Eye Tracking in Virtual Reality: A Proof of Concept for Assessing Autism Spectrum Disorders in Forensic Settings

A thesis submitted to the National University of Ireland, Cork
in partial fulfilment of the requirements for the degree of
Master of Science in Interactive Media
in the Faculty of Science

October 2024

Yannis Peloutier
Department of Computer Science

Contents

List of Tables

List of Figures

List of Algorithms

Abstract

Assessing Autism Spectrum Disorders (ASD) can be conducted using various tests that evaluate an individual's social communication and interaction skills; however, very few of these tests utilise eye tracking in Virtual Reality (VR). David Murphy, a Chartered Forensic and Consultant Clinical Neuropsychologist at Broadmoor Hospital, aims to develop a new tool to assist with ASD assessments in forensic settings, where this task is particularly challenging. This thesis presents the design, implementation, and evaluation of a solution that exploits eye tracking in VR. Using Unreal Engine 5.4 and a Meta Quest Pro, I developed a VR-based replication of the Revised Eyes Test (RMET) featuring MetaHumans—hyper realistic human-like avatars created by Epic Games—as new stimuli. Eye gaze data collected by the headset were processed in Python to generate a report compiling all relevant information to aid David Murphy in ASD assessment. The resulting software solutions serve as a proof of concept for a new tool that could enhance ASD assessment in forensic settings.

Declaration

No portion of the work referred to in this thesis has been submitted in support of an application for another degree or qualification of this or any other university or other institution of learning.

Signed:
Yannis Peloutier

Acknowledgements

Many thanks to my project supervisor, David Murphy, for his guidance and calm during my brainstorming sessions, which helped me stay focused on the task. I am also grateful to Dr. David Murphy from Broadmoor Hospital for his expertise and clear explanations on ASD and theory of mind. I would also like to thank my MScIM classmates for bringing positive energy to our windowless lab this summer.

I want to express my gratitude to my parents for their unwavering support during my studies away from home. Alicia, thank you for always being by my side in my highs and lows.

This project is dedicated to my little brother, Maxime.

Chapter 1

Introduction

1.1 Project presentation

This project aims to develop a Virtual Reality (VR) application as an assessment tool that might assist with the diagnosis of autism in adults.

It is a collaborative work with Dr David Murphy, Chartered Forensic and Consultant Clinical Neuropsychologist from Broadmoor Hospital. This hospital is a high secure psychiatric hospital located in Crowthorne, Berkshire, England and is the oldest of its kind in the country as it was build in 1863. It counts approximately 200 patients, all adult males.

Broadmoor is internationally renowned for its highly specialised care and research work. Dr David Murphy's work is centred around patients with an Autistic Spectrum Disorder (ASD) and their specific needs. In his mission to improve the caring of individuals with an ASD in high secure psychiatric hospital, he's looking for new tools to help with the diagnostic of autism in adults.

In recent years we have seen the development of new tools using VR and eye tracking for diagnostic and therapeutic use for ASD (Kelly et al., 2020,Chen et al., 2021). However these tools are suited for children, in non-forensic settings. Dr David Murphy is looking for an easy-to-use tool that can provide him with important information to help him to diagnose adults who may present with other psychological disorders such as schizophrenia. With that objective in mind I designed and developed a VR application on Unreal Engine 5, using the eye tracking feature from the Meta Quest Pro, a high-end headset from Meta.

In this thesis, after defining what ASD is, I will proceed with a literature review of the existing diagnosis methods of this disorder, from traditional techniques to the more recent ones using VR. Then, this thesis will provide an overview of how I dealt with the problem by presenting the design I came up with and its implementation. The final section of this work sets out the methodology of the research, a detailed evaluation of the implemented solution along with suggestions for potential improvements.

1.2 Autism Spectrum Disorder, Asperger syndrome and why the latter is no longer used

Autism Spectrum Disorder (ASD) is defined as follow by the American Psychiatric Association in the DSM-5 :

> "Autism spectrum disorder is characterized by persistent deficits in social communication and social interaction across multiple contexts, including deficits in social reciprocity, nonverbal communicative behaviors used for social interaction, and skills in developing, maintaining, and understanding relationships." *Diagnostic and statistical manual of mental disorders*, 2013

It's also defined by the World Health Organization in the ICD-11 :

> "Autism spectrum disorder is characterised by persistent deficits in the ability to initiate and to sustain reciprocal social interaction and social communication, and by a range of restricted, repetitive, and inflexible patterns of behaviour, interests or activities that are clearly atypical or excessive for the individual's age and sociocultural context. The onset of the disorder occurs during the developmental period, typically in early childhood, but symptoms may not become fully manifest until later, when social demands exceed limited capacities. Deficits are sufficiently severe to cause impairment in personal, family, social, educational, occupational or other important areas of functioning and are usually a pervasive feature of the individual's functioning observable in all settings, although they may vary according to social, educational, or other context. Individuals along the spectrum exhibit a full range of intellectual functioning and language abilities." "ICD-11 for Mortality and Morbidity Statistics", n.d.

Autism is a complex topic which requires prudency and before anything else to remember that this is a spectrum. On this spectrum, there are numerous and very different individuals who express themselves in as many numerous and very different ways, as it was well noted/said by the late Dr Oliver Sacks:

> "Moreover, there may be a most intricate (and potentially creative) interaction between the autistic traits and the other qualities of the individual. So while a single glance may suffice for a diagnosis, if we hope to understand the autistic individual, nothing less than a total biography will do." Sacks, 2011

During my research on the topic, I came across many terms to define different forms of autism, like Asperger syndrome to describe a form of 'high-functioning' autism. Asperger syndrome is a previously used diagnosis on the autism spectrum (along with 4 other forms of autism defined by DSM-IV). This term was introduced in 1981 by the British psychiatrist Lorna Wing who pioneered the idea that autism should be considered as a spectrum condition (L, 1981). However, she referenced Hans Asperger's work, an Austrian psychiatrist who aided the Nazis in euthanasia of children deemed mentally incompetent (Czech, 2018). Despite its

controversial origin, some people still refer to this terminology to describe themselves because their diagnosis forms an important part of their identity.

In 2013, this diagnostic alongside similar ones were folded into the broader Autism Spectrum Disorder (ASD) in the DSM-5, followed by the ICD-11 in 2019. In this thesis I will mention and cite outdated terms like Asperger syndrome, however they all refer to ASD.

Chapter 2

Analysis

This chapter will provide an overview of the most common methods used for the assessment of adults with ASD, both within and outside of forensic settings. I will also introduce emerging technology-based tools used in this field of research. Following this review, I will analyse the problem before determining the direction in which the design of my solution should proceed, considering the existing works in the field.

2.1 ASD assessment: the traditional way

Most ASD diagnoses are made during childhood and most of assessment tools are suited for children. However two of the assessments can be retained for adults : The Adult Asperger Assessment (Baron-Cohen et al., 2006) and the ADOS-2 Module 4 (Lord et al., 2016). The Adult Asperger Assessment (AAA) was designed following a set of diagnostic criteria more restrictive than those used in the DSM-IV. Despite the fact it was designed before the publication of DSM-5 and its definition of ASD, the AAA remains a frequently used tool. It is described as follows :

> "Before the clinical interview, patients are asked to complete the AQ (Autism Spectrum Quotient: Baron-Cohen, Wheelwright, Skinner, Martin, & Clubley, 2001) and the EQ (Empathy Quotient: Baron-Cohen & Wheelwright, 2004) as screening questionnaires...The AQ comprises 50 questions, made up of 10 questions assessing 5 different areas... social skill...; attention switching...; attention to detail...; communication...; imagination... The EQ comprises 60 questions, 40 assessing empathy and 20 filler (control) items"Baron-Cohen et al., 2006

AQ and EQ are calculated via a Microsoft Excel document then the clinician proceeds to an interview with the patient and their entourage with the intention to either validate or invalidate the symptoms mentioned in the AQ/EQ. This evaluation allows the specialists to make a diagnostic. The ADOS-2 consists of different tasks like describing pictures, construct something,etc, with a clinician. The ADOS-2 comprises different modules designed for specific age groups and Module 4 is the one developed for adults. Thanks to this module clinician will submit all gathered information to an algorithm that generate 2 values, Social Affect (SA) and

Restricted and Repetitive Behaviour (RRB). The ADOS-2 Module 4 considered as the golden standard to diagnose ASD in adults.

While AAA and ADOS-2 Module 4 are reliable tools for ASD assessment in adult populations, they do present some limitations in forensic settings.Some critiques have been addressed regarding the accuracy of the ADOS-2 Module 4 in the ASD assessment in adults with complex psychiatric conditions such as schizophrenia (Maddox et al., 2017). When the accuracy of the tool is not the main issue,then the problem lies in the environment in which it is used. In a high secure psychiatric hospital like Broadmoor, it is complicated to obtain a diagnosis with the AAA (Murphy and Radley, 2023). In other words,while these tools are reliable and easy to use outside of forensic settings , they have shown their limitations in facilities like Broadmoor Hospital. Despite these limitations, David Murphy found another criterion to evaluate: the theory of mind.

2.2 ASD assessment: the theory of mind

> "An individual has a theory of mind if he imputes mental states to himself and others. A system of inferences of this kind is properly viewed as a theory because such states are not directly observable, and the system can be used to make predictions about the behavior of others." Premack and Woodruff, 1978

The capacity to infer the mental states of others (emotions,beliefs,intentions,etc) is referred to as the theory of mind (sometimes referred as ToM in this thesis). Individuals with autism have difficulties accurately inferring the mental states of others using visual cues such as body language and facial expressions. This was initially demonstrated by Baron-Cohen, 2000. He came up with some tests for ToM in people with autism. David Murphy works on ToM assessments with forensic patients, using two tests : the Revised Eyes Test (Baron-Cohen et al., 2001) and the Modified Advanced Theory of Mind Test (Frith and Corcoran, 1996).

The Revised Eyes Test is revised version of the Reading the Mind in the Eyes Test published by Simon Baron-Cohen in 1997. It is used in order to evaluate the 'social perceptual' component of the ToM system or in other words, the ability to infer instinctively the mental state of other using visual cues. The test, abbreviated RMET, comprises 36 photographs, each presented with four word choices. For each set of eyes, the patient has to choose the word that best describes the mental state experienced by the person on the picture, what he or she is thinking or feeling.

Another test used by David Murphy is the Modified Advanced Theory of Mind Test, a modified version of the Advanced ToM Test (Happé, 1994). It is employed to evaluate the 'social cognitive' component of the ToM system or in other terms, the ability to infer consciously. The test presents a series of short stories that depict characters in various social situations involving deception, misunderstanding, or other complex mental states. Participants are asked to explain the characters' behaviour and the underlying mental states that drive their actions.

Theory of Mind is a complex ability with multiple dimensions measured in many different ways and which continues to be explored through ongoing research (Rivero-Contreras et al., 2023). The evaluation of this ability is really useful in ASD assessment and and thanks to the

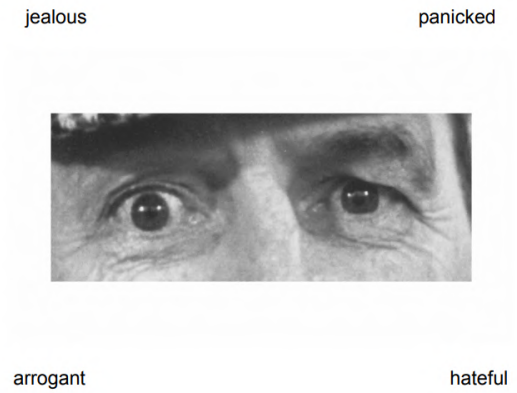


Figure 2.1: An item from the Revised Eyes Test (Baron-Cohen et al., 2001)

The modified advanced theory of mind test

First order "false belief" story and questions

John has five cigarettes left in his packet. He puts his packet on the table and goes out of the room. Meanwhile Janet comes in and takes one of John's cigarettes and leaves the room without John knowing.

