When the kernel reads the pixel values it looks for at least one white pixel. If there is one or more white pixels, the origin of the kernel is changed to white. If there are no white pixels, the origin of the kernel remains black. These new values are applied to a new image, this is done so the already changed pixels do not affect the neighbouring pixels [21].

Figure 9: Original image > threshold > dilation > erosion.

As shown in figure (9) the original image must first be converted to a binary image by performing a threshold operation on, in this case, the brightness of the individual pixels. Once there is a binary image, we apply dilation, which as you can see in the image dilates the white areas. For instance, the black areas on the ice disappear. After that, the erosion operation makes the white areas decrease in size and the image becomes more black. In figure (9) a large kernel for the erosion operation was used to clearly demonstrate the effect.

When the operations are performed in this order, that is to say, erosion and then dilation, it is

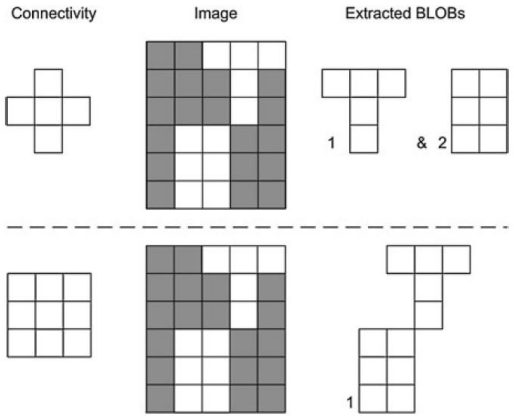

Figure 10: example of how 4- and 8-connectivity works [21].

termed opening. If the operations are performed with dilation first and then erosion, it is known as closing. These are very popular morphological operations used in image processing. For example, if you have a circle with a hole in it, you can perform the closing operation, this will dilate the circle until the hole is filled and then erode the circle back down to its original size. What remains will be the circle in its original size without the hole [21].

The median filter is a form of morphological operation as it does not use kernel correlation. Instead the values around the center pixel are sorted, the median is found and then replaces the center pixel. This is a more commonly used noise reduction operation as it has a weaker blur effect and outperforms the mean filter at removing the noise [21].

3.4.4 BLOB extraction

When an image is converted into a binary image, BLOB (Binary Large Objects) extraction can be used to identify groups of white pixels. In order to group together related pixels, connectivity is used to assign them a under a common label, each one of these labels represents a BLOB. The two types of connectivity that are used are 4-connectivity and 8-connectivity. These types of connectivity are used in an algorithm called grassfire. Connectivity describes the order which a program checks neighbouring pixels if they are connected to the base pixel [21].

The grassfire algorithm can use either 4-connectivity or 8-connectivity depending on how big the search zone for related white pixels should be. When using the 4-connectivity, only the neighbouring pixels to the right, left, above and below in regards to the center pixel will be checked. When using a 8-connectivity, it checks the same directions as the 4 connectivity but it also checks the diagonal pixels (above left, above right, below left and below right) of the center pixel.

When the algorithm is started, it will check each pixel to see whether it is a black or white pixel. If it is a white pixel it will label it and use connectivity to check for neighbouring pixels. If there is a neighbouring pixel which is white, it gets the same label as the first pixel found. Once it has found all the neighbours to that pixel, it will search for a new white pixel which will then get a new label. These groups of pixels will then become BLOBs. Each BLOB has individual labels which are stored in a list, that holds the locations of the pixels related to each individual BLOB [21].

When the BLOBS have been labeled, it is possible to get information such as how big they are or where they are. For instance, when removing unwanted BLOBs, finding out how big each BLOB is, can be a way to sort them. This can be done by counting the pixels in the BLOB, and from there a restrictions can be made to not show BLOBS outside a certain pixel threshold. To find them,

a frame or figure can be used to show the BLOB's boundary. One of these methods are called "bounding box" and is using the minimum and maximum x and y values for the pixels related to a certain BLOB. This can form a box surrounding the BLOB and can then show where the object is located on the source image [21].

3.4.5 Haar Cascades

Object detection through Haar Feature-Based Cascade Classifiers is a detection method proposed by Paul Viola and Michael Jones in their paper from 2001, *"Rapid Object Detection using a Boosted Cascade of Simple Features"* [22]. The overall method relies on training, through machine learning algorithms, a cascade function, also known as a classifier, by presenting it with a large amount of positive and negative images. A positive image is an image including the object that should be detected, for example a face, while a negative image doesn't include this object. After the classifier has been trained, the features it detects as being positive images are extracted. This extraction happens when various feature extraction templates (See figure 11) are placed on top of the image and the sum of the pixels under white areas are subtracted from the black areas. This subtraction will give an indication that the relationship of these values could be part of a face [23].

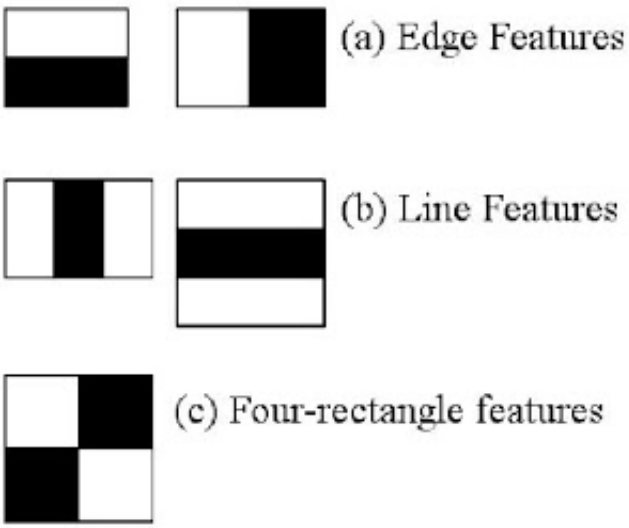

Figure 11: Various forms of feature extraction templates [23].

However, a very large amount of features are needed to determine if an item in the image is the object that is being looked for, and instead of running tens of thousands of feature-checks, the integral image algorithm was implemented, what this does is that instead of having to calculate each pixel in a grid, it generates the sum of values in the rectangular subset of the grid, effectively decreasing the amount of calculations needed to extract features [24].

Then, to further increase the efficiency of the algorithm, the AdaBoost algorithm was implemented into the method [25]. The AdaBoost algorithm is an algorithm that in short works by assigning weight to features that is known to identify positive images and negative images, running this learning algorithm several times makes it assign more weight to features that have been shown to create false-positives or negative images effectively creating what is known as a final classifier. This final classifier is the sum of all the weak classifiers together, which in turn forms a strong classifier. It is strong because it increases the efficiency of knowing what not to look for. With these algorithms in place, the amount of processing needed to determine an object is much lower

than the initial method. However, one last implementation made the algorithm what it is today, a highly efficient object detection algorithm. This last step was implementing the concept of a cascade of classifiers. Instead of applying all features at the same time, they are grouped into various classifiers and tested individually. If a classifier fails once, it is discarded and declared not usable. However, if a classifier has success, it keeps running until it can confirm that the object is positively in the image [23] & [22].

This means that an application can be made that determines if the viewers in front of a camera is smiling or not. This information about the viewer can be used for a multitude of applications, one of which could be for adapting the narrative of a media experience in response to the viewer's expression. This will be explored in the following section.

3.5 Interactivity, Personalisation and Branching Narratives in Media

This purpose of the following research is to ascertain why, if at all, interactivity, personalisation and "choose your own adventure" style of narratives can can serve to improve the experience of a media experience. Furthermore it will take a brief look into the history of personalisation and how it has infiltrated certain areas of media. Following the information gained, it will become increasingly clear how this project can take advantage of personalisation and in what context.

For many years now, marketers have been using data in order to personalise advertisements. There are many reasons why they would do such a thing. However, one salient point stands out; that is involving the Reticular Activation System in every person's brain. This powerful system is bombarded with up to 100 million impulses each second. Its job is to filter out any information that we deem irrelevant or unnecessary. Also known as selective attention, this is what causes us to naturally orient to ideas we are invested in. This is known as the cocktail party effect [26].

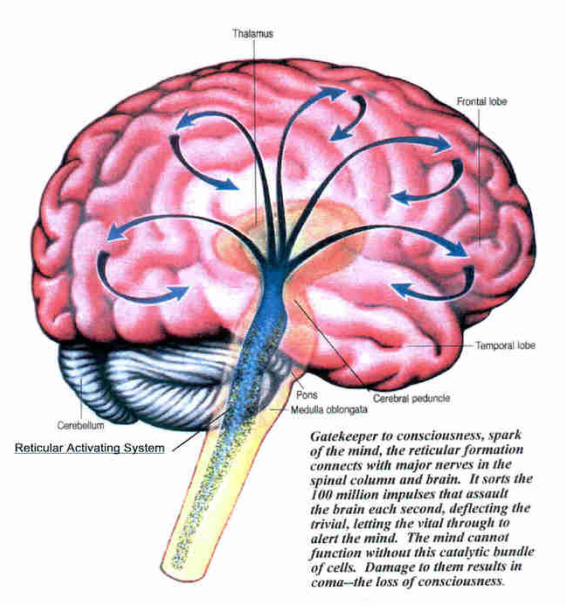

Figure 12: The Reticular Activating System [26].

Taking this into consideration, personalisation can have a massive effect on an individual's likelihood to notice, retain and interact with the advertisement. However, this fact is not exclusively pertaining to advertising. It can in theory apply to anything - even a narrative.

Interactivity in media has been around for a long time. Interactivity in narratives is relatively new. With early attempts at understanding the platform such as the choose your own adventure franchise first released in 1976. However, this kind of narrative personalisation didn't penetrate the games industry until years later and is only just today beginning to gain some noticeable traction in the film and television industry [27] & [28].

There is much debate in the online community over the difference between an interactive movie and a game. Many video games today can be personalised to an extent. Narratives can be followed in various directions and change according to the players decisions throughout the game. This is very present in a lot of role-playing games (RPG). Where, quite often, the player can choose between being good or evil and this can dramatically influence the story the player experiences. However, this has not always been the case. During the digital revolution, the golden age of arcade games bolstered, at the time, a very niche market. Video games took off and with it the innovation of said games.

One of the first 3D platformer games was Crash Bandicoot from Naughty Dog in 1996 [29]. It featured levels that the player must guide the protagonist through on a linear basis. While this was revolutionary at the time; the narrative was very weak, it was not personalised, and it was nothing in comparison to the latest games coming out today. However, it was noticed by the public and by video game developers. It was not long before Rockstar games would release Grand Theft Auto 3 [30].

Figure 13: Crash Bandicoot (1996) [29].

Grand Theft Auto 3 [31] was one of the first games to allow the player to explore a living, breathing open world environment on their own terms. It proved a huge success and would prove as an inspiration to other developers to follow a path of increasing the complexity of narratives and immersion in their own games. Since then, narrative complexity has only increased.

3.5.1 Personalisation in Games

How has personalisation infiltrated the gaming industry? And, how does it relate to branching narratives? As a result of constant advancements in hardware, software and creative innovation from game developers, different ways of personalisation have emerged. It is important to understand the different ways of implementing personalisation in games. Allowing the player to customise their in-game character is a very common technique of personalising the experience. Looking at games such as the NBA 2K [32] series where the player can import and project images of their own face

onto the in-game character to become the basketball player. Forza Horizon 4 [33] is a racing game that has a vast list of names to choose from when beginning a new game, if you can find your name, the car's virtual assistant will address you by name. The fallout series [34] along with many other RPG games have long since facilitated character customisation. However, a less obvious form of personalisation in games comes from branching narratives. For example, games such as Heavy Rain allow the player to constantly decide on the direction of the narrative at key plot points (See figure 14). The game offers 17 different endings, each one personalised to the choices that have been made by the player.

Figure 14: An example of a branch in narrative in the form of choice (Heavy Rain) [35].

3.5.2 Interactivity in Cinema

Researching into interactive cinema will reveal how it has been done in the past and present, what the reception of these experiences were and whether there is tangible audience for them today. Interactive cinema is by no means a new phenomenon. There is much debate in the online community over the difference between interactive cinema and a game [36] & [37]. The overlap between the two is a murky area and is only getting murkier as people try to divide the two. Interactive cinema is arguably more cinematic and narrative driven, with the option to alter the plot of the movie as a viewer. Interactive cinema was bolstered with the digital disc and digital disc players. With the invention of data that could be read in a non-linear fashion allowed for branching plot lines. However, one of the first forms we an see of interactive cinema was in 1967 – well before the invention of digital discs. The film was Kinoautomat and required a host to take the stage at each branch in the story line to take a diplomatic vote on which plot line to follow. In 1995, Interfilm and Sony collaborated to create Mr Payback, an interactive film where the audience could interact with the narrative using handheld controllers. This was received very poorly by critics. With one critic saying *"We don't want to interact with a movie. We want it to act on us. That's why we go, so we can lose ourselves in the experience." – Roger Ebert* [38]. Today we see modern adaptations of this same idea with features such as "Bandersnatch" from Netflix. The black comedy allows the audience to choose between two options at various points during the movie, which will change the narrative. This results in a high level of personalisation, as the audience gets to decide what happens. The reception of Bandersnatch has proven that there is an audience for this hazy area between games and movies. But, there is a lot to improve upon.

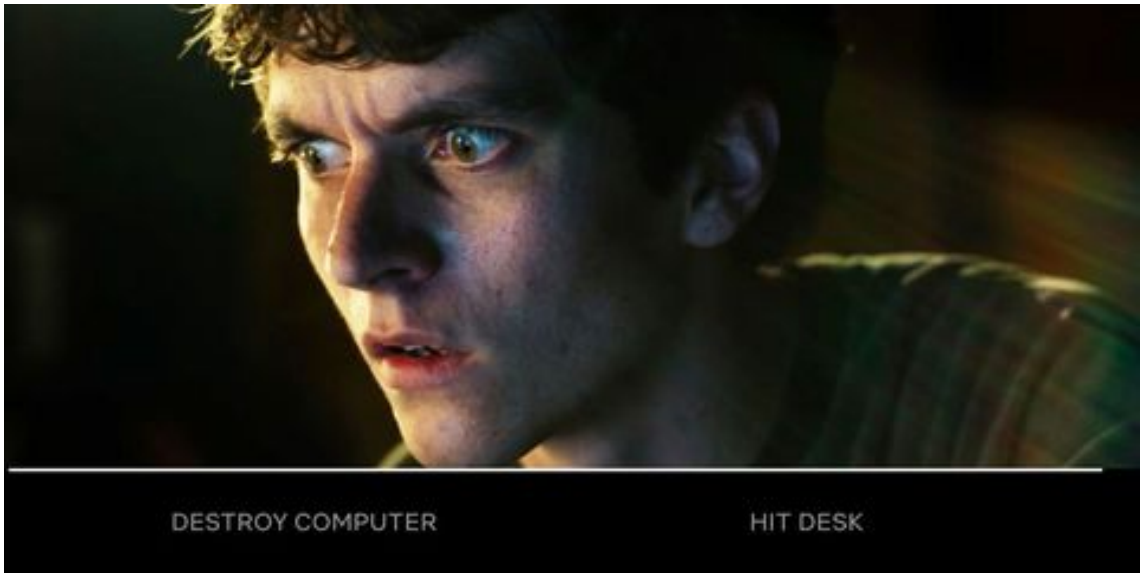

Figure 15: An example of a choice in Bandersnatch [39].

3.5.3 Personalised narrative through facial expressions

Facial expression has been explored earlier in the analysis. This kind of detection could be used in theory to personalise virtually any form of media; from advertisements to a music playlist. However, branching narratives and interactive cinema have begun to rise in popularity in recent years. With streaming services and other companies such as, Netflix, YouTube [40] and Twitch [41] investing in the technologies required to give this content a platform, the former niche area of cinema could be bolstered significantly towards becoming mainstream. With this said, explicit interaction with narratives can arguably reduce the viewer's immersion, there is room for improvement in how the audience interacts with the narrative. Taking the above points into consideration, interactive films and branching narratives could be programmed in such a way as they adapt and progress completely in accordance to the viewers facial expression. It would make for a seamless interaction between the viewer and the media.