ToM question: When John comes back for his cigarettes, how many does he think he has left?

Memory question: How many cigarettes are really left in John's packet?

Second order "deception-prediction" story and questions

Bill has just robbed a bank and is running away from the police when he meets his brother Bob. Bill says to Bob don't let the police find me, then he runs off and hides in the churchyard. The police have looked everywhere for Bill except the churchyard and the park. When they come across Bob they were going to ask him where is Bill? Is he in the park or the churchyard? But the police recognise Bob and they realise that he will try to save his brother. They expect him to lie and so wherever he tells them, they will go and look in the other place. But Bob, who is very clever, knows that the police don't trust him.

ToM question: Where will Bob tell the police to look for Bill, in the churchyard or the park? Why?

Memory question: Where is Bill really hiding?

Figure 2.2: An item from the Modified Advanced ToM Test used by David Murphy (Murphy, 2006)

advent of new emerging technologies in this field, it is now possible to access new metrics, thus assisting experts in giving a diagnostic.

2.3 ASD assessment: eye tracking and Virtual Reality

The "eye gaze" refers to the alignment of a person's eyes towards a specific point or area in space. It has been observed that many individuals with ASD experienced difficulties having eye contact with others, despite the lack of consensus on the eye avoidance hypothesis (Stuart et al., 2023). In classic assessments conducted in interview setting like ADOS-2 Module 4 and AAA, there is no measurement of the eye gaze. However the eye avoidance is usually observed in children when they are engaging in social activities (e.g. playing with friends during break in the schoolyard).

Advanced head-mounted displays (HMD) like the Meta Quest Pro or HTC Vive Pro Eye have two incorporated eye tracking sensors that allow to measure the eye gaze. In the last years, there has been a growing desire to develop new assessment tool for ASD that use VR and eye tracking. During my researches, I came across two projects based on VR.

The first one (Chen et al., 2021) is a collection of VR games designed for children (card game, puzzle, catch & throw ball with child-like avatars). Combined with wearable multi-model sensing technology, including electroencephalogram (EEG), eye tracking, heart rate variability (HRV), and breath-sensing strap, it allows to collect different metrics from children interacting in the VR environment. Game performance and physiological signals are processed with machine learning models based on traditional assessment tools like AAA to deliver helpful information for assessment and therapy of ASD.

The second one (Kelly et al., 2020) is also a game-oriented VR tool designed for children, in which they are required to complete a puzzle with a virtual human using only eye gaze in an environment that can be filled with some distracting elements. This task relies on joint attention, the ability to share focus on an object (in this case the puzzle) with another person. In addition to eye tracking, the team behind this project aims to also integrate electroencephalograms (EEG). The project is still in development.

These new tools look promising for assessment and therapy of ASD, however they are designed for children in non forensic settings and rely on other sensing technologies in completion of eye tracking. As of today, there is no ASD assessment tool using only VR and eye tracking for adults in forensic settings.

2.4 Analysis of the problem

In order to develop a new tool that might help with ASD assessment in adults in forensic settings, it is necessary to take into account what has been done already in this field, which I detailed previously.

Firstly, any design based on an interview setup like ADOS-2 Module 4 and AAA, is unsuitable for places like broadmoor Hospital and present some unavoidable limits. This new tool should be based on the existing tools used by David Murphy in his work, with the aim to provide accurate diagnostics and optimise the new perspectives offered by VR and eye tracking technologies. By taking inspiration from what has been done in the field of VR in ASD assessment and after discussing David Murphy's needs, I came up with two solutions.

Chapter 3

Design

After a field review of the subject of ASD assessment and some exchanges with David Murphy regarding the development of a new VR based tool that might facilitate diagnostic procedures for adults in forensics a diagnostic for adults in forensics, I came up with two designs to solve our problem. However, during the design process, I had to take into account some constraints. In this section, I will present those constraints, then provide a detailed account of my proposed designs and finally explain how and why one of these two has been selected to be the implemented design.

3.1 Constraints

The first constraint on this project is the hardware. David Murphy has a Meta Quest Pro, a XR head-mounted display from Meta in 2022 ("Meta Quest Pro", n.d.) that includes eye tracking technology. XR stands for eXtended Reality, a term that encompasses Virtual Reality, Mixed Reality and Augmented Reality. The tool that I want to develop will be a VR experiment. The HMD comes with two controllers, one for each hand, used to navigate in and interact with the VR environment.

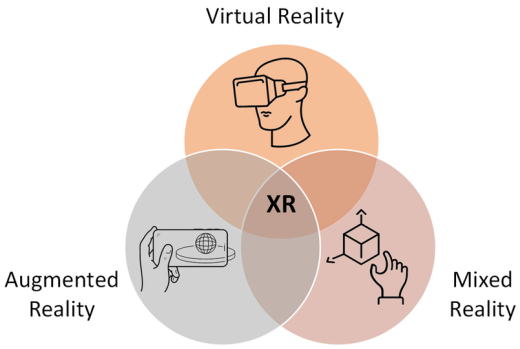


Figure 3.1: Extended Reality technologies (Janiszewski et al., 2021)

Another constraint to this project is the budget as I do not have any allocated funds to it. I am also limited in time, I have 4 months from the beginning of my researches and the thesis submission. An implicit constraint is that I am developing this tool on my own with no prior experience in VR software development. With those constraints in mind, I proposed two solutions to David Murphy.

3.2 First design: eye gaze based game

My first proposed design was to implement the Joint Attention puzzle game that I presented earlier, from a team of researchers from Aston University in UK (Kelly et al., 2020). Due to material constraints, I can't get an EEG, however I can do a relevant visualisation of eye tracking data to help David Murphy in his task to diagnose ASD.

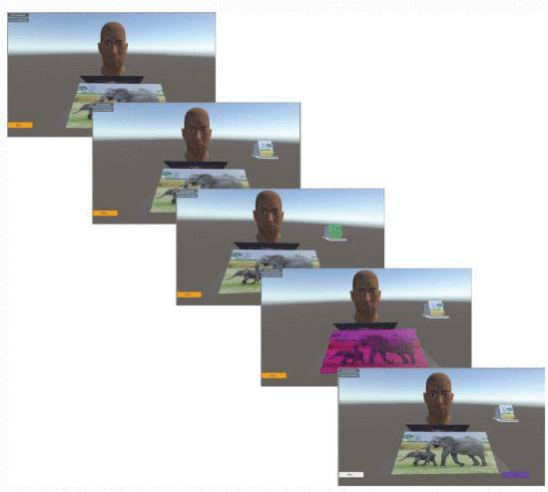


Figure 3.2: JA puzzle game implemented in Unity for HTC Vive Pro Eye (Kelly et al., 2020)

The strength of this design lies in the fully eye gaze based game that allows the participant to fully engage with the experience. It does not require any external intervention and can be easily used in Broadmoor's settings. The use of a human-like avatar to interact with the participant is also really interesting when evaluating social communication skills. On the practical aspect, it is easier to implement this technical solution because there is already existing work that would need some tweaks to work on a Meta Quest Pro.

However, this game was designed to study Joint Attention, a criteria that David Murphy doesn't use in his work to assess ASD. Moreover, it was designed for children in non forensic settings, which represent a completely different target user than the one I seek to investigate on. Since I am not an expert in ASD, I decided to come up with another design but this time, based on tools used by David Murphy.

3.3 Second design: RMET in VR

As presented in section 2.2, David Murphy uses two tests for ASD assessment : the Revised Eyes Test (Baron-Cohen et al., 2001) and the Modified Advanced Theory of Mind Test (Frith and Corcoran, 1996). Both tests made their proof in the field and are suited for adults in forensic settings. The Modified Advanced Theory of Mind Test is of limited interest to be implemented in VR as it consists to listen to little stories and answer questions, tasks that do not involve eye gaze. However, the Revised Eyes Test, abbreviated RMET (Reading the Mind in the Eyes Test) from now on, is interesting because it uses visual stimuli, pictures of pair of eyes from movies, to evaluate ToM abilities. Another interesting aspect is that the presentation of stimuli to the participant, done by the person conducing the RMET, is an easy task that can be automatised.

A basic design of this test in VR would be to put the participant in the VR environment and present him the same stimuli (i.e. pair of eyes pictures) while tracking his eye gaze. Monitoring this data allows the assessor either to identify where the patient is looking at and where he finds social cues to guess which emotion is expressed by the stimulus, or if he is avoiding eye contact with it.

Although this design uses VR, it can be reproduced with a classic flat screen to display stimuli and an eye tracking device. A relevant choice is to replace all pictures that are in 2D with 3D human-like avatars. Avatars like this have been used in the Joint Attention puzzle game presented before and a headset like the Meta Quest Pro is powerful enough to render and display high quality graphics. Replacing stimuli also offers the possibility to correct a default of the RMET: the lack of diversity in ethnicity, age and gender in representation because pictures are limited to adult Caucasian males and females.

Changing stimuli has a great impact on the validity of the RMET, however this project aims to create a new tool that *might* help with ASD assessment. It does not have to be viable but at least functional for David Murphy to test it and determine whether it could become a viable tool. Like the first one, this design would also include a relevant visualisation of eye tracking data.

3.4 Chosen design

After submitting my two proposals to David Murphy, we quickly agreed to go for the second design, the RMET in VR, a variation of the RMET making full use of the Meta Quest Pro and its eye tracking technology. Indeed, this design is best suited for high secure psychiatric hospital environment. The design must be minimal for several reasons. The first is to ensure the validity of the experience; the VR environment needs to be as neutral as the original paper-based test. The second is that I am limited in time, and since no other similar application has been developed, I have to write the code from scratch. For these reasons, my design includes only 3 key elements :

- A human-like avatar displaying different emotions as the source of stimuli
- A table with four push buttons, one per answer

- A screen to display the definition of an answer

Even though it has been discussed to reproduce an office with associated ambient sounds for the virtual environment, these ideas were not retained due to reasons I just mentioned. The final design of the VR application is as follows : the participant is facing a human-like avatar that displays different emotions, he is then asked to choose the correct one by pressing the corresponding button on a table between him and the avatar. A screen is on the table to display the definition of each proposed answer. This choice was done because David Murphy explained to me that when he is doing the RMET, the participant may have to look back and forth to the list of definitions. In parallel to the VR experiment, eye gaze data are recorded then processed to create a relevant report for the person announcing the diagnosis.

Now that the most accurate solution has been identified as the RMET in VR, I will go through the implementation process, the different issues I came across and how I solved them or not.

Chapter 4

Implementation

In this chapter I will go through the implementation process of my chosen design and the different issues I faced during the process along with how I solved them or not. The chosen design is a minimalist VR reproduction of the RMET presented in section 2.2 but with new visual stimuli, a human-like avatar displayed different face expressions and emotions. The participant must pick the correct one by pressing the corresponding push button while the system is collecting data about his eye gaze. Data are processed and delivered in a report relevant to the person giving the diagnostic. Before diving in the implementation, I'll do a review of my development environment, the hardware and software I've used, my software development methodology and how I split the task in half.

4.1 Development environment and methodology

As previously mentioned in section 3.1, I have to develop a VR application for the Meta Quest Pro, a high quality XR wireless headset from Meta released in 2022 ("Meta Quest Pro", n.d.) accompanied by two controllers. It has a 1800x1900 resolution per eyes, a maximum display rate of 90Hz, embarks a Snapdragon XR2+ chipset and 12Gbytes of RAM, allowing it to deliver high quality graphics for a total weight of 722 grams. The headset is also equipped with many sensors including one eye tracker per eye. The way eye tracking works is explained in subsection 4.2.3. Meta doesn't provide more information on its hardware however different block diagrams of the headset and controllers are provided in appendix A by Limas Lin on Karl Kuttag's techblog ("Meta Quest Pro (Part 2) – Block Diagrams & Teardown of Headset and Controller", 2023). Controllers also offer haptic feedback for the user.