3.6 Real Time Photorealism

As mentioned in the introduction 2, branching narratives take extensive resources to produce simply due to the increased content that is required, especially in real-life cinema. In the game industry, having a premade storyworld built digitally in 3D, while initially time consuming, allows the user to go where ever they desire. This allows the narrative to be built in a modular fashion in different areas of the game world. However, until recently photorealistic scenes (scenes that are indistinguishable to the real world) have had to be pre-rendered. Not only is this time consuming but it does not support multiple story lines unless they are specifically planned. Real time photo-realism is an important aspect of making a seamless experience and a natural flow while playing games. The real time aspect is especially important since interruptions and breaks such as loading screens can negatively affect immersion while watching an interactive movie or playing a game. in short real time photorealism will allow the story to be changed in real time while maintaining the realism.

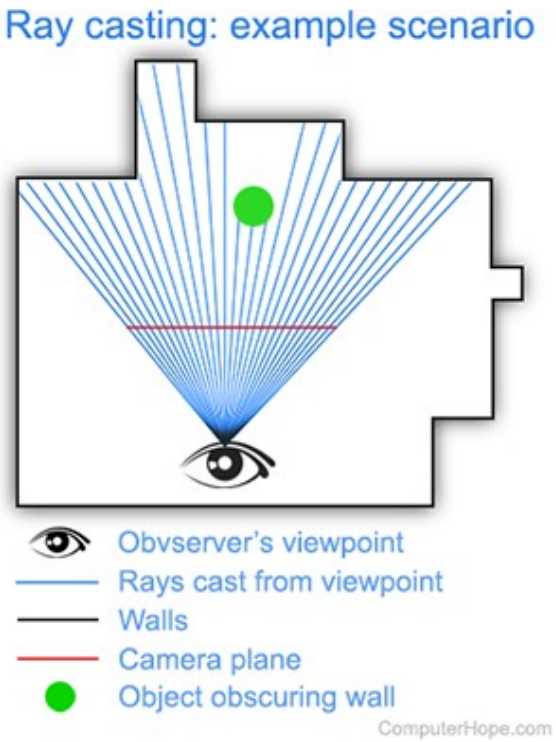

Figure 16: A visualisation of ray casting [42].

In real time photorealism an important aspect is ray casting. Ray casting is a state of the art algorithm technique which replicates how light acts in the real world through complex calculations. It is used in modern games, animation and computer-generated imagery (CGI) to increase the level of realism. Ray casting allows for better shadow realism as well as depth perception. It does this by locating the source of light and casting rays of light on the surfaces of the computer generated world and objects within it. This can be intensive to have running while playing large scale open world maps, so ray casting is only activated where the player is viewing at that moment; not on the whole map. [43].

3.7 Computational Creativity

"In (a game developer's) case we always have to think about how players might react to each depiction of a character or story line, and that's the part we can't predict. But if we manage to get over this hurdle, then I regard video games as a greater medium to provide people with deep emotional and exciting experiences." - *Yoshinori Yamagishi* [30].

As mentioned in the introduction 2 the planning and production time involved in creating branching narrative real-life movies can be extraordinary. Therefore, researching a possible way in which artificial intelligence could govern the story could open up new and innovative ways of creating these experiences. As quoted above creating deep emotional experiences with characters is a problem in game development due to fact the developers don't know how the player will react to the character. However, what if the character changed in response to the user's input. An interesting field of study into computational creativity lies in computer narrative intelligence. A machine learning algorithm created by Mark A. Finlayson is able to extract *"culturally-relevant plot patterns from sets of folk tales."* and detect patterns in the narratives which can help create other stories [6]. This is termed analogical story merging, experts in the field have created algorithms that can create both fiction and non-fiction narratives. In theory, this could be used to create

vast interactive experiences if the algorithm was able to alter the narrative in accordance with emotional input from the viewer.

3.8 5G

In the previous chapter, computational creativity was discussed and when combined with the idea of a realtime photorealistic interactive experience that can change depending on the physical expression of the user, it becomes clear that hardware requirements will be substantial. With the spike in prices from graphic cards after the cryptocurrency boom of 2017, these services will become very attractive to the everyday gamer as the prerendered world can be streamed straight to their device and does not require much processing from the computer. However, this will require a suitable bandwidth to achieve the steady seamless flow that a user might expect when using such media, which is why the emergence of 5G internet technology and cloud streaming is being explored.

Recently subscription services have taken off and the possibility of streaming games or interactive experiences over 5G is relevant to explore as a possible platform. This is due to the fact of the high expense a user would have to endure if they were to purchase the hardware capable of running photorealistic realtime worlds of any reasonable scale. The word latency describes the response time of a network, that is how quick the user receives a response to an input. Cloud based gaming is currently possible but at present not accessible to much of the world as streaming game worlds over a network with acceptable latency, frames per second and resolution due to bandwidth and data transfer speeds is not available on a global scale.

"In some ways, 5G will be necessary for these sorts of streaming services to really take hold," said Matthew Hudak, research consultant at Euromonitor International [44].

During peak time, fibre optic can only offer an average 11.7 milliseconds of latency. Google Stadia as of November 2019 is a new game streaming subscription service. It however, relies on fast bandwidth speeds. They recommend 10 Mbps to stream a game in 720p with stereo sound and 35 Mbps to stream 4K resolution with 5.1 surround sound [45].

Network Latency

Network	Latency (milliseconds)
2G	300-1000ms
3G	100-500ms
4G	50-100ms
5G	1-10ms

Figure 17: A comparison of generations of mobile network latency [46].

4 Final Problem Statement

In this sections, the journey from the Initial Problem Statement and to the Final Problem Statement will be presented. The Initial problem statement was as follows.

"How can you detect smiles in response to a musical or video stimuli?"

In the beginning of the analysis, facial expressions were researched. It was found that major expressions can last anywhere between, 0.5 seconds and 4 seconds and because of this they can easily be detected by a camera with a minimum of 4 frames per second. Microexpressions were also explored, these expressions were found to be as fast as one thirtieth of a second. This would make them difficult to detect reliably. Due to this, they will not be the focus of this project, although, they are worth exploring further in the future (3.2).

Following this, human vision was explored and its similarities to computer vision. This was researched to gain a better understanding of image acquisition and the basics of how both humans and computers can detect and recognise objects through vision (3.3).

Next, various image processing filters and operations have been explored. Point and neighborhood processing was investigated, as well as how BLOB detection helps detect and label objects in images. These preprocessing filters will be implemented and tested going further to gauge their value to the face and expression detection process. Haar cascades were found to be useful for both face and smile detection which use trained classifiers to detect objects that are in our interest to find (3.4).

After facial expressions had been explored and the relevant image processing that could help to detect them had been researched, the possibility of using this technology in conjunction with branching or adaptive narratives was investigated. It was found that people in general are attracted to ideas they are invested in due to the reticular activation system. This means that personalisation of a narrative can bolster an audience's experience. Additionally, the history of interactive narratives was explored to gain insight into how it has been done before (3.5).

The advances in photorealism was explored and how new technology can facilitate realtime photorealism. This means that immersive realtime experiences can be created that the audience cannot distinguish from reality (3.6).

Computational creativity was briefly researched as a possible solution to the long and expensive production times involved in the creation of branching narratives. The concept of having an artificial intelligence algorithm to create or govern narratives could significantly reduce production times. However, this is state of the art technology and would take significant knowledge and time to implement. Therefore, it will be considered for future works and will not be implemented in this project at present (3.7).

Finally, 5G was investigated as a possible streaming network for adaptive narratives. This is due to the large bandwidths of the new technology. However, due to 5G being a new technology, it is not feasible to implement or test this aspect of the project. Therefore, it will also not be implemented in this project but, will be considered for future works (3.8).

Due to the research presented in the analysis and concluded above, the following final problem statement was created:

"How can one, through the utilisation of image processing and facial analysis, detect a user's face and smile and use this data to adapt a narrative in a virtual realtime environment?"

5 Design Requirements

5.1 Non-functional requirements

- The program must work on a computer in conjunction with a camera (3.4).
- The program should adapt the environment depending on users facial expression (3.6) (3.5).
- The program can be built using open source software.
- The program should be testable on various users.

5.2 Functional requirements

- The software must be able to receive live video input.
- The software must be able to detect face and smile from a live video input (3.4.5).
- The software must be able to output data when a smile is detected (3.5.3).
- The software must implement realtime visual functionality (3.6).

5.3 Technical requirements

- The live video feed must operate at a minimum of four frames per second (3.2).

6 Methods

In this section an overview of the methods used in the Design, Implementation and evaluation chapters will be made. This will be done in order to give insight into the methodical approach for the report, as well as which theories and methods will be used in the project.

6.1 Expert Interview

An expert interview is an in-depth interview in which an expert in a relevant field is asked questions in order to gain knowledge upon the subject which is being investigated. The strengths of this type of interview is that it can yield new information and can confirm or deny any prejudices made by the group. However, it is time consuming and can yield a lot of data which can make it difficult to organise and analyse [47].

6.2 Convenience Sampling

Convenience sampling is a non probability sampling method which means that everyone does not have the same chances of being chosen for testing, and that people chosen are the people who are around and available at the time of testing. This might also introduce a larger amount of bias in the test results.[47]. Since there isn't an exact target group for people that can enjoy an adaptive narrative, convenience sampling was deemed most effective, because the opinion of everyone is equal as of now, and it is the easiest way of getting a hold of as many participants as possible within a short time [47].

6.3 Mixed Method

The term "mixed method" refers to the method of combining various forms of qualitative and quantitative data gathering techniques. The purpose of this is to gather various perspectives upon a subject that is being evaluated. In the following subsections the individual aspects will be discussed individually in further detail.

6.3.1 Questionnaire

A questionnaire is a data gathering technique that functions like an interview, but instead of having an interviewer conducting the test, the questionnaire will present the questions to the participant in writing. The participant will then be able to answer the question either by selecting a preformulated answer or typing it out in a response text field. Compared to the interview format, a questionnaire can ensure a higher level of consistency as it will be the same questions and format for each participant. Furthermore, questionnaires is an efficient method for data gathering since it allows for a multitude of a participants and effective analysing of data. A negative aspect of using a questionnaire is that it allows for less follow up or further explanation, if their answer is confusing or needs further explanation. Overall, a questionnaire is a great tool for data collection as it allows for several types of data gathering [47].

6.3.2 Qualitative data

Qualitative data is a form of data collection method, which refers to data that is non-numerical. This includes written, verbal and other non quantifiable responses. In the case of this project it refers to the types of questions where the participant will be asked to go into further detail on certain questions or describe in longer terms what they have experienced in the test. The main reason for using qualitative data is that it allows for answers with more substance, that can then be discussed in greater detail and create new insights and provide a way in which people can give feedback that would have otherwise not been considered [47].

6.3.3 Quantitative data

Quantitative data is another form of data collection method, and this refers to data that can be quantified and analysed statistically. This kind data of data is especially useful for rating an experience or aspect of a program, understanding the likeability or intuitiveness and gathering demographic data [47].

6.4 Scrum and iterative methods

The scrum method is an iterative approach to application development. Scrum is a way to manage and prioritise what tasks are done, in progress or need to be done. The scrum method works by dividing the workload into so-called sprints, at the end of each sprint, a test or evaluation is done. Depending on the results of the test or evaluation a new sprint is started that either builds on the previous or takes a new direction. Sprints can vary in length. However, ideally sprints last from one to four weeks. The scrum master has manager experience and knowledge of how long certain tasks take, they control what is known as the scrum board, which contains the backlog of tasks to be completed [47].

7 Production Overview

The following chapters will outline and show the iterative approach that was taken in order to answer the final problem statement. These chapters are based on both the design requirements and the knowledge gained from the analysis. They are made in accordance with the methods explained above in order to design, implement and test the applications. These iterations will be presented as productions moving forward and they represent the development and testing of the product.

7.1 Overall approach of development

The design requirements should all be implemented in the final production, however, they will be introduced iteratively in the productions. Therefore, it is necessary to have an overview of the design that will present the goals that need to be reached. In order to visualise the goal of the final product in a generalised form, a flowchart has been made. Each production will implement progressively more components of the flowchart. The final production should implement all of the flowchart components.

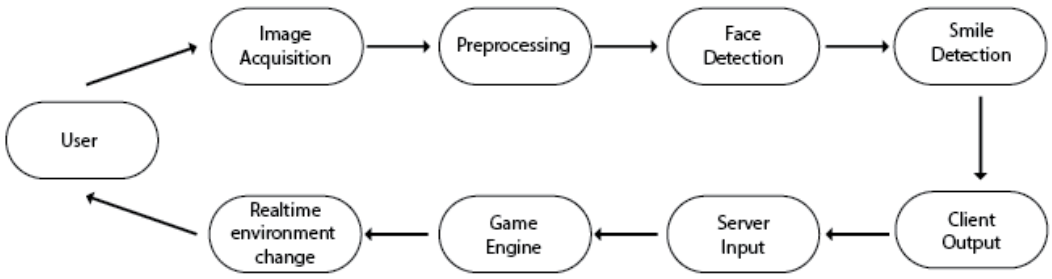

Figure 18: This is a flowchart that shows the outline of the design process.

8 Production 1

8.1 Production Introduction

The goal of this production is to reach the stage of preprocessing in the flowchart (see figure 19). It aims to accurately detect skin colour from a live video input. If successful, this implementation could be used in following productions.

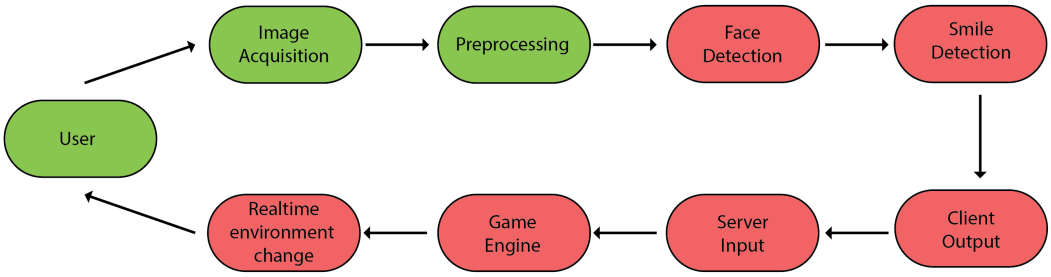

Figure 19: Flowchart stage goal for production 1.

8.2 Design

For this first production, the goal was to utilise preprocessing filters (3.4.2) in order to detect a skin color in a live video feed. To do this, the group wanted to use methods and operations that were taught in the "image processing" course. This would further the groups understanding of how these methods and theories work, furthering the knowledge of image processing for skin color detection.

Image acquisition will be used to receive a video feed from a camera device (3.3.3). When the video feed has been established, preprocessing filters will be applied, there are numerous orders in which the filters can be applied to find a skin tone. Thresholding is used for colour detection and thus it will be used to detect skin colour (3.4.1). In order to get a cleaner result, blur filters will be used (3.4.2). These blur filters can be applied before or after the thresholding operation and will yield different results depending on this. In order to clean up the image further, erosion and dilation operations can be applied respectively (3.4.3).

Now that the image processing operations and filters that can be used to detect skin have been established. The following section will describe the implementation of these.

8.3 Implementation

The initial plan was to make a skin detection in the program, Processing. Processing is a language build upon the programming language Java and is used for its visualisation capabilities. This was to get an understanding of how different image processing filters can be applied and how to acquire the video feed of the computer's camera.

First, Processing's video library is imported since the library has a method for capturing devices like a connected camera. From the library, a built-in variable called 'Capture' is used for the capturing of the camera. the 'Capture' variable will be named 'video'(20).

```
// Step 1. Import the video library.
import processing.video.*;

//Step 2. Declare a capture object.
Capture video;
```

Figure 20: Importing of video library with Capture variable called 'video'.

In Processing, there are two default functions called setup and draw with no return type. The setup function is called the initial frame of the programs lifespan, and the draw function is being called every frame of it's run time. In the setup, the size of the window is set to 1920x960 because the camera output is 640x480, which means that the window could hold six smaller windows in the output format simultaneously, i.e. in two rows of three columns (see figure (21). This would mean that there are six outputs, where five of them are showing the results after filter application and one with the original camera output.

Figure 21: How the windows are shown on the screen.

This was done mainly to check if each filter is functioning correctly. The capture variable 'video' is being declared as an object where it is requesting three arguments, the parent device, the height and width of the output. Lastly, the 'video' object is calling the start method from the video library which begins the capturing of frames from the camera (22).

```
void setup() {
  size(1920, 960);//640x480 default

  // Step 3. Initialize Capture object.
  video = new Capture(this, videoWidth, videoHeight);

  // Step 4. Start the capturing process.
  video.start();
```

Figure 22: The start of the Setup function.

To show the various images with the filters applied, the five images needs to be declared. The image is being created by using the function named 'CreateImage' with three arguments for width, height and color format for the image. The size of the images are going to be the same as the camera and the colour format is going to be in red, green and blue (RGB) colours (23).

```
  image1 = createImage(video.width, video.height, RGB);
  image2 = createImage(video.width, video.height, RGB);
  image3 = createImage(video.width, video.height, RGB);
  image4 = createImage(video.width, video.height, RGB);
  image5 = createImage(video.width, video.height, RGB);


}
```

Figure 23: Declaration of images and end of setup function.

When the code runs, the setup function will then start to loop the draw function. First it is going to check if the camera is available and if that statement is true, it reads the input from the camera. After that, the camera output is shown next to two camera outputs with filters applied. Underneath these, three more camera outputs with filters applied are shown. The first image is the direct output from the camera and the rest is displaying the camera output with filters applied. After that is done, the pixels of the video input is loaded and therefore refreshed, so a continuous video feed is being shown. This is done through the method called 'loadPixels' (see figure 24).

```
void draw() {
  if (video.available() == true) {
    video.read();
  }
  image(video, 0, 0);
  image(image1, videoWidth, 0);
  image(image2, videoWidth*2, 0);
  image(image3, 0, videoHeight);
  image(image4, videoWidth, videoHeight);
  image(image5, videoWidth*2, videoHeight);
  video.loadPixels();
```

Figure 24: Start of draw function with images declaration and loading of pixels from first image.

The filter functions have two arguments per function. The first argument requires an image that the filter can be applied to. The second argument decides where in the video output the image,

with the filter applied, will be shown. This is being used to see the changes the filters make and controlling if the functions work correctly. The 'skin detection' filter comes first is taking the camera output and the changes will be shown on the second image. The threshold function 'skinDetection' uses a double for-loop that is being used to calculate the position of the pixel of the selected image and that equation looks like this:

$$positionOfPixel = xPositionOnTheScreen + yPositionOnTheScreen * widthOfThecameraOutput$$

When the pixel's position is found, its RGB values are stored and this happens for each pixel location. These RGB values are then used in an if-statement to check if they are within the threshold for skin detection. If the values are within the threshold, the pixel will be given the value 255, turning it white and if not, it will be given the value 0, turning it black (25).0

```
void skinDetection(PImage selectedImage,PImage changedImage )
{
  for (int y = 0; y < videoHeight; y++)
  {
    for (int x = 0; x < videoWidth; x++)
    {
      int loc = x+y*videoWidth;

      float R = red(selectedImage.pixels[loc]);
      float G = green(selectedImage.pixels[loc]);
      float B = blue(selectedImage.pixels[loc]);

        if (R > 95 & G >40  & B > 20  & R > B & ( R -G ) > 15)
        {
          changedImage.pixels[loc] = color(255);
        } else
        {
          changedImage.pixels[loc] = color(0);
        }
    }
  }
}
```

Figure 25: skinDetection function.

The third image will show the second image after a blur filter has been applied to it. The function, 'blur' is requesting two images the same way the 'skinDetection' does.

In 'blur' there is a two dimensional array which is 3x3 matrix, where a pixel is in the center of the matrix and its neighbours are surrounding it. Each entries in the array is getting a value from the variable 'v' and after that, a double for-loop is being made. The for-loop index is starting at one and going toward the value of pixel width and height of the camera output minus one. This is done to not include the edges of the image. In the double for-loop, there is a local variable called 'sum'. The 'sum' variable is the cumulative values of the pixels within the kernel that surrounds the current pixel. The new double for-loop is for calculating this sum by looping through the pixels within the kernel.

There are two variable which is "pos" for the position and "val" for the red value of the current pixel, since the RGB values are identical in greyscale, only one of the three values is required.

Before exiting the second double for-loop, the 'val' variable is being multiplied by the kernel entry which effectively averages the values within the kernel. That value is then assigned to the 'sum' variable, which is used to colour the corresponding pixel of the following image (26).

```
void blur(PImage selectedImage,PImage changedImage)
{
  float v = 1.0 / 9.0;
  float[][] kernel = {{ v, v, v },
    { v, v, v },
    { v, v, v }};
  // Loop through every pixel in the image
  for (int y = 1; y < videoHeight-1; y++) {   // Skip top and bottom edges
    for (int x = 1; x < videoWidth-1; x++) {  // Skip left and right edges
      float sum = 0; // Kernel sum for this pixel
      for (int ky = -1; ky <= 1; ky++) {
        for (int kx = -1; kx <= 1; kx++) {
          // Calculate the adjacent pixel for this kernel point
          int pos = (y + ky)*videoWidth + (x + kx);
          // Image is grayscale, red/green/blue are identical
          float val = red(selectedImage.pixels[pos]);
          // Multiply adjacent pixels based on the kernel values
          sum += kernel[ky+1][kx+1] * val;
        }
      }
      // For this pixel in the new image, set the gray value
      // based on the sum from the kernel
      changedImage.pixels[y*videoWidth + x] = color(sum);
    }
  }
}
```

Figure 26: Blur Filter Function.

The median filter, dilation and erosion functions are applied afterwards and these three functions are using a similar process as the mean blur filter. The difference with the median filter comes when the kernel values are found. They are sorted and the median pixel value is applied to the target pixel, this facilitates noise reduction.

The dilation function is cataloguing the brightness of the pixels within the kernel and dividing it by 255 to convert it to a bit, the same goes for erosion. When the double for-loop for the kernels is done, a pixel with a given position's color is going to change. The dilation is checking if the sum of the kernels is higher than one. If that is true, the given pixel will turn white otherwise it is going to be black. The erosion is checking if the sum is equal to the kernel's sum and if that is true, then given pixel will be white, otherwise it is going to be black. After that, a grass-fire function(3.4.4) is being called, and this is being used to find blobs of white pixels and label them.

8.4 Production Reflection

This production began with the goal of detecting objects with skin tone by using theories and methods from the "image processing" course. Using image acquisition to get the output of a camera was quickly done, but there were some trouble with the filters. One of the main problems with them were the order of which they were applied. This production was made during the image processing course and it was brought to the attention of the group that a median filter applied in

the beginning could give a better result. Further testing with different ordering of filters could have improved the noise reduction (8.2). Overall, this production was successful in detecting skin tones. However, due to issues with extensive noise, the following production will take a new approach and attempt face detection.

9 Production 2

9.1 Production Introduction

The goal of this production is to apply a bounding box to a face as an initial step towards the face detection stage in the flowchart (see figure 27).

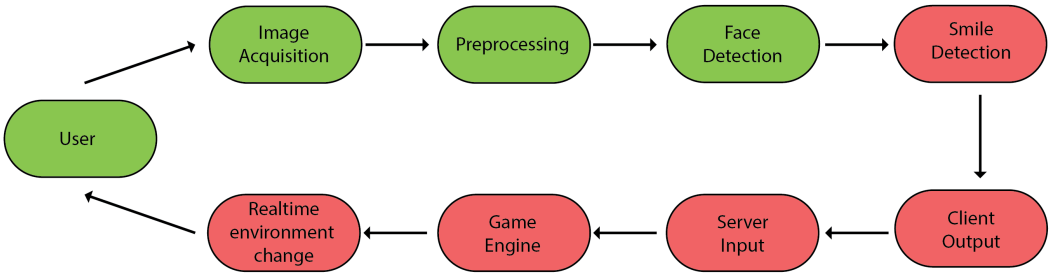

Figure 27: Flowchart stage goal for production 2.

9.2 Design

The goal of this second production greatly resembles that of the first production however this production aims to go one step further, in that the goal is reaching the face detection stage of the flowchart. To do this, we again need to read a video stream of a user through image acquisition, then apply image processing filters to the stream and use this as a step towards face detection. With our knowledge at this point being various image processing methods, it was decided to use these once again. However, in a slightly more optimised fashion with a focus on applying bounding boxes to the faces present in the video stream.

The image acquisition should be achieved through a programming language that will be able to run the needed image processing filters, in this case skin detection (3.3.3) & (3.4.1). It was decided to utilise skin detection because of the success in the first production. It will also be required to perform and apply blob extraction and apply a bounding box to the blob (3.4.4). With the design ideas presented above, we hope to be able to visually detect a face in the aforementioned video stream through skin detection and blob extraction.

9.3 Implementation and Testing

The first production, presented above, aims to apply several image processing methods in order to detect pixels with a skin tone in a video stream. This code is built in many ways similar, but focuses mainly on skin detection and blob analysis. The skin detection method serves as a pre-processing operation that converts skin coloured pixels into BLOBs. It works by detecting pixels in a certain color range in the video stream. When the first pixel matching the threshold is found, a blob is applied. Every pixel within a certain distance and same color range is then added to this blob, making it bigger for each time the draw function loops until there is no longer any

more pixels to add to the blob. It's important to point out that several blobs can be drawn on the video stream simultaneously, this is because we want to be able to detect several faces at the same time, and is facilitated by having an array list of blobs. However, an issue was encountered when doing this, when faces get close to each other, they are added to the same blob because of the distance threshold. The code also implements a timer that clears current blobs every 300 milliseconds before drawing them again. This is because a continuous cycle of blobs being drawn makes the blobs overlap each other, and you are left with a blackish box around the skin and not a rectangular frame. Another issue with this iteration is that blobs are being drawn on all skin it detects, which means it also draws blobs on arms as shown below. Below are screenshots from the video stream showing the issues explained above and also code snippets of the blob class, the delay timer and the code that adds blobs to the video stream.

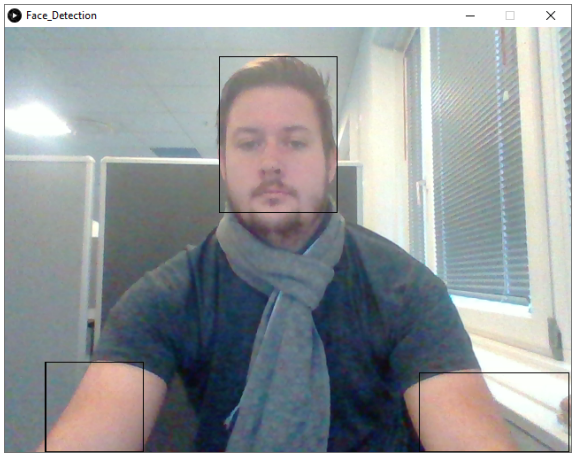

Figure 28: One of the issues was that all skin was detected and had blobs added to it.

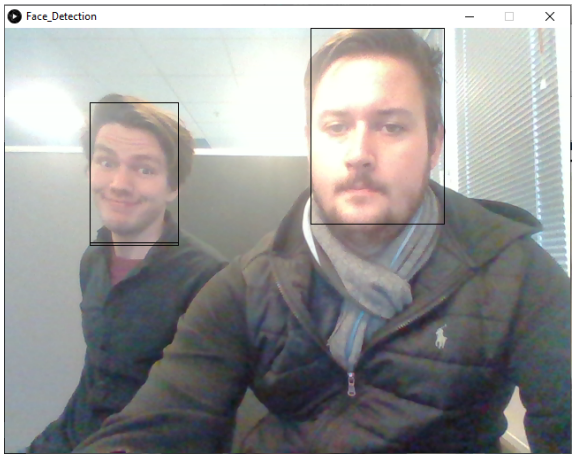

Figure 29: Example of detecting several faces at once.

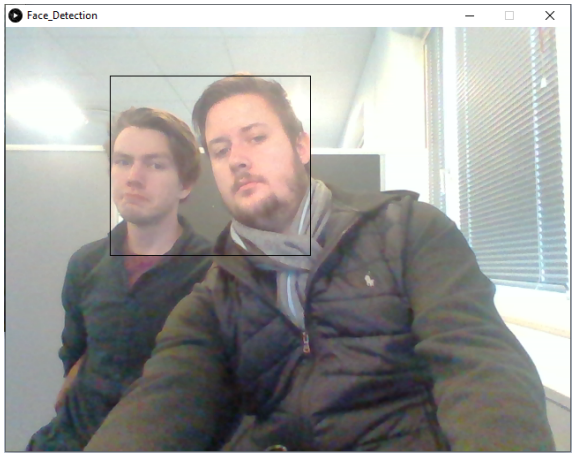

Figure 30: Example of blobs being merged together when distance between skin color becomes lower than the threshold.

```
75          boolean found = false;
76          for (Blob b : blobs) {
77            if (b.isNear(x, y)) {
78              b.add(x, y);
79              found = true;
80
81              break;
82            }
83          }
84
85          if (!found) {
86            Blob b = new Blob(x, y);
87            blobs.add(b);
88          }
89        }
90      }
91
92      for (Blob b : blobs) {
93        if (b.size() > 500) {
94          b.display();
95        }
96      }
97    }
98  }
99
```

Figure 31: The code that adds blobs and displays them using the 'display' method.

```
class Blob {
  float minx;
  float maxy;
  float maxx;
  float miny;

  Blob (float x, float y) {
    minx = x;
    miny = y;
    maxx = x;
    maxy = y;
  }

  void display() {
    //noStroke();
    //hint(DISABLE_OPTIMIZED_STROKE);
    strokeWeight(0);
    rectMode(CORNERS);
    rect(minx, miny, maxx, maxy);
    noFill();

  }

  float size() {
    return (maxx-minx)*(maxy-miny);
  }


  void add(float x, float y) {
    minx = min(minx, x);
    miny = min(miny, y);
    maxx = max(maxx, x);
    maxy = max(maxy, y);
  }

  boolean isNear (float x, float y) {
    float cx = (minx + maxx)/2;
    float cy = (miny + maxy)/2;

    float d = distSq(cx, cy, x, y);

    if (d < distanceThreshold*distanceThreshold) {
      return true;
    } else {
      return false;
    }
  }
}
```

Figure 32: The blob class implements several methods, the 'display' method draws the blob in the video stream, the 'size' method returns the size of the blob which decides if it will be shown or not. The 'add' method adds a new blob to the array list and the last method 'isNear' determines when two blobs should be added together in regards to the distance threshold.

```
 // Calculate how much time has passed
int passedTime = millis() - savedTime;

if (passedTime > totalTime) {
  blobs.clear();
  savedTime = millis(); // Save the current time to restart the timer!
}
```

Figure 33: Implementation of the timer that 'refreshes' the blobs every time 300 milliseconds have passed.

9.4 Production Reflection

The second production presented above was partially successful in the task of detecting a face in a video stream. As seen in (28) where a face is bound by a rectangular box. However, arms and other skin ended up being detected and was applied a bounding box as well. At the same time the bounding box was not correctly applied when faces were too close to each other. This means that the goal of reaching the face detection stage of the flowchart was not fully realised. This was however achieved in the following production partly because of the course of image processing introducing the team to OpenCV and Python, which will be explained in depth below.

10 Production 3

10.1 Production Introduction

The goal of this production is to reach the smile detection stage in the flowchart (see figure 34). As mentioned in the production 2 reflection (9.4), the component of face detection in the flowchart was not fully realised and will therefore also be implemented in this production.

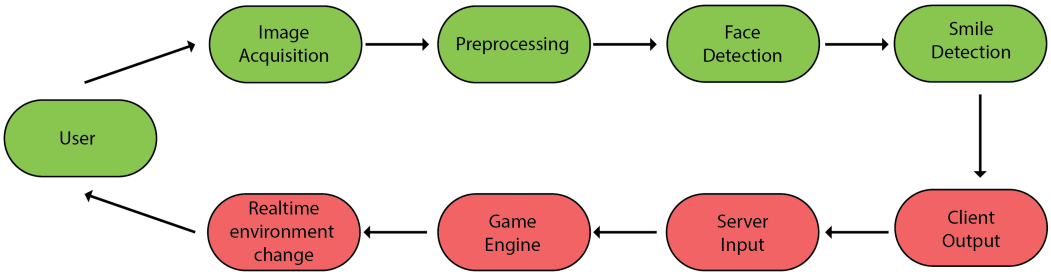

Figure 34: Flowchart stage goal for production 3.

10.2 Design

From the previous productions, the group has a better understanding of how image processing can be used to locate a face in a live video input. However, the implementations of these resulted in unsatisfactory results. OpenCV is an open source computer vision and machine learning library that can be imported into many programming languages. It is a collection of highly optimised methods and algorithms relating to a wide variety of computer vision and machine learning topics. OpenCV has been created and optimised over many years to facilitate realtime computer vision. OpenCV implements optimised algorithms for detecting faces and smiles, which greatly outweighs what could be implemented in processing without OpenCV, in the previous productions. Therefore, from this production and moving forward OpenCV will be [48].