To develop an application for this headset, I have a Windows 10 computer equipped with a NVIDIA TITAN Xp (GPU released in 2017) , an Intel Core i7-7700K 4.20Ghz (CPU released late 2016) and 16Gbytes of RAM. It's enough to preview VR game with high quality graphics at the frame rate of the Meta Quest Pro in all game engine. Another part of the implementation was done in python with a laptop computer with similar hardware characteristics on a GNU/Linux based distro, nixOs. Code editing was done with Visual Studio Code and version controlling was done with Git (local repo) and Github (distant repo). The methodology

employed for software development was the Waterfall model.

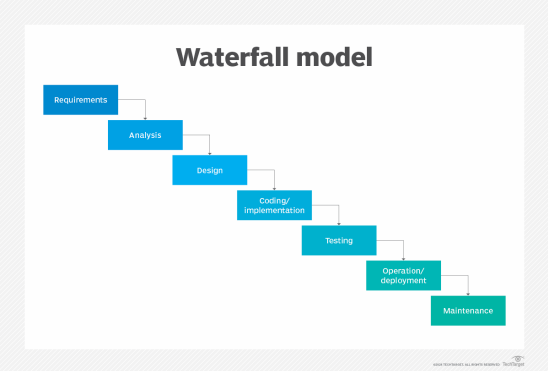


Figure 4.1: Waterfall model ("What is the Waterfall Model?", n.d.)

Due to time and skill restraints, I decided to split the implementation in half. The RMET in VR on one side, considered as the main application, and the eye tracking data processing and visualisation on the other. By doing this, it allows me to transfer a maximum part of the implementation on technologies I am familiar with like python and to leave the remaining work to a new technology, a game engine. I will go over the implementation process of the main application then the data processing part.

4.2 Main application

The main application is a VR game using eye tracking to gather data about the participant's eye gaze. It is a minimalist replica of the RMET with hyper realistic human-like 3D avatars as stimuli instead of pictures of pair of eyes from real actors. The first step has been to pick a game engine to develop my application. Two candidates were considered, Unity (Unity Technologies) and Unreal Engine (Epic Games). Both of them are free to use for small non commercial projects and support eye tracking. Unity is easier to learn and use than Unreal Engine but the latter offers better graphics and a feature really useful for my project, MetaHuman. MetaHumans are hyper realistic human-like avatars, highly customizable and offering many features to animate them. Stimuli being a key component of my implementation, I decided to use Unreal Engine as my game engine.

4.2.1 Unreal Engine

Unreal Engine (UE) is a game engine created by Epic Games in the late 90's. It is a staple in the video game industry with a large community of developers around it that create, share and sell many assets and new functionalities called plugins. For this project I'm using Unreal Engine 5.4, released on April 2024 ("Unreal Engine 5.4 is here! Find out what's new." n.d.). This is the last version of the UE that provides best graphics, performances and support XR features for Meta Quest headsets through the Meta XR plugin. However it also comes with bugs that I came across during this implementation.

There are two ways to develop a game on Unreal Engine, using C++ or the Blueprints Visual Scripting system. It is a node-based visual scripting system that offers the same features as C++ and follows the Object-Oriented Programming (OOP) paradigm ("Blueprints Visual Scripting in Unreal Engine — Unreal Engine 5.4 Documentation — Epic Developer Community", n.d.). With my restraints in mind, I have picked Blueprints because they are way easier to use.

A blueprint can be seen as a class with its properties and methods. In this implementation I deal with two types of blueprints, Pawn and Actor. A Pawn is the physical representation of a player in the game level. An Actor is an object that can be placed in the game level, it can be seen as a container of different types of objects called components. The behaviour of each blueprint is detailed in their EventGraph.

A common event in EventGraph is the Event Tick, an event that occurs on each tick of the engine. In Unreal Engine, the tick is defined as a periodic event that occurs, by default, before each frame render. For the Meta Quest Pro that can display up to 90 frames per second, it means that every function associated to Event Tick is evaluated every 1/90 second, before rendering the frame. Adding more time consuming functions to Event Tick may lower the frame rate and consequently, increase the period between 2 frames. It is important to keep this in mind when you modify the behaviour of a blueprint.

Unreal Engine offers templates as starters for different kind of games. I picked 'Virtual Reality' with its starter content and set the preset quality to 'high'. Every step of the RMET in VR implementation will start from this point.

4.2.2 VR Environment

The 'Virtual Reality' template is a open-air environment where the player can move via teleportation with its controller's joystick,grab little cubes with his hand and pick a gun that shoots little balls. From the player's point of view, he can only see his hand in futuristic gloves and in the background he can hear campfire sounds.

All objects have been removed except the player spawn point, side walls and ground, the blue sky background and environment lightning. The player spawn point component defines where the Pawn appears in the level and its orientation. In this template, the Pawn blueprint used is 'VR Pawn' and it contains both hands' mesh and textures, the player's camera and the EventGraph that defines player's controls (see appendix B for more details). In the classic RMET, the participant doesn't have to move, he's usually sitting on a chair. To reproduce this in my game, I have disabled unnatural movements like the teleportation and the camera snap turn, a command allowing him to quickly move its camera on the side without moving its head. The player spawn point will be adjusted once other elements have been added.

4.2.3 Eye tracking in VR

Eye tracking systems measure and record the movement of the eyes and pupils opening size over time. Coupled with head rotation information, we can obtain the eye gaze (i.e. where the subject is looking at in his environment). There are two main types of eye tracking methods : screen-based and head-mounted based. Screen-based requires the user to sit in front of a screen

Figure 4.2: Default Virtual Reality game proposed by Unreal Engine. It is a demonstration of the engine's VR features.

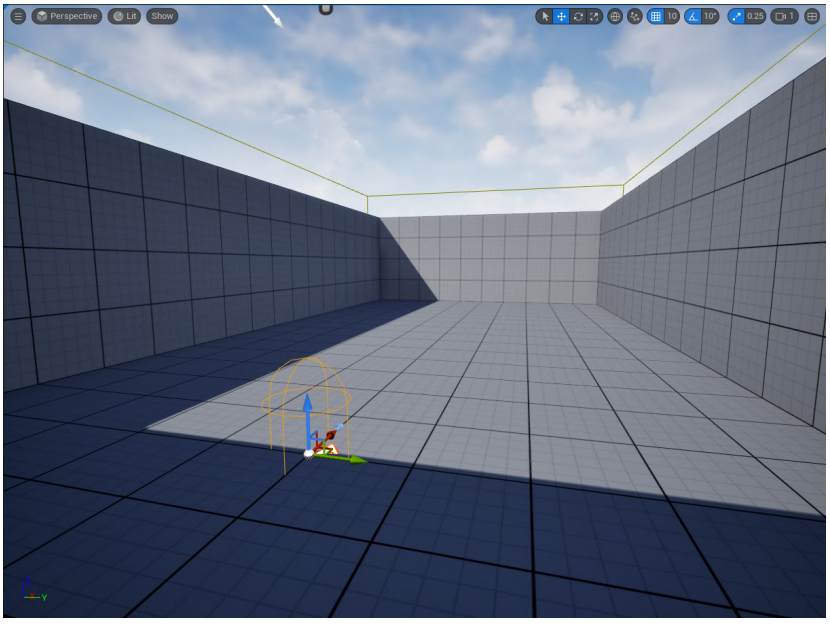


Figure 4.3: The same game after I removed every element not useful to my implementation

and look at screen-based content, whereas head-mounted based eye tracking allows the user to freely move and look all around him, either in real world or in a virtual space because the system is fixed to him at eyes level ("How do eye trackers work?", n.d.).

The Meta Quest Pro uses head-mounted based eye tracking but Meta doesn't provide much information on their hardware. However block diagrams in appendix A tells us that the camera

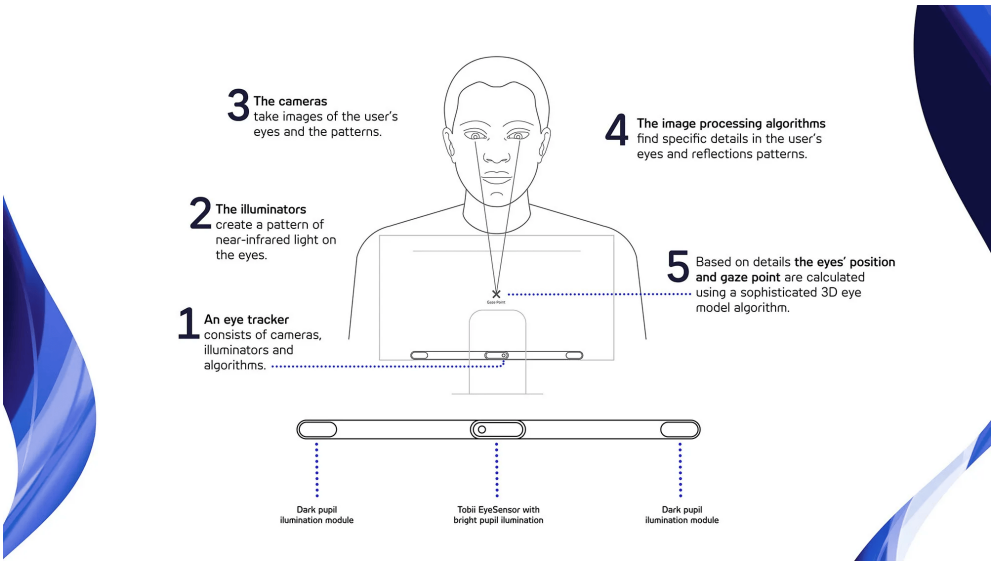


Figure 4.4: Screen-based eye tracking : near-infrared lights are emitted to the eyes and their reflection is captured to evaluate the eye gaze ("How do eye trackers work?", n.d.)

used is a OVM6211, a CMOS Image sensor based module capable to capture 400*400 pixels videos at 120 fps ("OVM6211", n.d.).

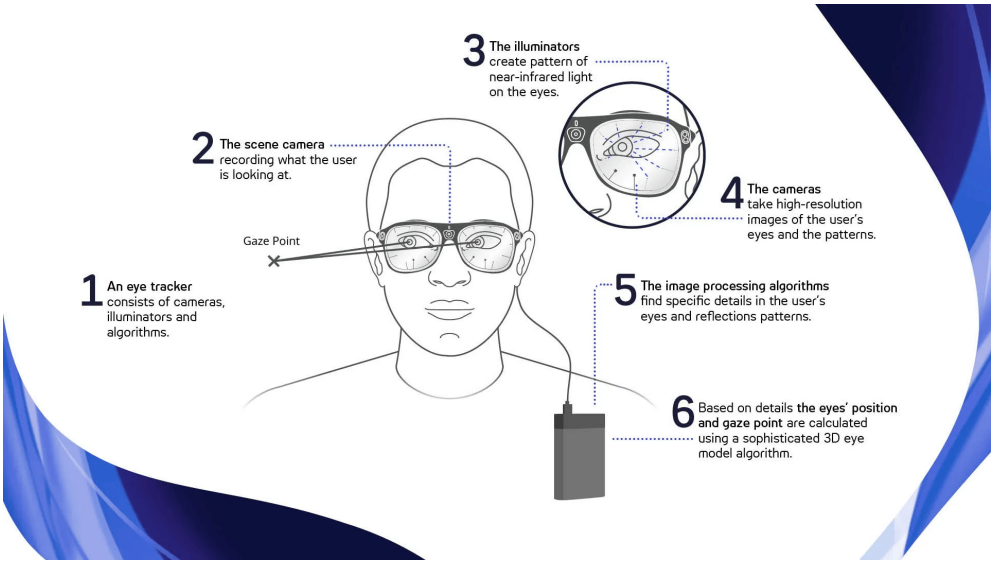


Figure 4.5: Head-mounted eye tracking : In this example it's done with eye tracking glasses from Tobii but it's the same principle for Meta Quest Pro "How do eye trackers work?", n.d.)