As mentioned in the analysis Haar Cascades are commonly used in face and smile detection. The OpenCV library contains many classifiers for detecting many different objects from a video or image input (3.4.5). Two of these classifiers are for detecting faces and smiles. The Haar cascades are trained with greyscale images, therefore to get an accurate detection, the live video input should be converted to greyscale using the optimised OpenCV method. The cascade classifiers have built in functions that can then search for "features" in the video. If the video contains a face, it will be detected and a bounding box will be drawn around it. Once this bounding box is made, it can become a region of interest to search for a smile. The number of faces and smiles can then be counted and used for further implementation.

10.3 Implementation

As mentioned in the design, this production will switch to implementing the face and smile detection in OpenCV, in order to achieve more effective and efficient results. Furthermore, the group has made the switch to Python, as research and experience gained through the image processing course showed it was more optimized. Python was chosen as the primary programming language for this project because it was taught in conjunction with OpenCV in the "image processing" course.

10.3.1 Smile Detection with OpenCV

First, in the Python code two libraries are imported. These libraries are OpenCV and numpy, where OpenCV is called 'cv2'.

Afterwards, there are two variables that use OpenCV functions. These utilise OpenCV functions that are called 'faceDetect' and 'smileDetect' and they are going to contain the cascade classifiers for faces and smiles respectively. They are using XML files that contain information about what the computer should look for to find a face and a smile. The last variable with OpenCV usage is the variable called 'cam' and is getting information about which camera it should use.

```
import cv2
import numpy as np


#the database of faces
faceDetect = cv2.CascadeClassifier('haarcascade/haarcascade_frontalface_default.xml')
smileDetect = cv2.CascadeClassifier('haarcascade/haarcascade_smile.xml')

#a variable for the webcam
cam = cv2.VideoCapture(0)
```

Figure 35: Libraries and variables in Python.

Next, a while-loop is created, which will be the main body of the script. Here, four local variables are declared and initialised. The first two, 'countSmile' and 'countFace' are integer values and will be used to count how many smiles and faces that are found. The next two are called 'ret' and 'img' and will read the camera's output, where 'ret' is a return value to see if the camera is readable.

After this, the 'gray' variable contains the same output as 'img' but converted into grayscale by using the OpenCV function called 'Color_BGR2GRAY'.

The face detection is implemented through 'faces' where it is using 'detectMultiScale' method from the 'faceDetect' class. This is followed by a for-loop of 'faces' where it is drawing a rectangle around what the classifier thinks is a face as well as adding one to the 'countFace' variable. The areas where faces are found, are going to become regions of interest(RoI) for both the grayscale output and normal output of the camera.

```
while True:
    count = 0
    countFace = 0
    #two variables that is reading the camera
    ret,img = cam.read()

    #converting the img variable to grayscale
    gray = cv2.cvtColor(img,cv2.COLOR_BGR2GRAY)

    #first argument is for the image, second is for the vector of the rectangles, and third argument is for how big the blob needs to be
    faces = faceDetect.detectMultiScale(gray,1.1,5)
```

Figure 36: the beginning of while loop.

These regions of interest will be named 'roi_grey' and 'roi_color' and are saving the area by using the RoI's x, y, width and height. The 'roi_grey' is going to be used for finding a smile, since it contains a RoI that could be a face. That information will be saved in 'smiles' and is going to use the 'detectMultiScale' from 'smileDetect'. Again, a similar for-loop as for faces is made for smiles. This for-loop is checking through 'roi_grey' to see if the classifier can identify something that looks like a smile. If a smile is then found, a rectangle or bounding box will be made surrounding the smile and 'countSmile' will be increased by one.

```
for(x,y,w,h) in faces:
    #drawing a square with x and y coordinates and adding the color and stroke
    cv2.rectangle(img,(x,y),(x+w,y+h),(0,0,255),2)
    roi_grey = gray[y:y+h,x:x+w]
    roi_color = img[y:y+h,x:x+w]
    countFace +=1
    smiles = smileDetect.detectMultiScale(roi_grey,lowerParameter,20)


    for(sx,sy,sw,sh) in smiles:
        count +=1
        cv2.rectangle(roi_color, (sx, sy), ((sx + sw), (sy + sh)), (0, 255, 0), 2)
```

Figure 37: Middle of while loop with the two for loop to find face and smile.

Once the for-loop has completed, a window will be displayed that can show the output of the camera and where the RoI is. Then, an if-statement checks whether there is a face by using 'countFace'. If 'countFace' is above 0 it will then change the variable 'data' to a new string and it will do another check. This check is for 'countSmile' to see if that is above 0. If that is true, 'data' variable will change again. If 'countFace' is 0, then no face is detected and the 'data' will change to "No face detected"(figure 38).

```
cv2.imshow("Face",img)

if  countFace > 0:
    data = "Face without smile"

    if count > 0:

        data = "Face with smile"

else :

    data = "No face detected"


print(countFace)
print(count)
```

Figure 38: Ending of while loop and printing the results in the console.

10.4 Testing

Our initial test goal is to ascertain whether we can accurately measure if a participant is smiling or not. The testing environment will be at a demo-day for 3rd and 5th semester medialogy student at Aalborg university, Copenhagen. Here the medialogists will have an opportunity to test and receive feedback on each others products and projects. For the test, the participants will asked to fill out an anonymous questionnaire.

10.4.1 Procedure

Participants came to the booth and they were asked to sit in a chair in front a computer with the test product running. Here the participant was told about the idea of the product and the outline of what has been made so far, as well as the idea of how the finished product might work. When the participant was told about the product and they had tried to use the product, they were asked to try to smile in order for us to check if the program is able to detect their smile. The test itself took about 5-7 minutes and the participant was afterwards asked to fill out a short questionnaire. The questionnaire included questions about the participants age, gender, interests in TV and movies and questions about what they think about the product and concept as well as security of their private data.

10.4.2 Results

In total there were 20 respondents and to start off the test the participants were asked to answer three questions about their age, gender and how much they watch TV and movies in their spare time. The 14 male and 6 female answered as seen below (39).

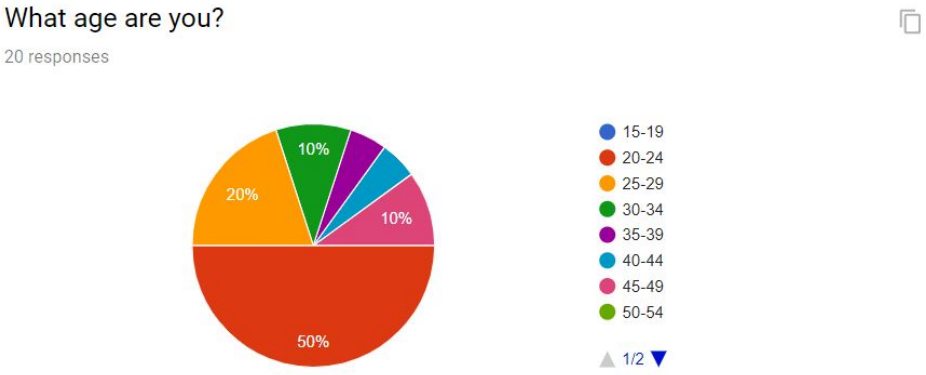

Figure 39: This is a graph that shows the ages of the participants

The figure below will visualize how often respondents watch TV and/or movies. The majority of participants answered that they watch movies or series everyday of the week and on the other hand no one answered that they did not watch movies or series during the week.

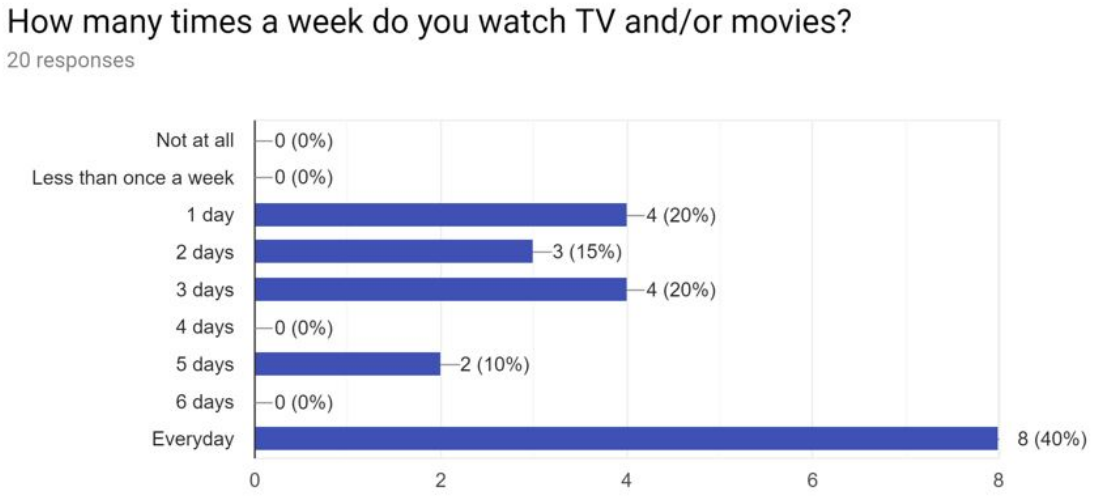

Figure 40: This is a graph that shows how many days a week the participants watch movies and series.

After this, the second installment of the questionnaire begins. In this part of the questionnaire the participants were asked about what they thought about the concept and idea of the product as well as the idea of their data being saved on a database in order to improve the product.

The first question the participants were asked was whether or not they had seen or heard about "Bandersnatch" as it is related to the kind of technology we are looking into and it is a good example of what the product could be used to improve upon. The response is shown below.

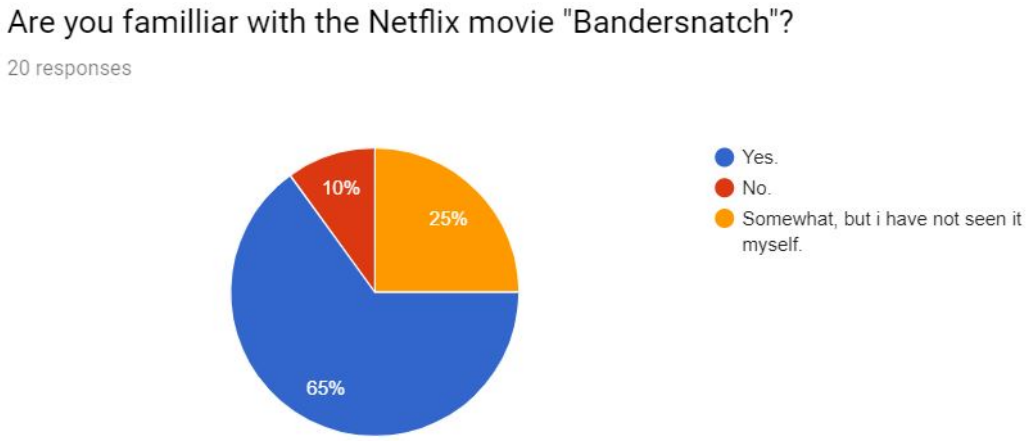

Figure 41: This is a graph that shows how many of the participants that had seen "Bandersnatch".

The second question was "Do you like the concept of a personalised movie/narrative experience?" 14 responded that they liked the concept while 6 said they did not.

When we then asked the participants question three "In your own words, what do you think about

the concept of personalised experiences in movies, TV and entertainment?" this was asked so the participants could elaborate on what they had answered in the previous question.

The negative feedback was mostly pointed towards the fact that some of the participants had concerns about the amount of production that would be needed. A participant said "It's like storytelling for non thinkers. Give me your full vision, not your perceived idea of my likes.", which point to the fact that the participant wants to view a movie like it is art, and want there to be one story told that can be interpreted by the viewer. This participant also said that they were not interested in personalised movie experiences and said in later questions that they would be uncomfortable having a camera looking at them while watching movies. This goes to show that for some people this product is not for them, but as seen with other technology, the more mainstream it becomes, the more likely it is for people to use it [49].

On the other hand most of the participants were positive about the idea of personalised movie experiences and the only real concern was the same as the people that said no, which was that is would demand a lot of extra work and time for the production team to make the movie. This is of course a real problem if movies of the future are made in the exact same way as they are now, but with technological advancements touched upon in the analysis it seems more and more likely that this issue will be less of a factor in the future 3.7. This could be solved with Computer Generated Images (CGI), Artificial Intelligence (AI) and Machine Learning (ML). Furthermore, the participants that had said yes to the previous question answered that the were exited, positive and liked the idea and wanted to see more of it.

In question four and five the participants were asked "Would you be comfortable with a camera detecting your face to extract emotional response that will alter the movie's story or plot. (The data is private and goes nowhere)" and "Would you accept the data being used for optimizing facial recognition and other related optimization?" here 15 participants answered yes and would not mind being filmed. 5 answered no and would mind being filmed and for their data to be used for optimization.

The final question is related to the two previous questions where the participants that said "no" if they would tell why they had answered as such. Here the participants said that they were scared of where the data is going, where it was being stored and who would have access to their data. One participant said "Answered yes to both of the above. However, I am generally reluctant to permit anyone to use any of my data. Thus, guaranteeing protection of my data in an explicit and transparent way would be paramount." and another said "I feel it is disturbing my privacy. I like to just be myself when i watch movies, and knowing a camera is filming me would ruin that." And two other participants said "Privacy" and "Big data..".

10.5 Production Reflection

Overall, this production was a success. Face and smile detection was achieved in an efficient way. This implementation will be used moving forward. The testing was successful and detected the participant's faces and smiles. The results showed that most of the participants were enthusiastic about this implicit form of interaction. The implementation of this production will be built upon in the following production in order for the smile detection to communicate with an adaptive visualisation.

11 Final Production

11.1 Production Introduction

The goal of this production is to complete the flowchart (see figure 42). The previous production implemented a functioning smile detection. This production will build upon this and cover the design and implementation of a game engine environment and a client-server socket, that will ensure the communication between these. Furthermore, the final test procedure and a short reflection of this production will be presented.

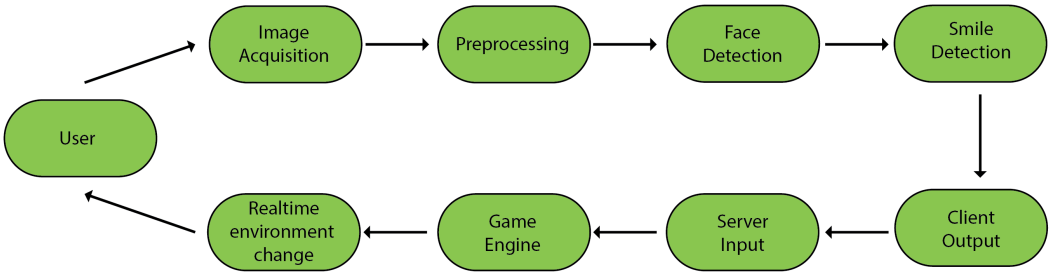

Figure 42: Flowchart stage goal for production 4

11.2 Design

The goal of this iteration is to create a client-server structure connection to a game engine with the python program. Furthermore, the game engine that will be implemented, must support the realtime environment change and visual output to the user (3.6). The design of the client-server structure is explained and illustrated with UML diagrams and more below. The game engine design will also be explained below.

11.2.1 Use Case Diagram

The use case diagram visualises the behaviour of the application that is being developed in regards to the person (actor) interacting with the application.