I don't know how the whole Quest Pro hardware has been configured and how raw eye tracking data are processed by Meta. At which frequency data are pulled from the OVM6211 ? Do I get an average of the eye gaze at an interval between t-1 and instant-t or the exact value at this instant ? These questions remain unanswered to me, the only thing I know from them is that they expose processed eye gaze data through their implementation of OpenXR APIs

(“Eye tracking on Meta Quest Pro — Meta Store”, n.d.)

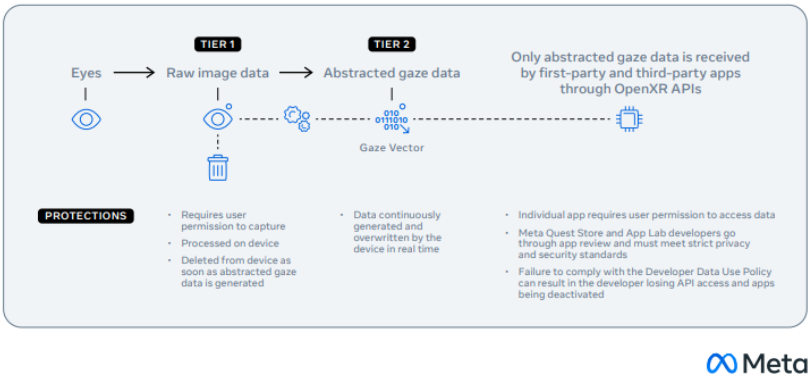


Figure 4.6: Meta’s eye tracking implementation for Meta Quest Pro from their white paper (“Eye tracking on Meta Quest Pro — Meta Store”, n.d.)

After configuring the Meta Quest Pro to activate and calibrate eye tracking, I activated Unreal’s built-in OpenXREyeTracker plugin and implemented a quick hit test in Unreal Engine following a tutorial from Varjo, a Finnish XR headset manufacturer (“Eye tracking”, 2024). The Meta OpenXR runtime fully implements the *XR_EXT_eye_gaze_interaction* extension (“The OpenXR™ Specification”, n.d.), however, some data are not currently supported by the plugin.

Property	**Description**	**Type**	**Available?**
Confidence Value	Value [0..1] that represents confidence in the gaze ray data	Float[1]	Yes
Fixation Point	Location that the eyes converge	Vector	No
Gaze Direction	Forward direction of the unified gaze ray	Vector[2]	Yes
Gaze Origin	Origin of the unified gaze ray (== Camera Origin)	Vector	Yes
Is Left Eye Blink	Left eye blink data. True if the eye is closed, False if open	Boolean	No
Is Right Eye Blink	Right eye blink data. True if the eye is closed, False if open	Boolean	No
Left Pupil Diameter	Diameter of the left pupil	Float	No
Right Pupil Diameter	Diameter of the right pupil	Float	No

Table 4.1: unreal.EyeTrackerGazeData : Represents a unified gaze ray with normalised values that incorporates both eyes (“unreal.EyeTrackerGazeData — Unreal Python 5.3 (Experimental) documentation”, n.d.)

[1]For the Meta Quest Pro, it is either 0.0 or 0.5
[2]Normalised 3D Vector

I get gaze data on each tick before each frame render. Each property is converted to a string, then they are all stored in an array with a string of the UTC as a Unix Time Stamp. The same process is done for the camera origin and camera forward vector[1] but I realised later in the project that they were not relevant. I also add the integer value corresponding to the current question in the RMET (0 is the control test question). Then every item of the array is joined with a comma as a separator to form a unique string. This string is saved into a Comma-Separated Value file (.csv), named after the starting time of the experiment, where each row corresponds to a frame. The file I/O and time related blueprint functions are provided by Rama's Victory Plugin ("Rama's Extra Blueprint Nodes for UE5, No C++ Required! - Programming & Scripting / Blueprint", 2021). The .csv format has been chosen because it's easy to read-write from it, specially for later data processing detailed in section 4.3. A full implementation of this function in the blueprint 'VR Pawn' is available in appendix B.4.

This operation occurs before every frame render but it doesn't take too much time so the game still runs smoothly at 90 FPS on VR Preview. One remaining issue is that some times, the eye gaze data is not correctly recorded, I just get Gaze Direction and Gaze Origin vectors equal to $\mathbf{0}_3$. Beside rebooting both headset and Unreal Engine between each VR Preview, there is no fix to this problem that looks recurring according to community post on Meta's and Unreal's forums. Another point to consider is that I am working in VR Preview mode, which means that the application is running on my computer CPU and GPU, the Meta Quest Pro is just a display with two eye trackers connected to the computer via a USB-C cable. I provide more in details about performances in section 5.1. At this point of the implementation, eye gaze is recorded but I still need to record answers from my participant.

4.2.4 Pick an answer

Before implementing a way to pick an answer in my game, it is necessary to import all answers from the RMET and the definition associated to each word. Unreal Engine offers two tools to make and hold data in a game, Structures and Data Tables. Structures are similar structures to ones in C++ and are set via Unreal's graphical user interface. For my game, the structure is really straightforward with 9 variables that represent question number, answers options and their definition. The term level will be used in this thesis but it doesn't refer to UE's level because my whole RMET in VR game is just composed of one level from game engine's perspective. In fact, it refers to the question number in the test. Level 0 is the control test of the RMET and the test ends after level 36.

With the structure defined, I can create a data table that will hold all my data necessary for the RMET. Unreal Engine allows me to fill a data table using data from a .csv file with headers corresponding to the targeted structure. Using python and the pandas library ("pandas documentation — pandas 2.2.2 documentation", n.d.), I created a .csv where each row corresponds to a question with its 4 proposed answers and their definitions. Some definitions are too long (79 is the character limit) to be displayed on one line in the game so each string is modified to include a `<br>` tag to return to the line when it is necessary. Some proposed answers like 'shy' or 'bored' are not defined in the RMET, in that case their definition is set to 'Word not

[1] Normalised 3D Vector

Variable	Description	Type
Row Name	Value[0;36] Corresponds to question number with 0 the control test	String
ans1	First answer option	String
def1	First answer associated definition	String
ans2	Second answer option	String
def2	Second answer associated definition	String
ans3	Third answer option	String
def3	Third answer associated definition	String
ans4	Fourth answer option	String
def4	Fourth answer associated definition	String

Table 4.2: Question Settings Structure defined in Unreal Engine in appendix C

defined'. The script can be found in appendix D.

The chosen design for the participant to submit its answer to a question is a push button. A push button of the size of a hand is easy to implement and easy to use. I followed a tutorial from VirtualRook's Youtube channel ("A Better VR Button In UE / Tutorial - YouTube", n.d.) to create my own actor blueprint named 'BP_Button'. This blueprint consists of two cubic meshes, button and its base, a textRender component to display the proposed answer, and two box collisions, 'Box' and 'hoverBox'(appendix E).

Once an actor or pawn appears in the level, it triggers an event named 'Event BeginPlay'. It's used for every operation associated to the actor/pawn that needs to be done once and that may be important for the actor/pawn. For my button, I use this event to get the reference of my DefinitionScreen actor. This operation is heavy for the engine and shouldn't not be done every tick. I also use this event to load answer and definition column from my Data Table 'Questions2'. Four buttons means four different sets of answers and definitions but they all look and function the same so following OOP paradigm, I created 3 children button blueprints from my default button blueprint. The parent is the first button that loads ans1 and def1 columns from 'Questions2' and children load their own ans and def columns after the parent 'Event BeginPlay'.

In Unreal Engine, 3D objects don't have any physic, which means that the player's pawn can pass through it. Box collisions are necessary to interact with them. First, I need to give a box to my 'VRPawn' hands so they can interact with other components in the game. In my 'BP_Button,' when a player's hand enters the 'hoverBox'—a square box collision that surrounds the entire button mesh—the definition displayed to the participant is updated to reflect the corresponding button's answer for the current level. Additionally, the button's base component material is changed to green, indicating which button the participant is hovering over. Once the hand is no longer in 'hoverBox', the button's base goes back to default black material. The hover interaction implementation is detailed in appendix E.2.

The second box collision, 'Box', enables interaction with the push button. When the hand's box overlaps the button 'Box', the button's component moves down and is brought back into its initial position once the overlap is over (appendix E.3). From the user point of view, his

hand control how far the button is pushed. However sometimes the button is stuck in down position, requiring the participant to press it again to get it unstuck. I couldn't fix this bug, it looks like it's coming from the engine.

The Event Tick (each frame) evaluates the word to display on the button depending on the current level, set it via the textRender component and checks if the button has been pressed by checking if it reached the down position once (the checking condition is reset once the button goes back to initial position). Once the button is pressed, the button's answer is set as the new answer to record, a variable located in the next presented blueprint, the Definition Screen. Also located in Definition Screen, the variable 'Level' is incremented by one after the new answer has been set. Event Tick implementation is detailed in appendix E.4.

The Definition Screen or 'BP_DefinitionScreen' has only 2 components, a white rectangular mesh as a screen and a textRender for text's definition currently displayed. In the original RMET, some words are in **bold**, however textRender doesn't offer any styling option and only supports the `<br>` tag to format the string. There are other ways to render text with style in Unreal Engine but it's more complex. 'BP_DefinitionScreen' also has two variables, 'Level' and 'answerToRecord' that are accessed for reading/writting by buttons actors and 'VRPawn'. After a quick modification, 'VRPawn' now loads a reference of the Definition Screen on Event BeginPlay and saves 'Level' and 'answerToRecord' in the same .csv used for eye gaze data. The default introduction text displayed on screen is an indication on how to display a word's definition. The text displayed on screen is then set every frame to the last definition selected by hovering a button. The full implementation of 'BP_DefinitionScreen' is detailed in appendix F.

The 4 buttons and the Definition Screen are placed in the environment on a rectangular black mesh that acts as an inclined floating table, at the right height and distance to reach buttons for a participant sitting on a chair in the real world. The RMET in VR is almost done, it's missing the main element, the stimulus, Cooper.

4.2.5 Stimuli and Uncanny Valley: Cooper, my MetaHuman

Revealed in February 2021 by Epic Games, MetaHuman Creator is a web browser application which gives the possibility to anyone to easily create hyper realistic human-like avatar with a large range of options. Hyper realistic mesh is not the only thing coming with a MetaHuman, it also has a skeleton that enables highly detailed motions from body members to micro face expressions with Unreal Engine 5 ("MetaHuman — Realistic Person Creator", n.d.,"MetaHuman Creator The starting point of the metaverse", n.d.).

Before diving into the implementation of my stimulus in the RMET in VR, it is important to have in mind the main problem that comes with the use of human-like avatars : the uncanny valley. Defined in 1970 by Masahiro Mori, a robotics professor at Tokyo Institute of Technology, in an essay published in the Japanese journal Energy that didn't get much attention at the time but got more attraction in the last decades due to the appearance of more human-like robots and virtual avatars in our life.

Figure 4.7: Cooper, one of many default MetaHuman offered by the MetaHuman Creator ("MetaHuman — Realistic Person Creator", n.d.)

An English translation of its essay has been published in 2012 where its editor presents Mori's hypothesis.