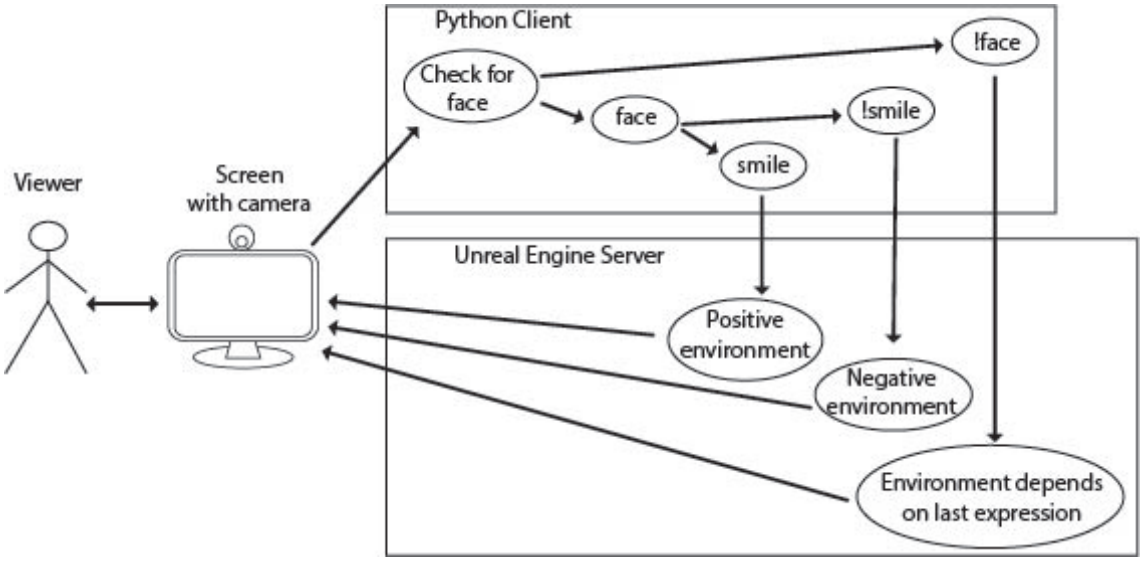

Figure 43: A use case diagram describing the interaction between the viewer and the system (Image by authors).

In this diagram it is possible to see how the client and server communicates. A person is watching some form of video content on a screen. This persons face is then detected by a camera, and that information is being interpreted in the python client. The python client sends the package to the unreal engine server via sockets. The unreal engine server changes the outcome of the program depending of the information it receives, and displays the outcome on the screen for the person watching. This cycle continues until either the camera or the program is terminated.

11.2.2 Class Diagram

The class diagram is a structural diagram showing the classes, functions and operations needed to make the application functional.

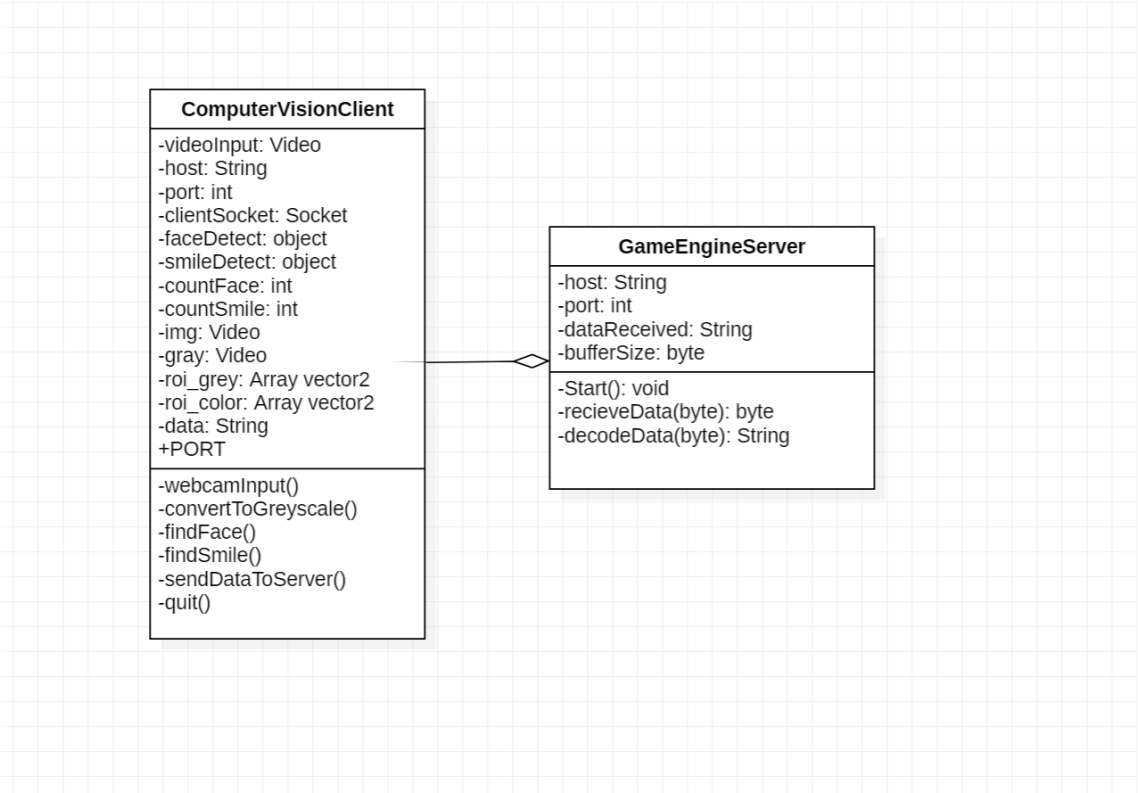

Figure 44: Class diagram design of server and client. (made by authors)

The "ComputerVisionClient" program is going to have the same IP address and port as the "GameEngineServer" since they are going to have a one way communication. The game engine server is receiving packages from the client. The server is then translating the data that the game engine can accept and from there, an event in the game engine is triggered.

11.2.3 Sequence Diagram

The sequence diagram shows the functionality of the application and how these communicate with each other and themselves.

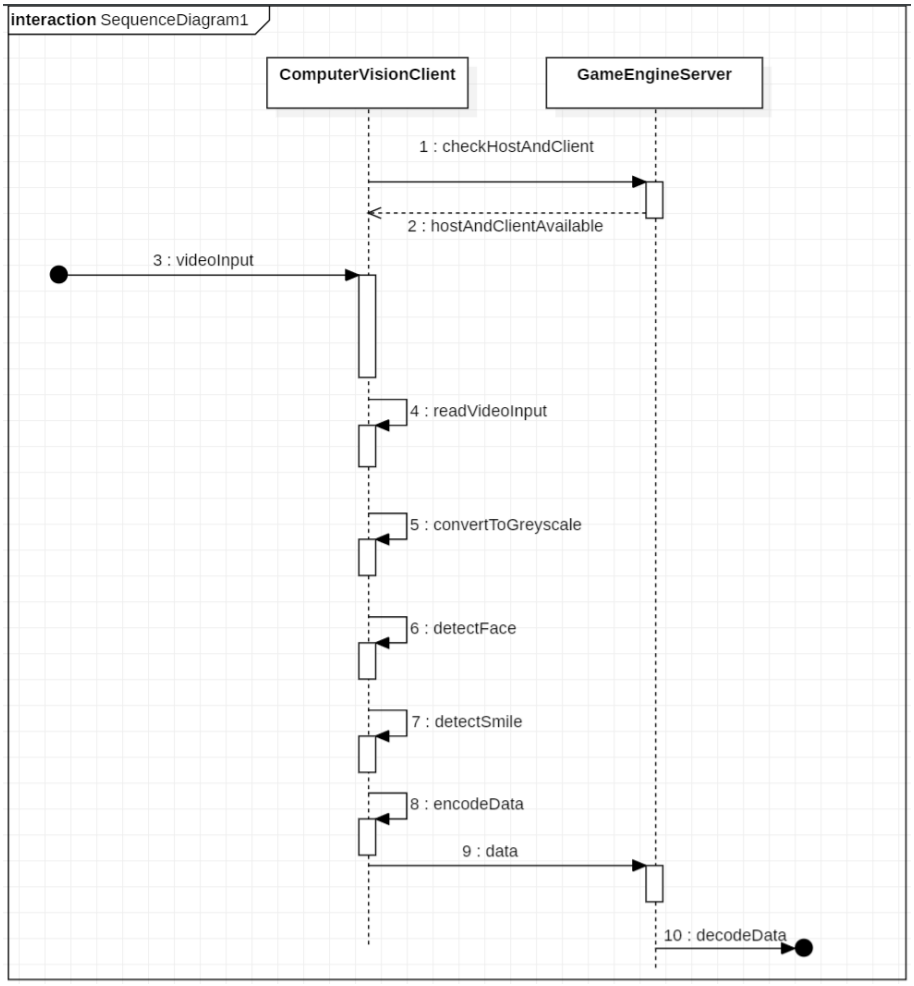

Figure 45: Sequence diagram illustrating the chronological flow of the software. (made by authors)

The sequence diagram shows the chronological flow of the program. As shown, the connection between the client and server is established first. Once this is completed, the program takes a live video input from the camera, reads it, converts it to greyscale. Then, the program will detect a face, if any, in the video. If there is a face, it will check for a smile. This data will then be encoded into byte form and then sent to the "GameEngineServer". The "GameEngineServer" will then decode the data and trigger an event depending on the decoded data.

11.3 Client Implementation

Since the data was going to be sent to a game engine server (as seen in chapter 7). A client in Python needs to be implemented. As most of the implementation in Python is completed, this section will only cover how the client was implemented on top of the existing code, which was implemented in production 3 (10). At the beginning of the code, the socket library is imported to the script, this contains many functions relating to socket programming. After this, the host and port of the socket are declared and initialised. For this production, a local connection is suitable as the environment and the Python client can run simultaneously on the same computer. Therefore, the local host is used.

```
import cv2
import numpy as np
import socket

host, port = "127.0.0.1", 25001
data = "true"

lowerParameter = 2.50
higherParameter = 20

s = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
```

Figure 46: The import of the socket library, initialisation of the host and port variables and the creation of the socket object.

Furthermore, a socket object "s" is created from the socket class which takes two arguments. The first argument is what type of address and protocol should be used, 'socket.AF_INET' is assigned here. The second is declaring that the socket should be a user datagram protocol connection (UDP). Then, the classifier objects are created. The object "s" calls the connect function from the socket library, this function takes the host and port variables as arguments.

```
#the database of faces
faceDetect = cv2.CascadeClassifier('haarcascade/haarcascade_frontalface_default.xml')
smileDetect = cv2.CascadeClassifier('haarcascade/haarcascade_smile.xml')

#a variable for the webcam
cam = cv2.VideoCapture(1)

s.connect((host, port))
```

Figure 47: The host and port variables being passed into the socket object.

Finally, in order to send the data to the server, the function sendall() is called from the socket object, a function from the socket library. This function will encode the data and send it to the local server. The string data is encoded into UTF-8. This is a standard encoding for transmitting messages between a server and clients.

```
cv2.imshow("Face",img)
print(lowerParameter)
if  countFace > 0:
    data = "Face without smile"
    if countSmile > 0:
        data = "Face with smile"
else :
    data = "No face detected"
s.sendall(data.encode("utf-8"))
```

Figure 48: The String variable is sent to the the server after being encoded to UTF-8.

11.4 Game Engine Design

In order to provide proof of concept that a narrative can be adapted to a user's smile in realtime, a virtual 3D environment will be created. This will also provide context to the test subjects and therefore could give more accurate feedback. Through brainstorming possible concepts, the most feasible approach was decided to be a simple weather system, that changes according to the input. When the viewer smiles the sun shines, birds sing and a relaxing soundtrack is played. When the viewer is not smiling the environment becomes overcast, dimly lit, stormy and the soundtrack is switched to an ominous genre. Although this a simple system, through the process of adapting a few variables in realtime to the viewer's smile, trying to prove that the narrative of a visual experience can be adapted on a much larger scale.

The weather system will be relatively simple. It will consist of a pre-built environment taken from a public asset libraries [50] & [51]. There will be a directional light that will act as the sun, a point light will imitate lightning and a rain particle system. Two custom events will be created that control the transition between the two environment "moods". These events will change the variables of the objects. For example, the intensity of the directional light. One of these events will then be triggered depending on the information coming from the client (Whether the viewer is smiling or not).

11.5 Implementation in Unreal

This section will explain how the virtual environment was implemented in a way that made it possible to both adapt it in realtime and to communicate with the python client that will be run simultaneously.

As mentioned in (11.4), it was decided to create a small environment in a game engine. Unreal Engine and Unity were both considered to be used as game engine. However, due to the group's familiarity's with working with weather in Unreal Engine, it was decided to implement the environment in this engine.

Generally transmission control protocol(TCP) servers are used to facilitate communication on the internet[52]. However, this type of protocol includes a lot of error checking that can greatly increase latency times. In the case of creating an adaptive *realtime* narrative, this may not be the best option. Further research found that UDP servers do not include a lot of this error checking and are frequently used in live broadcasts due to this. For example, if a TCP server momentarily loses connection, it will have a backlog of data packets to send. This means latency is introduced to the stream of data [52]. If a UDP server momentarily loses connection, it will "forget" about the lost packets and immediately resume sending live data packets, this keeps latency to a minimum. For this project, latency needs to be kept to a minimum, this is why a UDP protocol is the best option [52].

As mentioned in (10.3.1) A client socket has already been implemented in the Python code. A server was needed in Unreal Engine to receive the information from client. With the knowledge of how socket programming works, a UDP server plug-in for Unreal Engine was found that supported string communication and fulfilled all our requirements for the server [53].

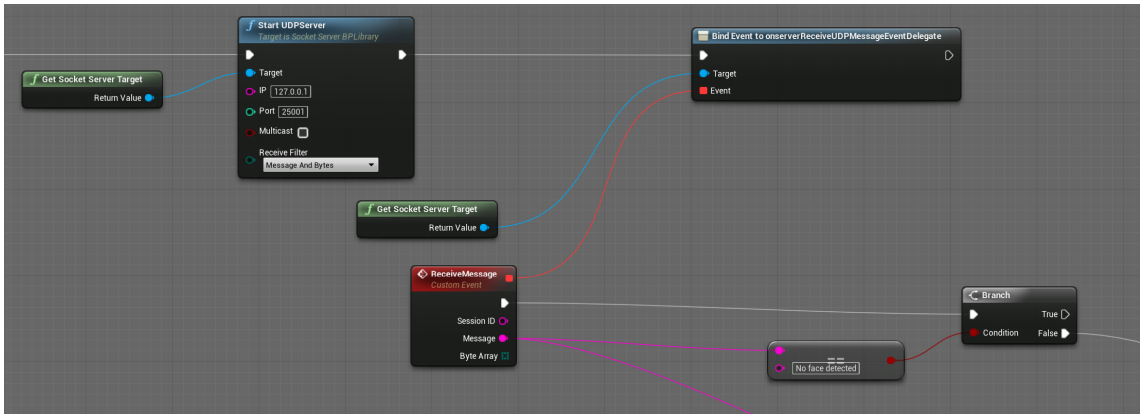

Figure 49: Implementation of the UDP Server plug in

As seen in (figure 49), the server is created and the IP string variable is set to the local host address. The port is set to our chosen integer value that matches the Python client port. The server is then bound to the 'RecieveMessage' event, which returns the message from the client in string form.

The message, as mentioned in (10.3.1) can be one of three strings. These are:

- "No face detected"
- "Face detected with smile"
- "Face detected without smile"

Each time a new packet is received from the Python client, the 'ReceiveMessage' event is triggered. Once started, it goes through two if-statements that check what the string message is. If the message is "Face without smile" then the environment needs to become "sad" (figure 53). Therefore, this message triggers a custom event named "L>D", which first checks if it is "Dark", if it is not Dark, a timeline (which changes the values of a chosen variable type over a given time) updates custom float variables over one second, as seen in figure 50. These variables are then input as parameters into functions that update the scene in realtime, as seen in figure 50.

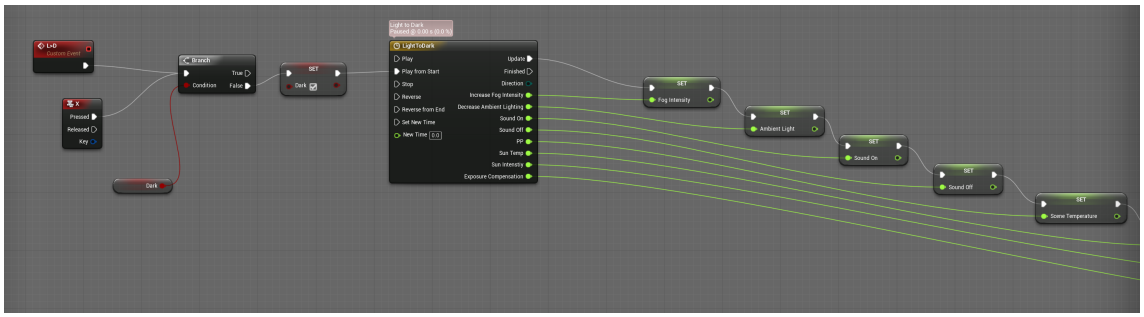

Figure 50: A timeline function that updates certain variables in the scene over a given time.

This is done, for a variety of elements from the scene. Such as

- Fog Density

- Ambient Lighting
- Volume Increase
- Volume Decrease
- Exposure Compensation
- Temperature (colour) of the sun
- Sun intensity

After implementing the core functions of the system, it was found that the UDP server was receiving packets from the Python server at a very fast rate, upwards of 50 packets a second. This high amount of packets mean that the smile detection works, but it regularly "flickers" between "Face with smile" and "Face without smile". Which in turn caused the environment to flicker between "happy" and "sad" (figure 52 & figure 53).

To combat this, it was decided that the modal facial expression over a given time should be used as the deciding factor of which event should run. To implement this, all the string messages received, over a 200 millisecond period, were put into an array. After these 200 milliseconds were over, the contents of the array were evaluated and the string that occupied the most index values was used to determine which event to run. After implementing this, the flickering was no longer a problem.

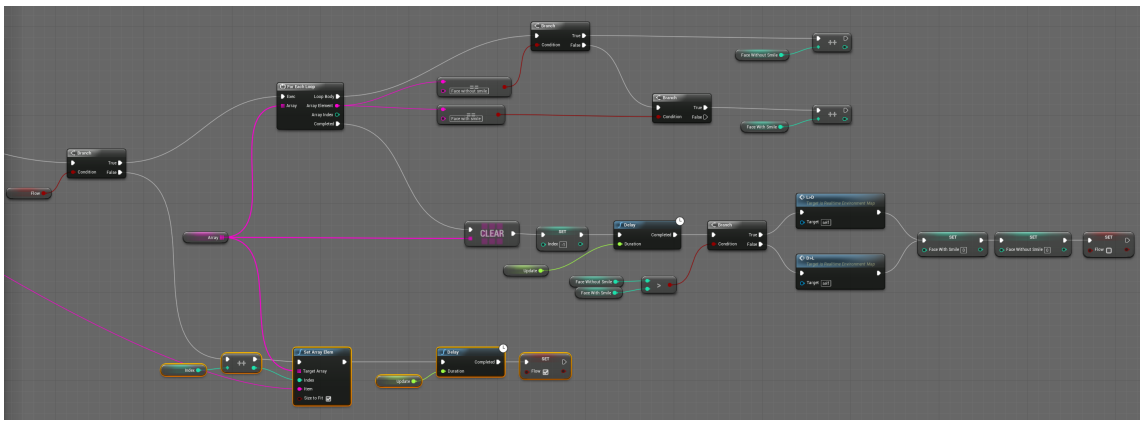

Figure 51: How the flickering problem was reconciled

Finally some extra effects were added to bring the environment to life. Such as lightning, a rain particle system and some sound effects. The resulting environment can be seen below.

Figure 52: The happy environment (if a smile is detected).

Figure 53: The sad environment (if a smile is not detected).

11.6 Final Test

The procedure of the final test will be outlined and explained in this section.

To start of the testing, the group was missing a proper camera, so instead the build in webcam on a laptop computer was used. After the first 13 participants had done the test a higher quality camera was used. The camera was placed on a tripod and connected to a laptop underneath the camera. The laptop was running the OpenCV client side window and the game engine, while a TV

monitor was showing the Unreal Engine environment. Test participants were selected at random as people were moving around the area of the event 6.2. The goal is to have as many participants as possible try the program and have them complete a questionnaire afterwards.

The testing would take place at ViZARTS summit. ViZARTS summit is a reoccurring event which is a place which discusses "Crafting the Future of Adaptive Real-Time Storytelling" by doing "VIsualiZations and Adaptive Real-Time Storytelling". At the event the group was given a demo booth where they could set up and test their application. When the participants of ViZARTS, went by the demo booth and showed interest in the application, a member of the group would tell them about the product. After explaining the product, the group member would ask the person if they would be willing to participate in the test. If they wanted to test the product, they were asked to fill out a consent form as well as a short questionnaire. When they had answered the first part of the questionnaire, they were ask to move over to the application testing setup. A group member would then tell the participant to smile and adjust the program to respond to their smile. Once a participant was ready to test, the participant was given a headset so that they could hear the sounds from the Unreal Engine environment. After the participant had tried the product for a few minutes, they were asked to take the headset off and answer the second part of the questionnaire.

11.7 Production Reflection

Overall, this production was a success. We were able to complete the flowchart by designing and implementing the client-server relationship between Python, the Unreal Engine and the environment within Unreal Engine. The results of this production's test will be presented and evaluated in full in the following chapter (12).

12 Evaluation

This chapter describes, evaluates and analyses the results of the final test and expert interview.

12.1 Final Test

The main motivation for testing the prototype was to answer the final problem statement, by figuring out if the participants thought that an adaptive environment had responded to their smile. The final test had 34 participants which were chosen through convenience sampling 6.2. The first 13 participants were testing the prototype whilst it was using a laptop webcam. The rest of the participants used a high quality webcam with a higher frame rate and resolution.

12.2 Results of Final Test

This section of results will be split into two parts and therefore separate quantitative answers from qualitative answers. The section will show the results of the test as well as graphical representations of the quantitative answers. To see all of the results, see the appendix (16).

12.2.1 Quantitative questions

In total 34 people answered the questionnaire and a total of 36 tried our product. The two people that did not answer the questionnaire were the expert we interviewed and a representative from Unity. Out of the 34 that did answer the questionnaire, 12 were women, 21 were men and 1 preferred not to say. The boxplot below shows the ages of the participants (figure 54).

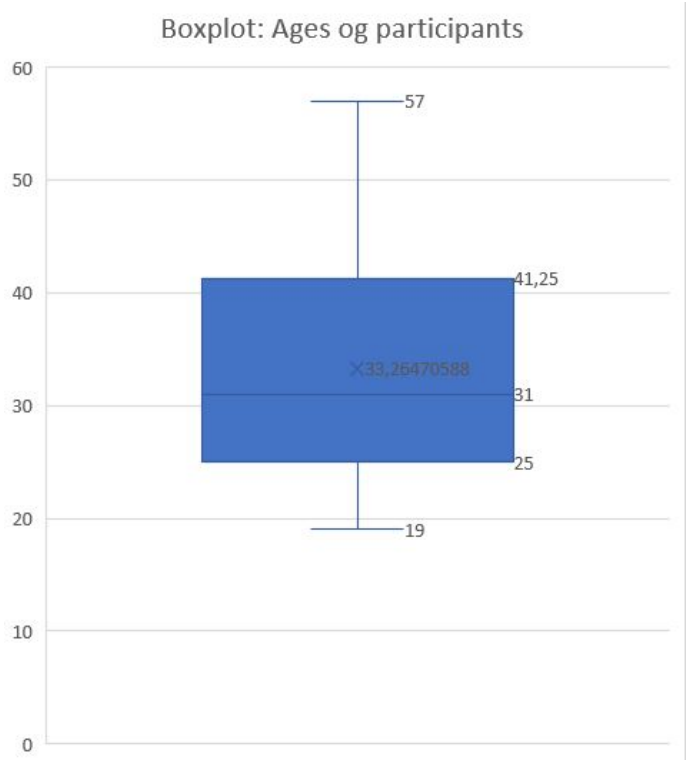

Figure 54: This is a box plot that shows the ages of the participants.

The second part of the demographic questions, most of the participants, a total of 18, do not watch regular TV in their free time, or at least less than once a week. While the other 16 participants were equally divided amongst the other answer options (figure 55).

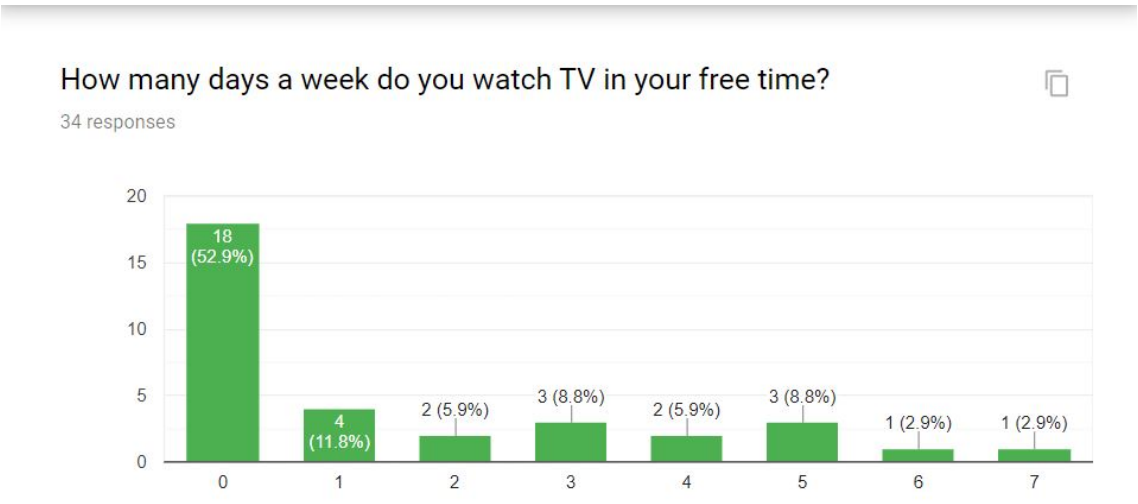

Figure 55: This is a graph that shows how many days a week the participants watch TV.

When the participants were asked "How many days a week do you watch movies and/or series?",

only 2 participants answered that they did not watch movies or series in their free time. However, a total of 18 participants gave the answer of more than 5 days a week, the rest of the answers can be seen below (figure 56).

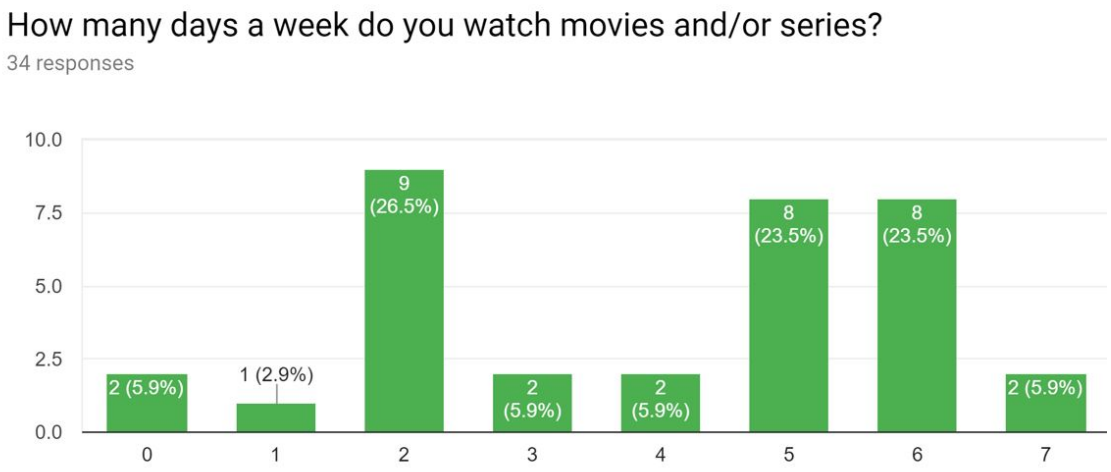

Figure 56: This is a graph that shows how many days a week the participants watch movies and series.

The third media outlet related question was "How many days a week do you play video games?", and here 11 participants said 0 days a week or less then once a week, the rest of the results can be seen below (figure 57).

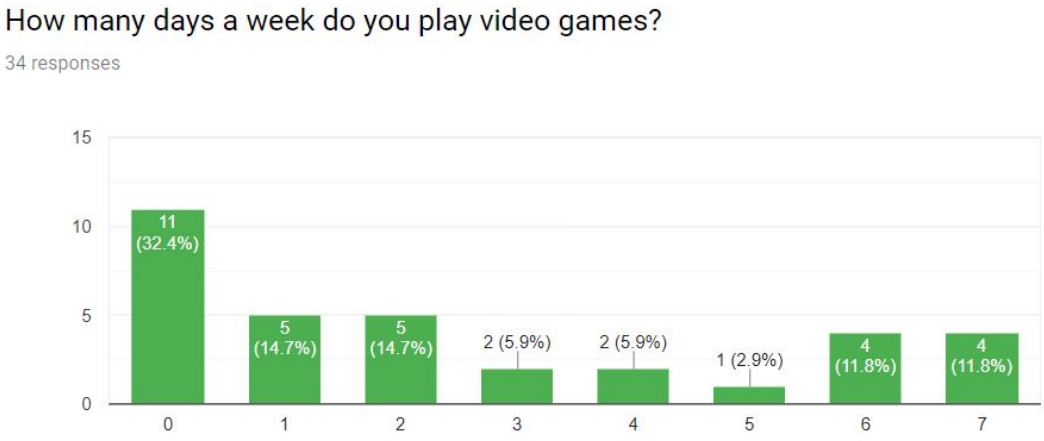

Figure 57: This is a graph that shows how many days a week the participants play video games.

Overall, these questions were being asked to see and give insight into what form of media would make the most sense to work with, for future work and research. According to the findings, most

people would be reached through movies and series. Another question was asked, "What genres do you like?". This question was only asked to give participants some standardised questions that would ease them into the rest of the test.

Moving to the "Final Questions" section of the questionnaire. In this section, the quantitative questions and answers will be looked at. The first question was "On scale of 1 to 10, how responsive did you feel the weather system was to your facial expression?" and here the participants leaned towards a positive response, with a majority of answers placed on or higher than 7. The lowest answers were between 3 and 5 on the 10 step scale. This could possibly be because for some of the participants the program was especially sensitive to their beards and/or lips, which made it difficult for the program to accurately track the features (figure 58).

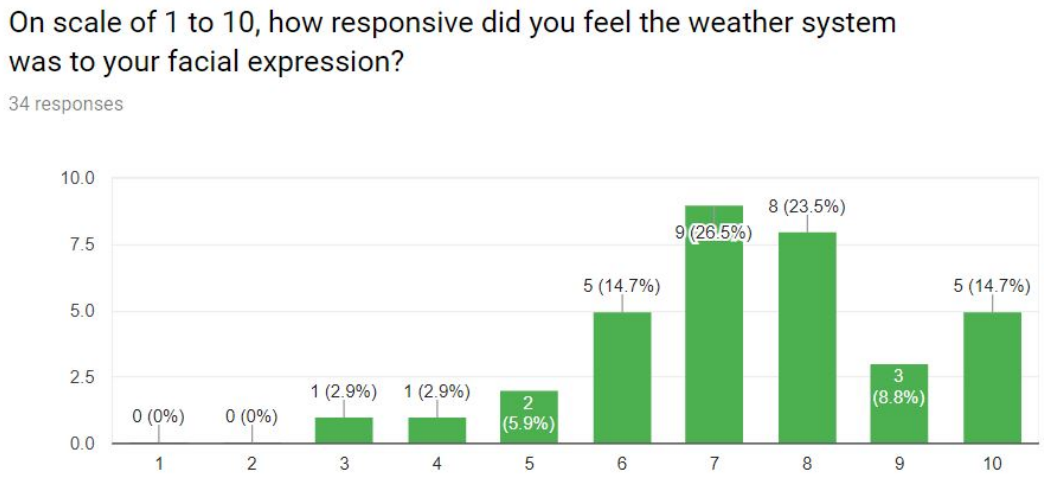

Figure 58: This is a graph that shows how responsive the participants felt the system was to their facial expressions.

When taking a look at this data, the overall impression is that the product was successful at responding to the change of facial expressions from the participants and that they felt that the product was able to properly detect their facial expression.

For the second question, the participants were asked "How smooth did you feel the transition was in the scene?". Like the previous question, the participants seemed mostly positive with 67.8% of participants rating the smoothness at a 7 or above, indicating a relative success in regards to smoothness of the transition in the environment (figure 59).

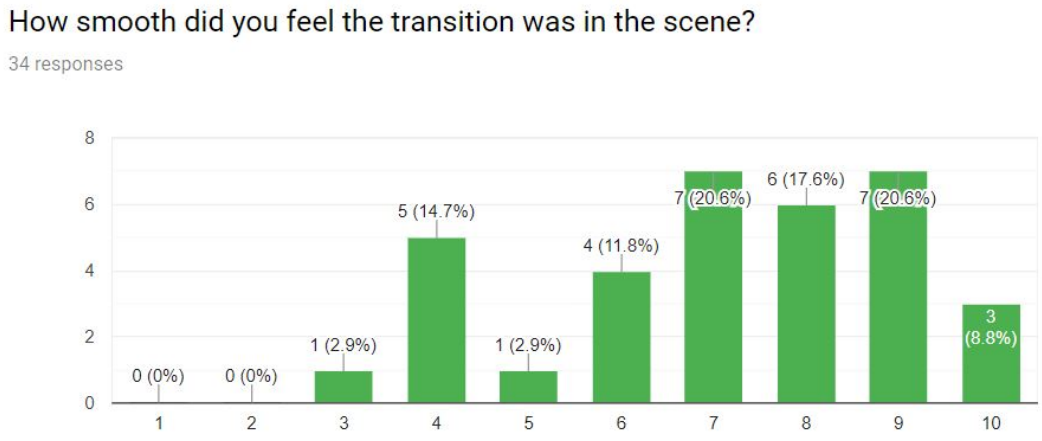

Figure 59: This is a graph that shows how smooth the participants felt the transition was in the scene.