> "he hypothesized that a person's response to a humanlike robot would abruptly shift from empathy to revulsion as it approached, but failed to attain, a lifelike appearance. This descent into eeriness is known as the uncanny valley." Mori et al., 2012

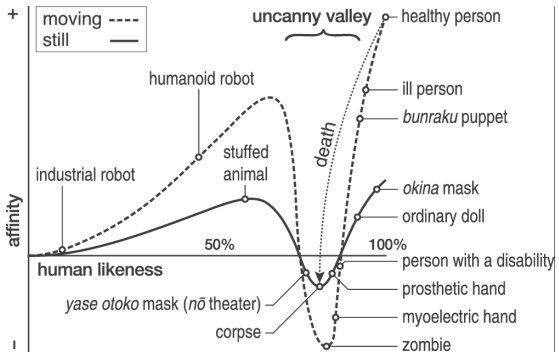


Figure 4.8: Another hypothesis from Masahiro Mori is that the movement amplifies the uncanny valley (Mori et al., 2012)

Does the uncanny valley exists ? Yes and no. There are many theories to explain it but no clear evidences (Burleigh et al., 2013). Moreover, there is no universal consensus on methodologies to test this hypothesis due to fast progress in computer generated graphics, between two experiments the quality of stimuli varies a lot (Diel et al., 2021). In my case with the RMET, it was important to consider if adults in forensics with ASD experience the uncanny valley. Until now, it's known that children between 5-7 years old with ASD do not experience the uncanny valley (Feng et al., 2018), however there is no definitive answer for adults with ASD as it has only been studied in the robotic field without a clear answer (Jaramillo, n.d.). It is obvious to say that there is no work on uncanny valley and forensic adult population on the spectrum.

However, recent studies have shown that the uncanny valley doesn't interfere with level 1 perspective taking (a component of the ToM ability) and that its effect can be elicited with a 50ms exposure to the uncanny stimulus (MacDorman et al., 2013, Yam et al., 2024).

Uncanny valley is a complex topic and even if there is no consensus, it has to be considered when designing a VR application because its effect is amplified on HMDs (Hepperle et al., 2020). With the progress of computer generated graphics that tend to photo-realistic renders, it is right to ask :

Can we cross the uncanny valley ?

Computer generated faces produced by artificial intelligence (AI) and machine learning algorithms that are indistinguishable from real faces to the human eyes, already exist and are used in applications like Generated Photos ("Generated Photos — Unique, worry-free model photos", n.d.). Uncanny valley has already been crossed and MetaHumans have the potential to leap over it too. That's why I have chosen them as the stimulus for my RMET in VR.

To start the implementation of my design, I have picked Cooper, a MetaHuman proposed by default by the MetaHuman Creator, that I downloaded into my Unreal project. After activating the MetaHuman plugin and the Groom plugin (this one manages Cooper's hair), I placed Cooper in my game. First issue I encountered is that Cooper's hair make the application crash on start, even if I decrease the quality of its hairs. I had to remove any hairy component from its blueprint and to disable the Groom plugin for the rest of the implementation. Cooper's model can be seen in appendix G.1.

Figure 4.9: Early implementation of Cooper. His lower body can't be seen from the participant and his shirt has been changed from a floral pattern to a monochrome grey to avoid distractions.

Cooper has a neutral expression by default but its face can be animated. Considering the uncanny valley effect, it has been decided with David Murphy to keep face expressions that are not animated for the test. The RMET shows 33 different face expressions over 37 questions (including the control test question). Due to my poor modelling skills, the idea to reproduce the

37 original stimuli was quickly discarded. Moreover, these stimuli are low resolution pictures from old movies so they can't be exploited. Unreal Engine 5 offers a wide range of different face expressions poses for MetaHumans but none of them is present in my list of 33 emotions.

If Unreal Engine and its marketplace can't offer me these 33 MetaHuman face expressions, I have no choice but to make them. Fortunately, Epic Games developed an iOS application, Live Link Face, that allows anyone to record its face with an iPhone frontal camera and export the recording as facial animation asset for MetaHumans. The application offers two methods to do this : Live Link based on Apple's ARKit (an API used to develop AR functionalities from camera footage) and MetaHuman Animator that uses raw data from the frontal camera and the LIDAR next to it to create an animation from lightning and depth information. The second method offers better quality animation but requires a real person in front of the camera to record all its facial features. I have poor acting skills and I have no money to pay an actor to play my 33 needed facial expressions so this method got discarded. The other one that uses ARKit, only process the video footage without depth information as long as it can recognise a human face in front of the camera. It means that I can print a picture of a human, hold it in front of my phone and the application will recognise it as a real person in front of it and create an animation sequence. During my researches, I came across the McGill Face Database, a database of 93 face expressions, including my 33 needed ones, displayed by 2 professional actors (Schmidtmann et al., 2020).

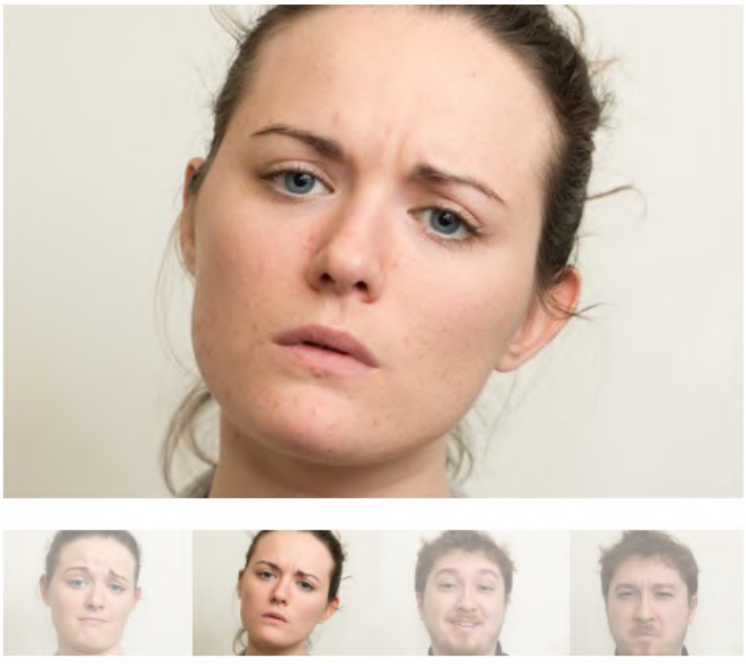


Figure 4.10: The McGill Face Database consists of pairs of pictures from a male and female actor displaying 93 face expressions ("Stimuli & Software", n.d.)

These stimuli have been used and validated in different studies oriented on face recognition so they are a solid base to create my face expression animations for Cooper. With the help of a little python script (see appendix H), I created a slideshow from every frontal picture of the

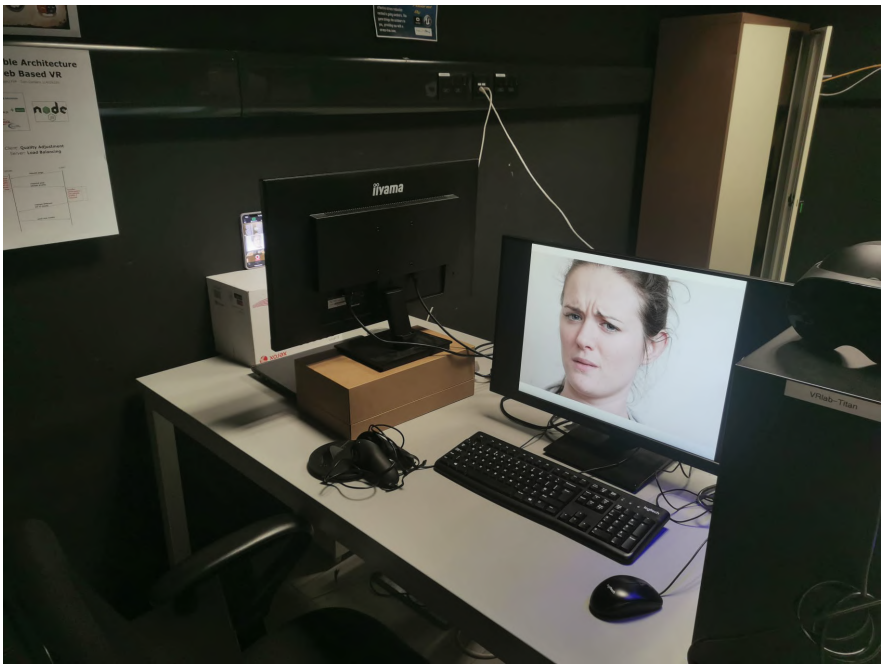


Figure 4.11: Each picture is displayed for 20 seconds with a 10 seconds pause between two, allowing me to easily identify each face expression in the resulting .csv

male actor and played it on my computer's screen in front of my phone running the Live Link Face application. This hacky setup resulted in the creation of a .csv file describing the different micro expressions position key frames to animate a MetaHuman's face at 60 frame per second.

With the LiveLink and LiveLinkFaceImporter plugins activated in Unreal, I can drag and drop my .csv into the engine to be automatically converted to a Level Sequence asset. The sequencer is a new feature from Unreal Engine 5, allowing developers to apply different animations in an additive way to any actor controlled by the Level Sequence. In my case, the level sequence applies an animation to the face of a MetaHuman. After adding my Level Sequence object to the scene, I add Cooper in this sequence and move around the timeline to see if my 33 face expressions are present in the sequence. Even though my phone and slideshow screen were still during the whole process, there are little variations between two key poses. The Level Sequence is generated from a .csv, a file format that I can easily edit with python. From the sequencer, I get the Non-drop frame time code of the best looking key pose for each of my 33 expressions to create a new two minutes long Level Sequence where the MetaHuman face alternates, in the test order, approximately every 3 seconds. Each expression has 200 identical frames and the whole sequence is played at 60 frames per second. The .csv file behind this new Level Sequence asset has been done from the first recorded .csv with a python script that can be read in appendix H.

After creating a new Level Sequence from this new .csv, I bake an animation asset for my MetaHuman's face (in Unreal Engine, baking is the action to create an asset from the sequencer) and assign it to Cooper's blueprint. The whole process to bake an animation asset is detailed in appendix I. From here, I just need to edit Cooper's Event Graph blueprint to set the position

of the animation asset to the corresponding facial expression in the sequence based on the current Level. The position is a floating point representation of seconds and milliseconds and is determined by this formula :

$$\text{Position} = \text{Level} \cdot \frac{\text{Sequence Length}}{37} + \frac{\text{Sequence Length}}{37 \cdot 2} \quad \forall \text{Level} \in [0{,}36].$$

The position is set on each tick so I don't need a long animation sequence like this (123 seconds) because, in theory, only one face expression frame is displayed and if it was 2 or 3, there would be no variation on Cooper's face because there are 200 frames for each face expression. This implementation choice was made to avoid relying on precise float position calculations and to simplify the visualisation of the animation in the sequencer.

The RMET in VR is now complete. After viewing Cooper's face, the participant submits their answer by pressing a button, then Cooper displays a new facial expression, and the test continues until all questions are completed. Simultaneously, all participant responses and eye gaze data are recorded in a .csv file for later processing. The application functions in VR Preview mode, but it still needs to be built to run on the Meta Quest Pro rather than on a computer.

Figure 4.12: Final implementation of the RMET in VR.

4.2.6 Building the application

Meta Quest Pro runs on Meta's operating system designed for their Meta Quest headsets, Meta Horizon OS. This OS is based on Android, an open source mobile operating system sponsored by Google built on the Linux kernel ("Introducing Our Open Mixed Reality Ecosystem", 2024,"Android — Do More With Google on Android Phones & Devices", n.d.). Basically, any Meta Quest application is an android application and Unreal Engine can build an android application from a project's code base. The building process can be tedious as it requires to install all android dependencies such as Android Software Development Kit (SDK). A SDK is a set of tools provided for third-parties developers to develop applications on a specific platform or

framework. Android SDK are tools that use Java and Kotlin, the two main languages to develop android applications. Another dependency to install in order to build an android application is the Android Native Development Kit (NDK). It's similar to the SDK however it's used to compile C/C++ code (language used by Unreal Engine) to Assembly, a low level language that can be called from Java and Kotlin.

After following a tutorial from Unreal Engine community forum ("Unreal Engine 5.4.x for Meta Quest VR — Community tutorial", 2024) to build my application, I came across an error that makes the building process fail at the end. My first hypothesis was that my project contained a problematic component like a plugin or blueprint. In order to verify it, I did the same build process with a new project from Unreal's VR game template but the same build error occurred. With more research on developer's forums, I came across a post indicating the origin of the problem. In the middle of this project's development, Unreal Engine 5.4.4, the latest hotfix version, was released. Since then, certain Android SDK 33 components are used during the build process, even when the target SDK is set to Android SDK 32. However the Meta Quest Pro only supports Android SDK 32 so far ("Unreal Engine 5.4 is here! Find out what's new." n.d. The solution to bypass this, except modifying the building process in source code, is to build the application from the previous engine's version, UE 5.4.3 but it's not possible to install a previous hotfix build of the engine from Epic Games Launcher. The only way to do this is to download the whole UE 5.4.3 code base from Epic's Github repo and build the engine on my computer. The code base is already heavy to download but when the engine is build, it takes a lot of space on the hard drive to install necessary binaries. The setup to build the engine is also tedious and long, it can take up to 5 hours. Due to time restraints and lack of space on my machine's hard drive, I couldn't build the engine and so the VR application. Another idea was to build the project from Unreal Engine 5.3, a previous version that can be easily installed from Epic Games Launcher but it's impossible to downgrade a project to a previous version. I could redo the whole project in Unreal 5.3 but the Meta XR plugin is only compatible with UE 5.4.

My VR application will have to stay as an Unreal Engine project until a fix is deployed. The last step to finish the implementation now is to deliver data gathered by the application in a comprehensive form for David Murphy.

4.3 Data processing and visualisation

This part of the implementation consists to process the .csv file generated by the VR application to create a comprehensive report of the experiment that would help experts like David Murphy in ASD assessment. First, I will do a python script to process data and create a report in .pdf format and in a second time, I will wrap this script in a Graphical User Interface (GUI) and convert the whole thing into an executable that works without any additional configuration or dependencies required.

4.3.1 A script to generate a report

The spreadsheet obtained after running the VR application contains many data about eye gaze and answers from the participant. Each row corresponds to one frame in the game and each column to one property. With python libraries, pandas and NumPy, I get each column and save them as one numpy.array. NumPy is a python library that provides a multidimensional array object and many methods to do fast operations on these arrays and it's also a library I am already familiar with in the context of data processing ("NumPy documentation — NumPy v2.1 Manual", n.d.). In Unreal, data have been saved as strings but after creating my arrays, some of them have been changed to numpy.float64 and numpy.int64.

numpy.array object	**Description**	**Type**
gazeOrigin	Origin of the unified gaze ray (same as cameraOrigin)	String
gazeDirection	Forward direction of the unified gaze ray	String
gazeFixationPoint	Location where the eyes converge	String
gazeConfidenceValue	Value [0..1] representing confidence in the gaze data	numpy.float64
leftEyeBlink	Left eye blink data. 'true' if the eye is closed, 'false' if open	String
rightEyeBlink	Right eye blink data. 'true' if the eye is closed, 'false' if open	String
leftEyePupilDiameter	Diameter of the left pupil	numpy.float64
rightEyePupilDiameter	Diameter of the right pupil	numpy.float64
timestamp	UTC Time of the recorded data	numpy.int64
cameraForwardVector	Forward direction of the camera	String
cameraOrigin	Origin point of the camera	String
answer	Participant response to a question	String
question	Question being asked or displayed	numpy.int64

Table 4.3: List of each numpy.array. 3D coordinates are strings of the same format: "X=0.000 Y=0.000 Z=0.000"

Another thing I have noticed with my .csv is that camera origin and camera forward vectors are one frame late compared to my eye gaze data. I did not find the exact explanation to this, but it must be coming from the fact that these data are coming from the Camera object's, a component with its own blueprint, contained in the 'VRPawn'. A roll is done on these 2 arrays, moving the whole content by -1, this operation doesn't have any consequence on data validity because first and last item of the list correspond to frames where eye tracking is not active.

Before cleaning my data, I still have to convert every string array that describe a 3D coordinates to numpy.float64, thus making cleaning operation easier. I extract each coordinate value and store them in a numpy.array assigned to them : gazeOrigin[String] becomes pGazeOriginX[numpy.float64] + pGazeOriginY[numpy.float64] + pGazeOriginZ[numpy.float64], and this pattern continues similarly for other 3D coordinates.

I join all my numpy.array into a single multidimensional numpy.array, 'pData', on which I

perform multiple operations to keep only relevant data. I delete every row where the confidence value is below 0.5 because the Meta Quest Pro eye tracker confidence value is either 0.0 (off) or 0.5 (on). I employ the same process for every row where one of both eye is blinking but once again, my headset only returns false which means that the feature is not enabled.

With cleaned data, I can do groups of rows corresponding to each question N. Due to my implementation in Unreal, the answer value in each row corresponds to the recorded answer from question N-1. After splitting 'pData' into 37 subsets, I set the correct answer value to each one of them.

There are various methods to visualise eye gaze fixation points in VR, ranging from the widely-used 3D heat maps (Clay et al., n.d.) to more innovative approaches, such as rendering materialised points within the 3D scene (Ugwitz et al., 2022). The latter is way harder to implement and I want a visualisation that can be printed on paper. Moreover, the gaze fixation data is not provided by my headset, I only have the gaze origin and gaze direction. These two values are not enough to calculate a fixation point in a 3D space, however it's enough to get the intersection point of a line with a plane in a 3D space. With x_plane, a plane parallel to yz-axis at a fixed distance x equal to the x position of Cooper's face, and the eye gaze direction, I can calculate the intersection of the line defined by the eye gaze and this plane. Every intersection point on this plane and around Cooper's face are the points that I will represent on my 2D heat map.

Without being overly rigorous in my maths, the solution to my problem can be expressed as :

Given:

1. A normalised vector $\mathbf{v} = (v_x, v_y, v_z)$ with $v_x = pGazeDirectionX; v_y = pGazeDirectionY; v_z = pGazeDirectionZ$.

2. A point $P_0(x_0, y_0, z_0)$ that lies on the vector $\mathbf{v}$ with $x_0 = pGazeOriginX; y_0 = pGazeOriginY; z_0 = pGazeOriginZ$.

3. A plane parallel to the yz-axis at a fixed position $x = x_{\text{plane}}$ with $x_{\text{plane}} = x\_plane[0]$.

The line intersects the plane when the x-coordinate of $P(t)$ equals x_{plane}, thus the intersection point is :

$$P_{\text{intersection}} = \left(x_{\text{plane}},\ y_0 + \frac{x_{\text{plane}} - x_0}{v_x} v_y,\ z_0 + \frac{x_{\text{plane}} - x_0}{v_x} v_z \right)$$

The calculated coordinates are float values accurate to three decimal places, however a 0.001 distance variation is to small to distinguish in the VR environment and there low chances that I have 2 intersection points in the same place, meaning no region of interest could emerge on my heat map. Considering this, I round them to the nearest half, to one decimal place.

Figure 4.13: Every calculated intersection points inside the green square are used to create a heat map. Orange square dimensions : 0.5 x 0.5.

Algorithm 1 Calculate intersection points to 0.5 precision

```
1: CooperXPosition ← 392.928
2: x_plane ← [CooperXPosition, 0, 0]
3: Initialise intersectionPoints as an empty list
4: for i = 0 to 36 do
5:     Initialise frames as an empty list
6:     for j = 0 to length of questionsData[i][0] do
7:         Define gazeDirectionVector ← questionsData[i][3:6, j]
8:         V ← gazeDirectionVector
9:         V ← V.astype(float64)
10:        V ← V.round(3)
11:        Get gazeOriginX ← questionsData[i][0, j]
12:        Get gazeOriginY ← questionsData[i][1, j]
13:        Get gazeOriginZ ← questionsData[i][2, j]
14:        t ← (x_plane[0] − gazeOriginX) / V[0]
15:        y ← gazeOriginY + t × V[1]
16:        z ← gazeOriginZ + t × V[2]
17:        intersectionPoint ← [x_plane[0], y, z]
18:        intersectionPoint ← intersectionPoint.round(2)
19:        if y ≠ ∞ and z ≠ ∞ and y ≠ −∞ and z ≠ −∞ then
20:            y ← round(y × 2)/2
21:            z ← round(z × 2)/2
22:        frames.append(intersectionPoint)
23:    intersectionPoints.append(frames)
```

Once every intersection point has been calculated, I then proceed to create a heat map of fixation points on and around Cooper's face within the plane. Each figure is done with the Matplotlib library that also allows to save them in a .pdf file ("Matplotlib — Visualization with Python", n.d.). For each question there is a grid with identical dimensions to those of the green square shown in the figure 4.13. Every point on the grid corresponds to a corner of an orange square from the same figure. The value associated with each point starts at 0 and is incremented by one each time an intersection point from a given frame matches the corresponding point on the grid (Pramuditya, 2023).

Algorithm 2 Count intersection points on the grid

Import numpy as np
$CooperXPosition \leftarrow 392.928$
$x\_plane \leftarrow [CooperXPosition, 0, 0]$
$greenSquareSideSize \leftarrow 37.0$
$greenSquareLocation \leftarrow [x\_plane[0], 0.0, 161.0]$
$x\_axis \leftarrow$ np.arange$(-\frac{greenSquareSideSize}{2} + greenSquareLocation[1], \frac{greenSquareSideSize}{2} + 0.5 + greenSquareLocation[1], 0.5)$
$y\_axis \leftarrow$ np.arange$(-\frac{greenSquareSideSize}{2} + greenSquareLocation[2], \frac{greenSquareSideSize}{2} + 0.5 + greenSquareLocation[2], 0.5)$
for $i = 0$ to 36 **do**
    $x \leftarrow$ np.array([])
    $y \leftarrow$ np.array([])
    for $j = 0$ to length of intersectionPoints[i] **do**
        $x \leftarrow$ np.append$(x, intersectionPoints[i][j][1])$
        $y \leftarrow$ np.append$(y, intersectionPoints[i][j][2])$
    $X, Y \leftarrow$ np.meshgrid(x_axis, y_axis)
    $Z \leftarrow$ np.zeros_like(X)
    for $l = 0$ to length of x_axis **do**
        for $m = 0$ to length of y_axis **do**
            for $n = 0$ to length of x **do** ▷ len(x) == len(y) so no need for another loop
                if $x[n] = x\_axis[l]$ **and** $y[n] = y\_axis[m]$ **then**
                $Z[l][m] \leftarrow Z[l][m] + 1$

The resulting Z 2D array, is then used to plot a coloured 2D heat map. The higher is the Z value for a given point, the more it tends to red on the heat map. The heat map opacity is lowered so a picture of Cooper and the green square can be inserted below, allowing a better visualisation of different regions of interest. 37 figures are generated like this and saved into a .pdf file ("pylab_examples example code: multipage_pdf.py — Matplotlib 2.0.2 documentation", n.d.).

The entire process, from extracting data from a .csv file to generating a .pdf file, is encapsulated within a single function that takes three arguments: the .csv file path, the participant's name or ID as a string, and the author's name for the report.

The python script is working well but it's not easy to use for someone who doesn't know how to type commands in a terminal. To be usable for David Murphy, I must create a GUI to call my data processing function and convert the whole thing into an executable that can run on most computers.