Another question the participants were asked was "How would you rate the experience on a scale from 1 to 10?". Here the question was intended to get feedback on how the overall experience was. 1 participant rated the experience a 3 which was the lowest score and an overall outlier in regards to this question. Other than that there is a large amount of participants that rated the experience above average. 14 of participants answered 7, which goes to show that most participant did not rate the experience to be perfect. Finally, 7 participants rated the experience a 10, which possibly shows that the participants were interested and intrigued by the experience and therefore rated it highly (figure 60).

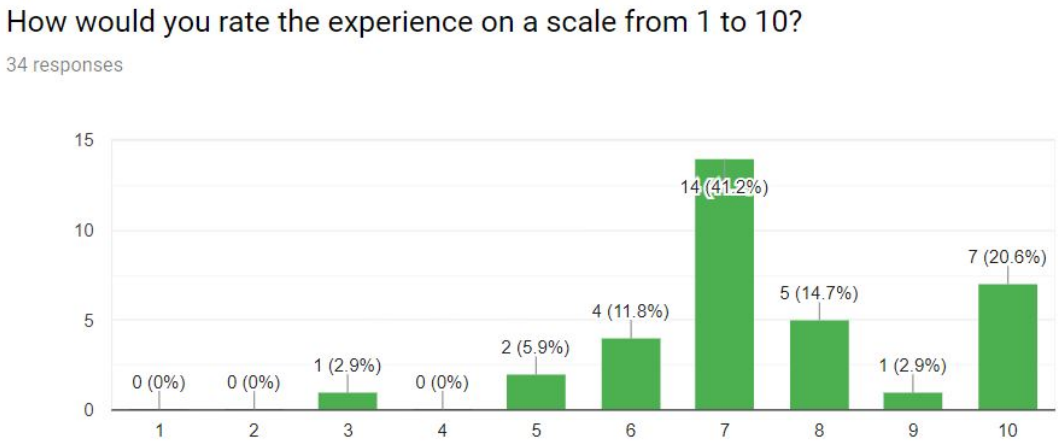

Figure 60: This is a graph that shows how the participants rated the overall experience of the test.

The final question was "How intuitive did you feel the program was to use?" and here, once again, most participants felt that the program was generally intuitive, but some verbally commented that

if they had no introduction they would not know that they had to smile to begin with. Most participants rated the intuitiveness above a 6 and only 5 participants rated it lower than that, with 4 being lowest. This is an overall positive result and in general most people rated it as at least slightly above average in regards to intuitiveness. This is possibly because it is a simple, single-expression detecting prototype (figure 61).

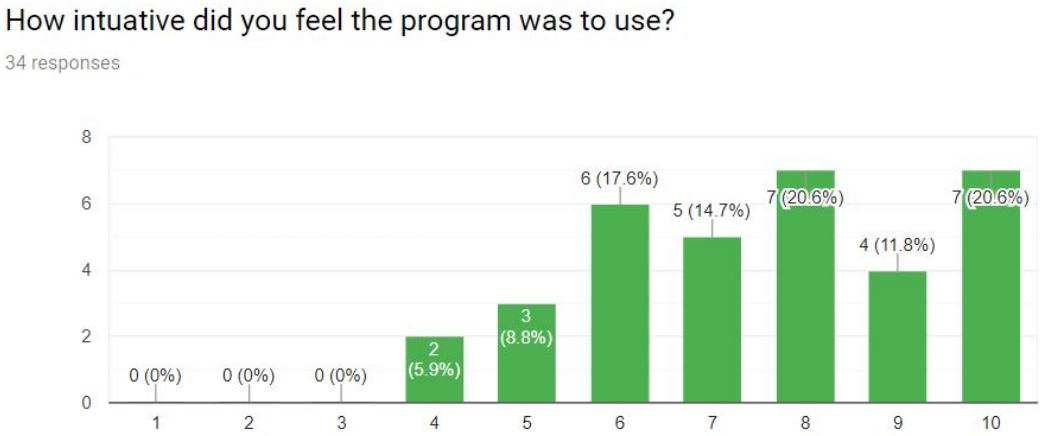

Figure 61: This graph shows how the participants rated the intuitiveness of the program.

12.2.2 Qualitative questions

For the qualitative responses, the first question asked was "In your own words what did you just experience?". The participants were mostly positive about the experience and had given short answers, describing what they experienced. Some of the responses were also single words or similarly short answers. Most of the participants mentioned keywords such as "environment", "adaptive", "facial expressions", "responsive" and "smiling". Some participants also mentioned "weather system" and "narrative".

The second question was "What changed in the scene?". Here participants mentioned "sound", "visuals" and specifically "the weather". Which is also the overall change in the environment that was implemented. A few participants mentioned the lighting change in the scene. To sum up this question the participants were asked "If you noticed changes, did you feel in control of the change?" and all of the participants answered yes, with a few giving extra comments.

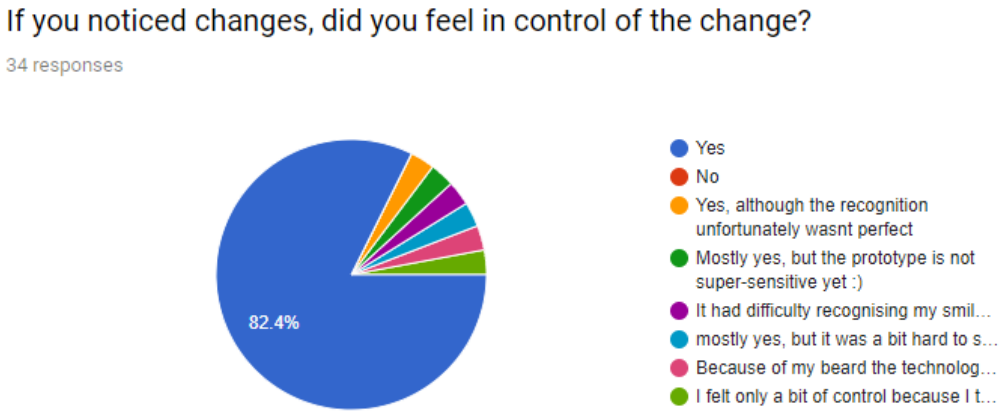

Figure 62: This is a pie-chart that shows how the participants answered the question "If you noticed changes, did you feel in control of the change?".

Taking a closer look at the answers the participants gave to the question regarding having control of the changes, one participant said "Yes, although the recognition unfortunately wasn't perfect". Another said "Mostly yes, but the prototype is not super-sensitive yet :)" and a third said "It had difficulty recognising my smile (maybe include eyes)" (figure 62). The last three responses gave slightly longer answers which can be found in the appendix (16).

For the next two qualitative questions, the participants were asked about the data and privacy in regards to the program. The first question was "Would you feel comfortable being recorded while watching a movie at home, knowing that the video will never be seen by a human being? (The data would be used to optimize the program and only stored short term in a cloud server, and then deleted)" and here the participants were split. 10 participants would not be interested in using the program if the data was handled in such a way, while 4 where on the fence. The rest of the participants either answered "yes" or were otherwise okay with it (16).

For the second version of the question "Would you feel comfortable being recorded if the data was never stored and only analysed in the running live video feed? (not stored or sent anywhere, not even to a local cloud)" More of the participants would feel comfortable using this kind of technology in their own home. While 6 participants would still not use it in their own home, the participants on the fence had decreased to two participants, which leaned slightly more to the side of being okay with it (16).

The next two qualitative questions are also related to each other and the first of the two questions was "Did you try any other facial expressions other than smile/frown?". Here the participants were split in half, as one half had tried other facial expression while the other half had not. The participants that had answered "Yes" were asked to write down which expressions. They answered "sad", "mad", "fear", "excitement", "surprise" and "angry" among a list of other expressions. Some participants also answered that they had tried "Open mouth", "showing teeth" and "neutral face" (16).

For the final question, the participants of the test were asked "For the final question, imagine this

software was implemented for a TV-show such as "Game of Thrones" every person would have a unique version of the episode. The next day you discuss the differences in the episode with your friends. What are your thoughts on this concept?", the participants were mainly positive in regards to this idea but a large majority doubted the implementation of such features in large scale productions, such as Game of Thrones.

12.3 Expert interview

For further evaluation of our product, as well as to gain more knowledge about adaptive real time narratives, an interview with an expert was conducted, below is the evaluation of the interview. The transcription of the interview can be found in full length in the appendix folder and the questions asked will be listen below.

First, questions were asked in regards to general real time adaptive narratives, immersion and types of input used for changing these. The last question was withheld for when the expert had tested our product, this was done in order to let the expert get insight on the product before answering the question.

When asked in regards to which input the expert uses for adapting narratives, the expert said that there are many types of input used for adapting narratives. He said that what he is researching is more than people making simple choices, but more on how people react or behave when encountering different things and characters in narratives.

He then continued on to talk about engagement, or as he defines it, an individuals desire to continue an experience. He mentions that this is measurable by asking test participants from a scale from 0 to 10 how much they want to continue the test, which also invites questions like why they would like to continue or why not.

The expert then mentions a master thesis. In this thesis a program was made that tracked faces and then through deep learning, was able to track the emotion of the viewer with up to 90% accuracy. This program could then be used to detect if a person would like to continue any given experience.

The expert then argues, that facial expression might not be the only or most suitable kind of emotion detection and then gives an example, a group of people engaged in an activity would often exhibit more facial expressions than when one person is sitting alone.

The expert points out that he definitely thinks that immersion is broken through explicit interactions, e.g. when viewers are presented with an option such as route A or route B and have to choose. However, the expert then expanded on this point, mentioning that he thinks engagements stays intact, which is also why the expert chose measuring experiences based on engagement via the desire to continue.

The expert explains that through his own research in continuation desire, people feel compelled to explore branches in narrative and as a result of this engagement is increased. However, if facial expression replaced the explicit interaction in such movies, people will not have the ability to explore the branches in the narrative.

He then explains that there is no specific target audience for adaptive narratives, as it depends on what media and medium is used, and that it can be beneficial for all ages. He also points out that a benefit of these unconscious choices is that kids and elderly people could make use of the technology without having any knowledge about how to use a computer.

When asked about the future of movies and television he describes that the amount of people

playing video games is increasing and that we will see a new breed of experiences, *"which is not interactive storytelling go left or right or choose the man or the woman or whatever but more - kind of in a dialog with the user in a way."*

He then ends with testing our product, saying that he enjoyed it and told us that, yes, it definitely could be used as a proof of concept.

The questions asked in the interview are below:

- What kind input do you use to alter narratives in realtime at ViZARTS?
- Do you think immersion is broken, to an extent, through explicit interaction?
- Some have said that Bandersnatch's success is due to the fact the viewer's feel in control. Do you think implicit interaction (expressions) would hinder this aspect?
- Do you think people exhibit facial expressions to an extent that can be accurately measured?
- In your research have you identified a target audience for interactive movies?
- How do you think this technology will change the way we watch TV/movies?
- In your opinion, is changing the weather suitable as a proof of concept for changing narratives?

12.3.1 Evaluation summary

Through our questionnaire, it was seen that test participants mainly enjoy the product they tested in several regards, how intuitive it was, how smooth the transition was, how responsive it was and the overall experience of testing it. Through our expert interview, we learned that our product served as a proof of concept and was also given inspiration for future testing methods. He also mentions that there are several ways to adapt a narrative which doesn't necessarily include facial expression detection.

13 Discussion

This section will discuss the results from the final test and the implementation of the final production in relation to the final problem statement.

13.1 Evaluation Discussion

During the final test, the participants were asked to answer some quantifiable questions and some qualitative questions as well as trying the final product (12). The test and the questions were made in such a way that they would help answer our final problem statement: *"How can one, through the utilisation of image processing and facial analysis, detect a users face and smile and use this data to create an adaptive narrative in a virtual realtime environment?"*. Specifically, the final test tried to answer the part of the final problem statement relating to whether or not we were able to create an adaptive narrative using face and smile detection.

During testing it was decided to use another camera instead of the built-in laptop camera. The laptop camera had some issues recognizing faces in the area the group was testing in, mainly due to low light conditions. Therefore, the first 13 test participants might have had a different experience with the program than those who tested with the second camera. However, when analysing the results of the questionnaire, there is no noticeable correlation between which camera was used and which answers were given.

The answers gathered from the quantitative part of the questionnaire pointed towards the fact that people felt the program was responsive, that they had control over the narrative and overall had a good experience testing it. There was however still participants who responded negatively and without interviewing these participants it will be difficult to narrow down why they didn't have the same experience as others (12). In future projects, it might be worth it to have a focus group test the program and then interview them in depth about the same subjects that our questionnaire focused on, to get a deeper understanding of any negative response. However, the positive response means that it would make sense to further research and build on the existing program in order to reach a more complete product.

The answers gathered from the qualitative part of the questionnaire gave a deeper understanding of the participants view on the technology and their willingness to use it. 14 out of the 34 participants said that they wouldn't agree to being recorded in relation to using the technology if the data was sent to a cloud server for optimization purposes. 6 out of 34 felt this way if the data wasn't stored anywhere. These responses could be in line with ongoing discussions about data usage and public awareness of the dangers of corporations selling or misusing their data. However, the majority would still allow this kind of technology. Since the technology is new and slowly emerging, it would also be interesting to compare the rate at which it could be adopted compared to the technology adoption life cycle, those who answered positively in regards to using the technology could then possibly become early adopters [49].

The expert interview shed some overall light on the technological field in which this paper relates to. One of the important points is, that giving people the opportunity to have a narrative adapt, without having to do any explicit action, retains the immersion of the individual (12). In future iterations of the program, it would be interesting to use the continuation desire testing method that the expert mentions, in which a participant is stopped at a given point while testing the program and asked whether or not they would like to continue and why/why not this is.

The fact that the test was done at the ViZARTS summit, that convenience sampling was used and the test was done amongst other groups of people testing other technologies, possibly introduced some bias into the test results (6). To give further credence to the results it would be required

to do other tests in a more controlled environment. It would be possible to develop other virtual environments and introduce and instruct other test participants in the use of facial expressions. Then leave them to test it in a room on their own, instead of having several people standing around watching people try the program. This would also give more privacy to answer truthfully on any questionnaire/interview that would be administered by the group.

13.2 Implementation Discussion

The final implementation was executed in such a way that it would help answer the final problem statement. The implementation aims to answer the technical part of the final problem statement whereas the evaluation tried to answer the narrative part as seen from a user perspective.

One of the main issues, which was also touched upon above, was that the facial recognition worked well in the environment it was initially programmed and tested in. However, when the setup was moved to the testing area, unforeseen issues came to light. The issue was mainly that the facial recognition didn't work as well in the lighting conditions that the test took place in. This could easily have been avoided in future projects by having a more controlled environment to test in, for example, a quiet, well lit room possibly with a flat colour background behind the test participant.

It came to the group's attention that OpenCV can be imported in the C++ programming language which is the native programming language of the Unreal Engine. If this was discovered earlier, the face and smile detection could have been directly implemented in Unreal Engine, eliminating the need for a client-server socket.

In the beginning of the paper, major expressions and microexpressions were explained (3.2). The program created in this paper focused on major expressions, which last longer than microexpressions. This meant that the program did not respond to subtle smiles or expressions. This could easily be the main goal if further development was carried out, because microexpressions, by nature, are uncontrollable by the user and therefore might be more valuable to adapt a narrative around.

Furthermore, the narrative that was decided to change was the weather. It might be arguable that changing the weather is not changing the narrative, as some would argue that more components would have to be involved and changed to portray a narrative. It would be interesting to research narratives and in future iterations, figure out what else needs to be done in order to change the narrative. However, for the purpose of this paper and the test, as a proof of concept it was decided that the weather change would be suitable to represent a narrative change, this was also confirmed by the expert in the expert interview (12).