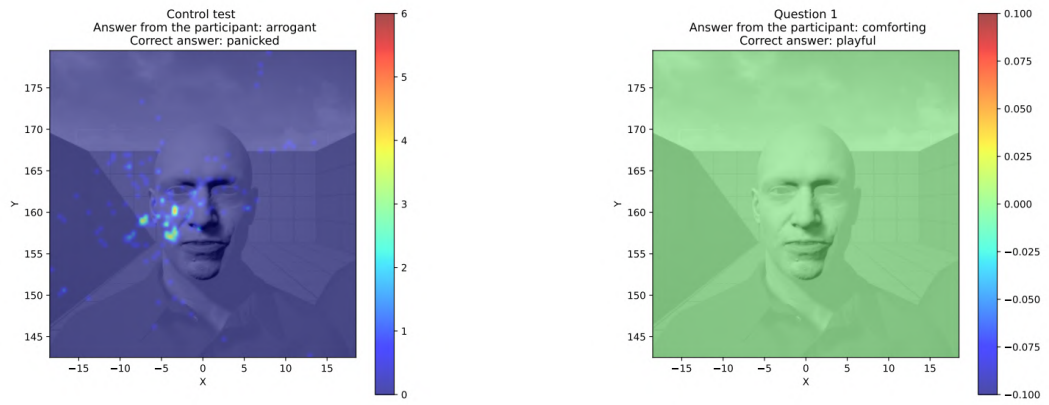


Figure 4.14: There is one figure per page. Participant's answer and the correct one are displayed over the heat map.

4.3.2 An application easy to use on every machine

The first step to convert my python script to an usable application is to add a graphical user interface. Python 3 comes with a standard library to develop GUI, tkinter ("tkinter — Python interface to Tcl/Tk", n.d.). I am already familiar with this library because I have used it during an internship two years ago as well as this year for small python project done in class. In my script, I quickly add 2 entry fields and their labels to my application window so the user can enter the participant's name and its own name as the author of the report. Another label indicates to pick a .csv file by clicking on the 'Browse' button, opening a file explorer window. Once the file selected to get its path, the function developed in the previous subsection 4.3 is called with the corresponding arguments and a .pdf appears. The full script can be read in appendix J

David Murphy's computer runs on Windows, an operating system that use .exe files to run applications. Thanks to PyInstaller, a pip package that bundles any python script and all its dependencies in a single application. However the application created depends on the OS where PyInstaller has been used. All my python development has been done on Linux so I have to reinstall python and all of my script dependencies on Windows then run PyInstaller with the following command :

```
pyinstaller --onefile --add-data "MetaHumanFace.png;." eyeTrackingProcessing.py
```

The resulting .exe runs on Windows 11, however all my figures in the report do not include my MetaHuman picture under the heat map, the corresponding .png must be located in the same directory as the application to come back on my figures.

The full implementation of my design is done, it needs to be tested and evaluated now.

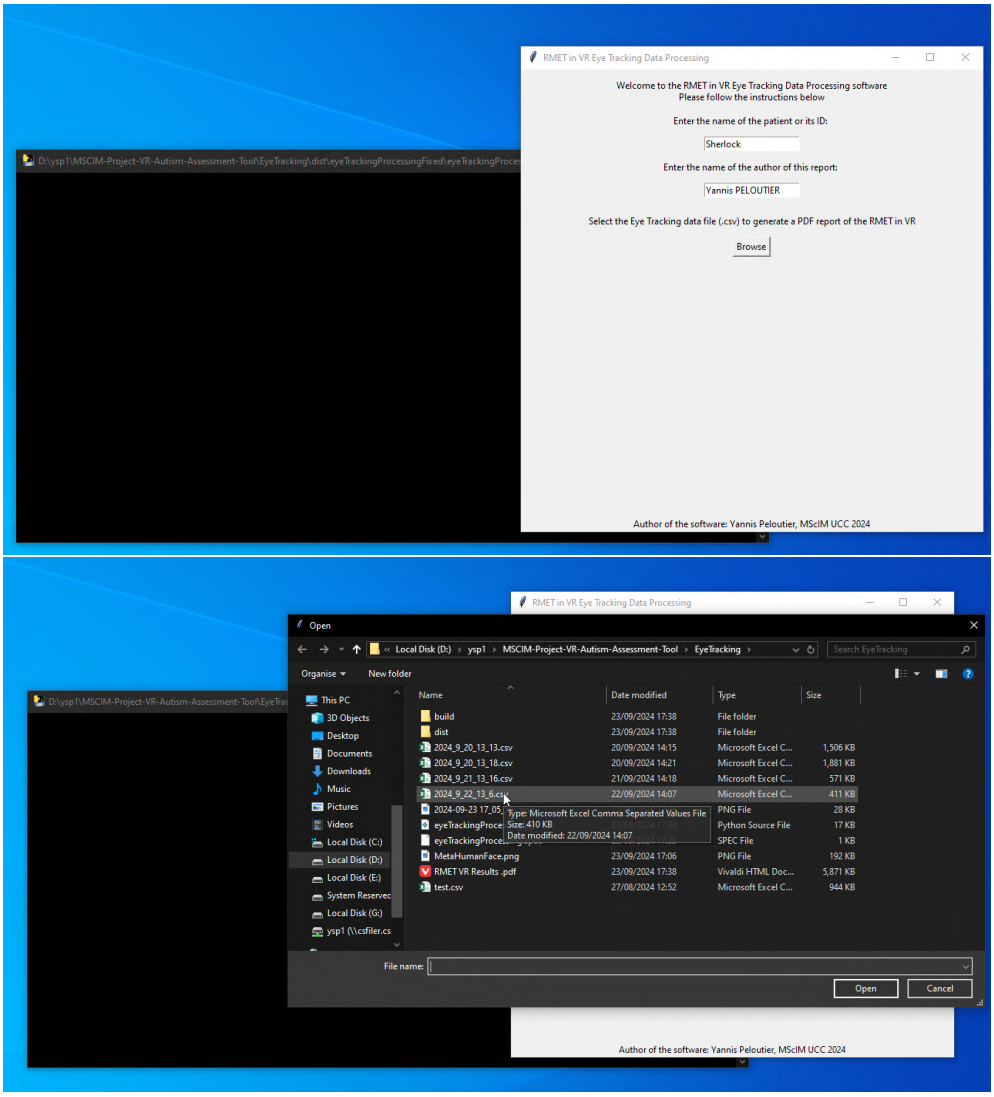


Figure 4.15: Data processing software running on Windows 11 with the file opener function

Chapter 5

Evaluation

In this chapter, I conclude the evaluation process of my implemented solution. After a system testing where I evaluate both my VR application and data processing software performances, I will proceed to user testing where participants tried the RMET in VR and David Murphy tried the data processing solution. Finally, I will give my feedback on the whole system, what can be improved and conclude if this solution is indeed, an tool that might assist in ASD assessment.

5.1 System testing

As seen in subsection 4.2.6, I couldn't port the VR application on the Meta Quest Pro. The application has to run on my computer, using its GPU to render frames on the HMD. The computer is equipped with a NVIDIA TITAN Xp from 2017 and the Unreal Engine's project is configured to 'high' graphics settings since it's creation. These graphics settings range from 'low' to 'cinematic' but I didn't have time to experiment with them. A good indicator of game performances is frame per second (FPS) which indicates how many frames a system can render each second. The higher it is, the smoother the game looks.The eye tracking sampling rate is also tied to this number. Reaching the 90 FPS mark, corresponding to Meta Quest Pro maximum display rate, is crucial to have a smooth VR experience for the participant and relevant data for the evaluator. To monitor my computer's GPU, I use GPU Shark, a free and lightweight monitoring tool for NVIDIA GPUs ("GPU Shark - lightweight and free GPU monitoring tool for NVIDIA GeForce and AMD/ATI Radeon graphics cards — oZone3D.Net", n.d.). To monitor my headset, I use Meta monitoring tools integrated in their developer's desktop application ("Meta Quest Developer Hub", n.d.).

After a cold start, the application runs at a frame rate varying between 85 and 90 FPS, using on average 65% of the headset's CPU. On my computer, the GPU's core is used between 75% and 80% and 85% of its memory. Moreover, the two devices are not overheating. Overall, the application runs nicely and is stable with not a single crash occurring during tests. Detailed logs are included in appendix K.

To evaluate my data processing software, I couldn't collect a sufficient amount of eye gaze data similar to what I would obtain if the RMET in VR experiment was done in real conditions.

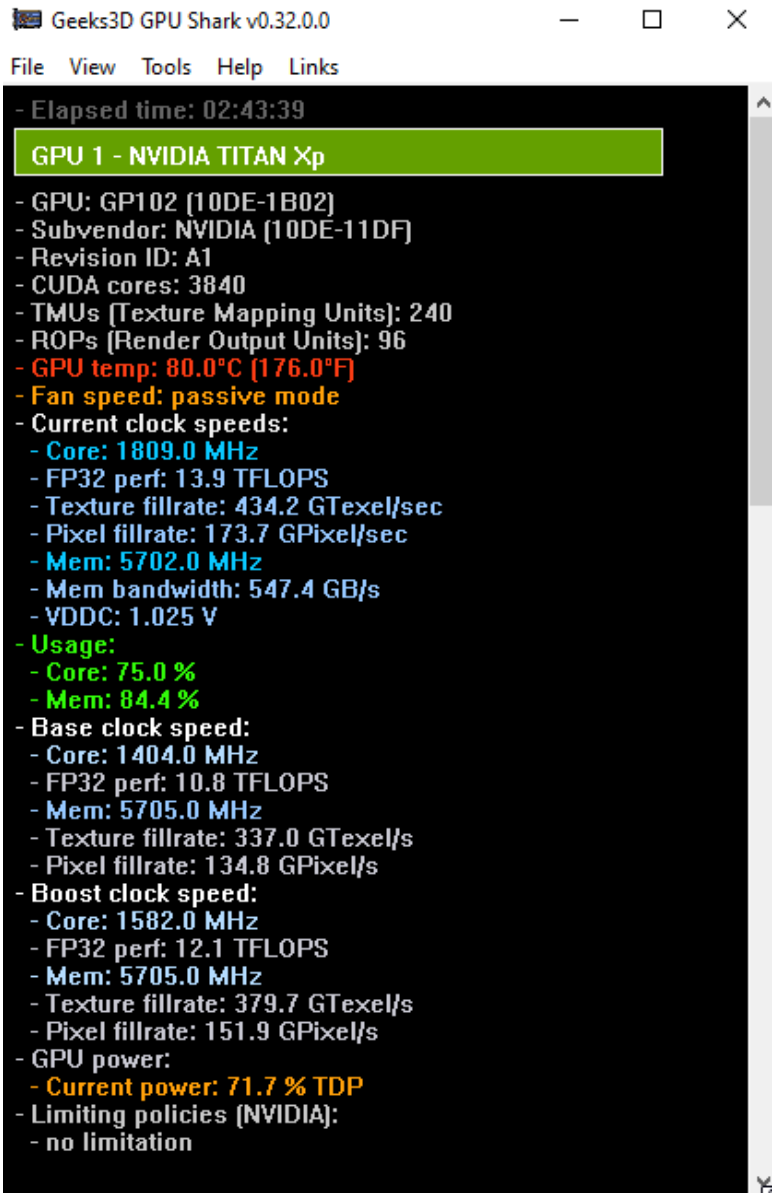


Figure 5.1: GPU Shark screen capture. It's the TITAN Xp that does all the work to render frames on the HMD.

My data processing script takes about 9,5 seconds to process 3000 samples. This amount of sample corresponds to about 1 minute in the VR application. I had to run the latter many times until I obtained a functioning eye tracking that wouldn't return me zeros. By guesstimating my script time complexity to $O(n^2)$ at best, it would approximately take 2 hours and 20 minutes to process 30 000 samples of a RMET in VR session that took 30 minutes. With this system testing done, I can carry on to user testing.

5.2 User testing

This part of the evaluation is split in two sections : the VR application tested in UCC by people external to the project and the data processing software tested by David Murphy at Broadmoor Hospital.