14 Conclusion

The initial motivation for writing this project was to detect smiles in response to visual or musical stimuli (2). Through exploration of topics related to the initial motivation, it was found that facial analysis through image processing could be used to adapt a narrative based on facial expressions of a user in realtime. With this information in mind, the final problem statement was formulated (3). With a complete analysis exploring the topics of the final problem statement, a series of design requirements for an application was made. The application was developed and implemented using an iterative approach to create the product that was used to test the final problem statement 6.

The final problem statement asks, "how can one", in relation to the creation of a product (see 4). This was answered through the overall implementation which will be outlined here. The face detection and smile detection was created using OpenCV, specifically, using Haar cascades. Furthermore, a client was implemented that sent the data to a server in a game engine that adapted the narrative according to the received data. However, it should be noted, as mentioned in the discussion, that OpenCV could be implemented directly in Unreal Engine which would remove the requirement for a network connection (13.2.

When testing the product, the results showed that the participants felt that the environment was adapting to their facial expression and that the transition in the environment was smooth and responsive. As the change in weather in the environment served as a representation of the change in narrative, it can be said that an adaptive narrative controlled by facial expressions *could* be created. This was solidified in the expert interview, when the expert was asked if he believed the application served as a proof of concept to which he responded *"...as a proof of concept, definitely yes..."* (16).

Overall, we believe the final problem statement, which in full was,

"How can one, through the utilisation of image processing and facial analysis, detect a user's face and smile and use this data to adapt a narrative in a virtual realtime environment?",

was answered to an extent. The project was able to prove that an adaptive narrative controlled by facial expressions is possible. This was seen in the final proof of concept application, subsequent testing and the experts review of the product. However, the product changed an environment depending on whether the viewer was smiling or not. As an environment change, arguably, can not be considered as a narrative change, the product can not be considered as an implementation that fully satisfies the final problem statement. It can, however, serve as a proof of concept.

15 Future work

The analysis at the beginning of this project, presented many different technological concepts that could have been implemented to create the product. Some of these were not considered in the final production of this project, because some of the technologies were still in development or they were simply beyond our capabilities for implementation at the of writing this project. However, many of these concepts are worth exploring for future development and will therefore be presented here.

15.1 Major expressions

As of now, a smile was detected and used to adapt an environment. In future development detection of other major expression would increase the variety of input that a narrative could be adapted to, making for a more detailed and even more personalised experience.

15.2 Microexpressions

As mentioned in the expert interview, the degree to which people exhibit major expressions varies widely from person to person. Therefore, microexpressions are an interesting topic of research to implement in future development. This is because they inherently happen at a faster rate and with little to no conscious action, which could make the experience more personalised (3.2).

15.2.1 Combination of input types

Facial expressions can be used for adapting narratives but as the expert also mentioned in his interview, facial expressions are not the only type of input can be used. The reason for not only using facial expressions as an input, is because these vary from person to person. Input such as eye tracking, pupil dilation, or the posture of the viewers in conjunction with facial expression detection could improve the effectiveness of the adaptive narrative.

15.3 Computational creativity

Computational creativity is an interesting topic that was briefly explored in the analysis(3.7). It relates to state of the art deep learning algorithms that analyse art made by humans and creates new art based on its findings. Such algorithms could be implemented into an adaptive narrative that could govern the progression of the narrative based on a number of variables created by the developers. In further development of this product, implementation of this would eliminate the issue of high production cost and time involved in creating adaptive narratives with multiple storylines.

15.4 Photorealism and 5G

The analysis explored realtime photorealism in order to create experiences that were indistinguishable to reality. Realtime photorealism requires expensive hardware to run at a sufficient frame rate. Therefore, if further iterations build upon the photorealism of a narrative experience, 5G could act as a gateway in which experiences such as adaptive narratives could be streamed over. 5G is capable of streaming large amounts of data, and would remove the requirement of expensive

hardware as everything can be rendered remotely before being streamed to the viewer. This, could not be tested in this project due to 5G not being supported in Copenhagen at the time of writing this report.

15.5 Multiple Viewers

As mentioned in the analysis(3.5), interactive movies have been shown in cinemas for years. These early interactive movies used explicit interaction and voting to control the narrative. This technology could be implemented in a cinema or other environments with multiple viewers. The modal facial expression of the entire audience would then be used as an input to adapt the movie.

References

[1] Nitzel Ben Shaul. *Hyper-Narrative Interactive Cinema: Problems and Solutions.* Rodopi, 2008.

[2] Ju Wendy and Larry Leifer. The design of implicit interactions. http://www-cdr.stanford.edu/~wendyju/publications/Ju-DesignOfImplicitInteractions.pdf, 2006.

[3] R Cowie, E Douglas-Cowie, N Tsapatsoulis, G Votsis, S Kollias, W Fellenz, and J.G Taylor. Emotion recognition in human-computer interaction. https://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.125.8176&rep=rep1&type=pdf, 2001.

[4] sightcorp. Facial expression recognition. https://sightcorp.com/knowledge-base/facial-expression-recognition/, 2019.

[5] Hartmut Koenitz, Gabriele Ferri, Mads Haahr, Digdem Sezen, and Tonguc Sezen. *Interactive Digital Narrative: History, Theory and Practice.* Routledge, 04 2015.

[6] Michael Mateas and Phoebe Sengers. Narrative intelligence. https://www.aaai.org/Papers/Symposia/Fall/1999/FS-99-01/FS99-01-001.pdf, 1999.

[7] Henrik Warnn Jensen and Tomas Akenine-Moller. The race for real-time photorealism: the coevolution of algorithms and hardware is bringing us closer to interactive computer graphics indistinguishable from reality. https://go.galegroup.com/ps/anonymous?id=GALE%7CA229835710&sid=googleScholar&v=2.1&it=r&linkaccess=abs&issn=00030996&p=AONE&sw=w, 2010.

[8] 80LV. Real-time photorealism. https://80.lv/articles/real-time-photorealism/, 2017.

[9] Chris Hoffman. What is 5g, and how fast will it be? https://www.howtogeek.com/340002/what-is-5g-and-how-fast-will-it-be/, 2019.

[10] Megan Gannon. Feeling envious or lustful? brain scans can tell. https://www.livescience.com/37603-brain-scans-can-read-emotions.html, 2013.

[11] amrita.edu. Hormones and chemicals linked with our emotion. https://www.amrita.edu/news/hormones-and-chemicals-linked-our-emotion, 2018.

[12] interaction design.org. Emotional design. https://www.interaction-design.org/literature/topics/emotional-design, 2019.

[13] David Matsumoto and Hyi Sung Hwang. Reading facial expressions of emotion. https://www.apa.org/science/about/psa/2011/05/facial-expressions, 2011.

[14] Ferris Jabr. The evolution of emotion: Charles darwin's little-known psychology experiment. https://blogs.scientificamerican.com/observations/the-evolution-of-emotion-charles-darwins-little-known-psychology-experiment/, 2010.

[15] Kendra Cherry. The 6 types of basic emotions and their effect on human behavior. https://www.verywellmind.com/an-overview-of-the-types-of-emotions-4163976, 2019.

[16] Abrams Richard A. Yantis, Steven. Sensation and perception, 2017.

[17] Nelson R Kolb H, Fernandez E. Webvision: The organization of the retina and visual system [internet]. https://www.ncbi.nlm.nih.gov/books/NBK11530/, 1995.

[18] Mallory Ferland. Comparison of the human eye to a camera. https://sciencing.com/comparison-human-eye-camera-6305474.html, 10. April 2018.

[19] Kyle Schurman. How does a digital camera work? https://www.gadgetreview.com/how-does-a-digital-camera-work, 12 September 2019.

[20] Chris Woodford. Digital cameras. https://www.explainthatstuff.com/digitalcameras.html, 2006.

[21] B. Moeslund, Thomas. *Introduction to Video and Image Processing (Building Real Systems and Applications*. Springer London Dordrecht Heidelberg New York, 2012.

[22] Paul Viola, Michael Jones, et al. Rapid object detection using a boosted cascade of simple features. *CVPR (1)*, 1(511-518):3, 2001.

[23] opencv.org. Cascade classifier. https://docs.opencv.org/3.4/db/d28/tutorial_cascade_classifier.html, 11 december 2019.

[24] Wikipedia. Summed-area table. https://en.wikipedia.org/wiki/Summed-area_table, 11 december 2019.

[25] Wikipedia. Adaboost. https://en.wikipedia.org/wiki/AdaBoost, 11 december 2019.

[26] Erik Devaney. The psychology of personalization: Why we crave customized experiences. https://blog.hubspot.com/marketing/psychology-personalization, 2017.

[27] Miriam Gillinson. Interactive theatre: five rules of play from an audience perspective. https://www.theguardian.com/culture-professionals-network/culture-professionals-blog/2013/jan/17/interactive-theatre-rules-audience-perspective, Jan 17 2013.

[28] ChooseYourOwnAdventure. History of cyoa. https://www.cyoa.com/pages/history-of-cyoa, Oct 28 2019.

[29] Naughty Dog. Crash bandicoot. [CD-ROM], 1996.

[30] Laura Parker. Once upon a time: Narrative in video gamess. https://www.gamespot.com/articles/once-upon-a-time-narrative-in-video-games/1100-6214951/, 2009.

[31] Rockstar Games. Grand theft auto iii. [CD-ROM], 2001.

[32] Visual Concepts Entertainment. Nba 2k series. [CD-ROM], 2019.

[33] Playground Gamesm. Forza horizon 4. [CD-ROM], 2018.

[34] Bethesda Game Studios. Fallout. [CD-ROM], 2015.

[35] Quantic Dream. Heavy rain. [CD-ROM], 2010.

[36] Kevin Veale. "interactive cinema" is an oxymoron, but may not always be. http://gamestudies.org/1201/articles/veale, 17 december 2019.

[37] Quora. What is the relationship between video games and movies? do they share something? https://www.quora.com/What-is-the-relationship-between-video-games-and-movies-Do-they-share-something, 17 december 2019.

[38] Raffi Khatchadourian. The movie with a thousand plotlines. https://www.newyorker.com/magazine/2017/01/30/alternate-endings, 2017.

[39] Byrne Katie. All 10 main endings to black mirror's bandersnatch explained. [https://www.digitalspy.com/tv/a25721836/bandersnatch-endings-black-mirror-explained/], 2019.

[40] Lucas Shaw. Netflix boosts bet on interactive tv with wave of new kids shows. https://www.bloomberg.com/news/articles/2019-08-08/netflix-boosts-bet-on-interactive-tv-with-wave-of-new-kids-shows, 2019.

[41] Julia Alexander. Twitch wants to let its users create a choose-your-own-adventure tv series. https://www.polygon.com/2017/5/5/15563268/twitch-tv-choose-your-own-adventure, 2017.

[42] Computer Hope. What is ray casting? https://www.computerhope.com/jargon/r/ray-casting.htm, 08/02/2019.

[43] Andrew Hayward Bill Thomas. What is ray tracing? the games, the graphics cards and everything else you need to know. https://www.techradar.com/news/ray-tracing, 20. august 2019.

[44] Eustance Huang. 5g could change the video game industry forever. https://www.cnbc.com/2018/04/27/5g-could-change-the-video-game-industry-forever.html, 27. April 2018.

[45] Andy Baryer. Cloud gaming: From 4g to 5g. https://www.futurithmic.com/2019/10/15/cloud-gaming-from-4g-to-5g/, 15. October 2019.

[46] Angelo Ilumba. 5g vs 4g vs 3g: Comparing generations of mobile network technology. https://www.whistleout.com/CellPhones/Guides/5g-vs-4g-vs-3g, 25. March 2019.

[47] Thomas Bjørner. *Qualitative methods for consumer research*. Hans Reitzels, 12. januar 2015.

[48] OpenCV team. about. https://opencv.org/about/, 17 december 2019.

[49] Wikipedia. Technology adoption life cycle. https://en.wikipedia.org/wiki/Technology_adoption_life_cycle, 15 december 2019.

[50] NatureManufacture. Environment set. https://www.unrealengine.com/marketplace/en-US/product/environment-set, 07 December 2017.

[51] UNEASY. Good sky. https://www.unrealengine.com/marketplace/en-US/product/good-sky, 04 March 2018.

[52] Chris Hoffman. What's the difference between tcp and udp? https://www.howtogeek.com/190014/htg-explains-what-is-the-difference-between-tcp-and-udp/, 03. July 2017.

[53] Socke. Simple udp tcp socket server. https://www.unrealengine.com/marketplace/en-US/product/simple-udp-tcp-socket-server, 11 January 2018.

16 Appendix

Everything that has been referenced to the appendix, it can be found in the additional appendix folder, which consist of the questionnaire for the final production, questionnaire for production 3, the expert interview, the avs video, productions from 1, 2 and 3 as well as the final production.

And other knowledge gained through semester courses and research.

16.1 Questionnaire for Final Production

Consent form

This is a consent form.
If you agree to the terms press "Yes" otherwise press "No" and the questionnaire will end.
This is an anonymous questionnaire and will not include your name or contact information.
The questionnaire consist of two phases. Phase one is strictly to put you in a demographic group, the second phase consists of questions related to the product.
If you have any questions during the test please ask one of the members of the group and they will be of assistance.
If you have any inquiries, questions or wish to no longer have your answer be apart of the questionnaire please contact the the group on our group email (k19ml308@create.aau.dk)

* Required

1. **Do you agree with the terms above?** *
   Mark only one oval.

   - ◯ Yes
   - ◯ No *Stop filling out this form.*

Med3 test questionnaire

These questions are related to which demographic group you belong to. Please answer all the questions.

2. **How old are you?** *

   _____

3. **What gender are you?** *
   Mark only one oval.

   - ◯ Male
   - ◯ Female
   - ◯ Other
   - ◯ Prefer not to say

4. **How many days a week do you watch TV in your free time?** *
   Mark only one oval.

	0	1	2	3	4	5	6	7	
Less than once a week	◯	◯	◯	◯	◯	◯	◯	◯	Everyday of the week

5. **How many days a week do you watch movies and/or series?** *
Mark only one oval.

	0	1	2	3	4	5	6	7	
Less than once a week									Everyday of the week

6. **How many days a week do you play video games?** *
Mark only one oval.

	0	1	2	3	4	5	6	7	
Less than once a week									Everyday of the week

7. **What genres do you like?** *
Check all that apply.

- [ ] Horror
- [ ] Comedy
- [ ] Thriller
- [ ] Action
- [ ] Romance
- [ ] Other: _____

Post-test questionnaire

These are a list of questions we would like you to answer after having tried our product. Please be as detailed and descriptive as possible.

8. **In your own words what did you just experience?** *

9. **What changed in the scene?** *

10. **If you noticed changes, did you feel in control of the change?**
Mark only one oval.

- ◯ Yes
- ◯ No
- ◯ Other: _____

Final Questions

11. **On scale of 1 to 10, how responsive did you feel the weather system was to your facial expression?** *
Mark only one oval.

	1	2	3	4	5	6	7	8	9	10	
Not at all resonsive	◯	◯	◯	◯	◯	◯	◯	◯	◯	◯	Very responsive

12. **How smooth did you feel the transition was in the scene?** *
Mark only one oval.

	1	2	3	4	5	6	7	8	9	10	
Not smooth at all	◯	◯	◯	◯	◯	◯	◯	◯	◯	◯	Very smooth

13. **Would you feel comfortable being recorded while watching a movie at home, knowing that the video will never be seen by a human being? (The data would be used to optimize the program and only stored short term in a cloud server, and then deleted)** *

_____

14. **Would you feel comfortable being recorded if the data was never stored and only analysed in the running live video feed? (not stored or sent anywhere, not even to a local cloud)** *

_____

15. **How would you rate the experience on a scale from 1 to 10?** *
Mark only one oval.

	1	2	3	4	5	6	7	8	9	10	
Bad	◯	◯	◯	◯	◯	◯	◯	◯	◯	◯	Good

16. **Did you try any other facial expresions other than smile/frown?** *
Mark only one oval.

- ◯ Yes
- ◯ No

17. **If yes, what expressions?**

18. **How intuative did you feel the program was to use?** *
Mark only one oval.

	1	2	3	4	5	6	7	8	9	10	
Not intuative	◯	◯	◯	◯	◯	◯	◯	◯	◯	◯	Very intuative

19. **For the final question, imagine this software was implemented for a TV-show such as "Game of Thrones" every person would have a unique version of the episode. The next day you discuss the differences in the episode with your friends. What are your thoughts on this concept?** *

Thank you for participating in our test and answering our questionnaire.

We appriciate you :) <3.