Nine participants, three females and six males, aged between 20 and 27 years, were asked to complete the RMET in VR then complete the NASA Task Load Index (NASA-TLX), the System Usability Scale (SUS) and answer to a question about stimuli quality. Before entering in VR, they were given the same instructions from the RMET and sat on a chair. I also explained to them that the this test was not part of an ASD assessment process. One participant's results were excluded of final analysis because they have ASD.

The NASA-TLX is a widely used tool to assess the perceived workload during or after performing a task (Hart and Field, n.d.). It evaluates the workload by using six dimensions rated on a scale from 0 to 100 :

1. **Mental Demand**: How much mental or cognitive effort was required?
2. **Physical Demand**: How much physical effort was required?
3. **Temporal Demand**: How much time pressure did you feel?
4. **Performance**: How successful do you think you were in accomplishing the goals of the task?
5. **Effort**: How hard did you have to work to accomplish your level of performance?
6. **Frustration**: How irritated, stressed, or annoyed did you feel during the task?

The questionnaire was done on Google Forms with a rating scale from 0 to 10 and results were multiplied by 10 afterwards to get unweighted scores calculated by an Excel spreadsheet ("Measuring Workload – Test Science", 2023). The group unweighted score is equal 31.63, a value corresponding to a somewhat high workload. The highest workload is on the performance dimension, with a group score of 72.50 and the lowest workload on the physical and frustration demand with respectively 28.33 and 28. Through exchanges with participants, I have noticed that one source of frustration is the "button stuck" bug that appeared for few of them.

The second questionnaire, the SUS is also another tool to assess the usability of a system. It consists of a 10 items questionnaire, with responses on a 5 point scale (from "strongly agree" to "strongly disagree"). The questions alternate between positively and negatively worded statements to avoid response bias (Brooke, 1995). Participants were presented the following statements :

- I think that I would like to use this system frequently.
- I found the system unnecessarily complex.
- I thought the system was easy to use.
- I think that I would need the support of a technical person to be able to use this system.

- I found the various functions in this system were well integrated.
- I thought there was too much inconsistency in this system.
- I would imagine that most people would learn to use this system very quickly.
- I found the system very cumbersome to use.
- I felt very confident using the system.
- I needed to learn a lot of things before I could get going with this system.

Group Score Results

Weighted		Raw/Unweighted	
Overall	36.25	Overall	31.67
Diagnostic Subscores		Diagnostic Subscores	
Mental	51.43	Mental	51.43
Physical	28.33	Physical	28.33
Temporal	25.00	Temporal	63.00
Performance	72.50	Performance	72.50
Effort	55.00	Effort	55.00
Frustration	28.00	Frustration	28.00

Figure 5.2: NASA-TLX Group Score for 8 participants ("Measuring Workload – Test Science", 2023)

I didn't calculate the SUS score because my supervisor for this project told me that the test was not relevant because the RMET usability had really been established. Through participant's vocal feedback and a quick look at their response, they all found that the application was intuitive to use by pressing buttons like they would do in real life.

Finally, I had to evaluate the quality of my stimuli, Cooper and his many faces. There are no standardised test to evaluate the quality and authenticity of a human-like avatar. I came across Jonathan Gratch's work on virtual agents used to interact with machines. He evaluates the rapport between a user and a virtual agent (usually a human-like avatar) or in other words, their ability to communicate and interact well with each other. However these evaluations are done in task (Gratch et al., 2007,de Melo et al., 2014) and none of them can be reproduced for my project. Nevertheless, after reading his work, I came up with a question for my participants to evaluate my stimuli :

How authentic are the facial expressions and emotions displayed by Cooper, the MetaHuman in front of you during the test?

I got mixed impressions on Cooper. Some participants found face expressions fairly accurate and easy to read, especially the ones where Cooper is smiling. Some participants, sometimes, had a hard time to associate any of the 4 proposed answers to the facial expression shown,

leading to confusion. Another criticism made to Cooper was that his eyes look "a little dead inside" and for one participant, he looked "robotic". By discussing with participants, none of them expressed an awkward feeling in presence of Cooper that could be a sign of the uncanny valley effect. The main difficulty for them to progress in the test came from the variation of quality between his face expressions.

Figure 5.3: The reflective face expression displayed by a human (left, Schmidtmann et al., 2020) and a MetaHuman in the RMET in VR (right). Cooper doesn't move its head and is showing his teeth.

In parallel of this evaluation, I have sent my data processing software with some datasets to David Murphy. With instructions on how to extract the application and datasets from their archive, I asked him to create a report for each one of them. He came across one bug : the file picker allows to choose files in compressed folders but the application can't open the chosen file. Beside this problem, the application is easy to use and runs smoothly on his computer. I also asked him if data present in the report were sufficient in terms of quantity and quality. The general layout for each questions is good to him. However both axis and the coloured bar around the stimulus are not clear. The report lacks time related information like how long the participant spends on each question and the amount of time spent to look at parts of the virtual scene that are not Cooper's face. David Murphy would also need a summary page with the global RMET score.

Overall, the implementation of my design is good on both system and user aspect. Both software run smoothly with few non-critical bugs. Participants had a good experience with the RMET in VR but were sometimes troubled by Cooper's face expressions. David Murphy appreciates the data processing software and the layout of the reports it generates, however these reports still lack important data to him. Before concluding on this project, I will quickly address what can be done to improve it with some leads to explore.

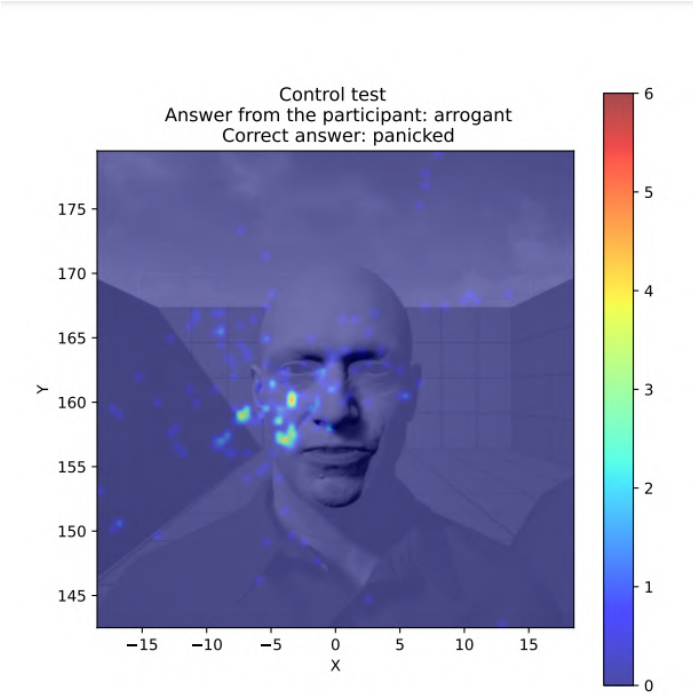


Figure 5.4: First page of the report. The scale on the right indicates how many intersection points were counted at this position on the grid.

5.3 What can be done to improve the application and leads to explore

I implemented my design with success but it comes with some defaults. Obviously, the implementation needs more optimisation on the VR application and the data processing software. Some graphics settings from Unreal Engine and blueprints functions can be enhanced to achieve a constant 90 FPS. On the data processing side, my implementation was really straightforward and with more time and experience, it would be possible to reduce the time complexity,to add missing data to the report and to design a better looking GUI.

Beside performances, the VR Application has two major problems : quality of stimuli and eye tracking accuracy. The process to get an animation sequence was a bit 'hacky', by recording face expressions from pictures with an application designed to record real faces. With the help of professional actors and usage of LIDAR technology proposed by the Live Link Face application, I could have had better animation assets. This solution would also require a study to validate MetaHuman's face expressions. For the eye tracking, all eye gaze related data are not available at this time and the Meta Quest Pro eye tracking is not really reliable because it returns $\mathbf{0}_3$ in most of the time. This issue seems common for a lot of Meta Quest user's on the net. Another headset with more accurate eye trackers should be considered for further research that benefit from allocated budget.

Another headset could also allows for better quality graphics at a high frame rate but at this time it's a technological limitation that all XR headsets have to face.

Immersion is also a forgotten aspect of this project. Depending on David Murphy's needs, a more immersive VR environment could be developed, with better lightnings and the use of sound and haptic to enhance the participant's immersion.

A recurring criticism of the RMET is that it lacks of diversity in its stimuli. It is mostly middle aged Caucasian men and women. MetaHuman Creator proposes limitless possibilities to design hyper realistic human-like avatars that could bring company to Cooper in the RMET in VR.

The VR application was developed with the last Unreal Engine version that comes with bugs like the building pipeline or bold MetaHumans. Once these bugs are fixed, it would be possible to develop a process allowing a tighter collaboration between developers and Broadmoor's experts that could test the application on their headset after each requested adjustment.

My vision of the perfect version of this project is a single application that runs on any headset with eye tracking features and automatically delivers a report of the RMET in VR to David Murphy's e-mail box. However, like every project, this one needs time, money and people to be perfected.

5.4 Conclusion

This MSc Interactive Media project aims to develop a new tool based on VR and eye tracking that could assist with ASD assessment in forensic settings. After months of research on ASD, MetaHumans, VR, eye tracking, and collaboration with David Murphy from Broadmoor Hospital, I delivered a VR-based replication of the Revised Eyes Test as a proof of concept.

The use of hyper-realistic human-like avatars, such as MetaHumans, as stimuli for facial expression recognition tasks coupled with eye tracking, offers new possibilities to develop new tools that can help experts like David Murphy in ASD assessment.

Through the last months, I went across many problems that come with novel technologies like eye tracking in VR and MetaHumans with Unreal Engine but I managed to deliver an application that demonstrates the potential behind these novelties, a proof of concept. My Unreal Engine project is named "HelloWorld" like the first program that many students write when learning a new programming language. I wrote my first "HelloWorld" program in C six years ago at the beginning of my studies, and I am finishing them with this project of the same name. After this collaboration with David Murphy, I am definitely convinced that this "HelloWorld" is a first step in the right direction to make ASD assessment easier in forensic settings.

References

Android — do more with google on android phones & devices. (n.d.). Retrieved September 28, 2024, from https://www.android.com/intl/en_ie/

Baron-Cohen, S. (2000, January 1). Theory of mind and autism: A review. In *International review of research in mental retardation* (pp. 169–184, Vol. 23). Academic Press. https://doi.org/10.1016/S0074-7750(00)80010-5

Baron-Cohen, S., Wheelwright, S., Hill, J., Raste, Y., & Plumb, I. (2001). The "reading the mind in the eyes" test revised version: A study with normal adults, and adults with asperger syndrome or high-functioning autism [_eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1111/1469-7610.00715]. *Journal of Child Psychology and Psychiatry*, *42*(2), 241–251. https://doi.org/10.1111/1469-7610.00715

Baron-Cohen, S., Wheelwright, S., Robinson, J., & Woodbury-Smith, M. (2006). The adult asperger assessment (AAA): A diagnostic method. *Journal of autism and developmental disorders*, *35*, 807–19. https://doi.org/10.1007/s10803-005-0026-5

A better VR button in UE / tutorial - YouTube. (n.d.). Retrieved September 20, 2024, from https://www.youtube.com/watch?v=bSKQGL9znTs&t=674s

Blueprints visual scripting in unreal engine — unreal engine 5.4 documentation — epic developer community [Epic games developer]. (n.d.). Retrieved September 26, 2024, from https://dev.epicgames.com/documentation/en-us/unreal-engine/blueprints-visual-scripting-in-unreal-engine

Brooke, J. (1995). SUS: A quick and dirty usability scale. *Usability Eval. Ind.*, *189*.

Burleigh, T. J., Schoenherr, J. R., & Lacroix, G. L. (2013). Does the uncanny valley exist? an empirical test of the relationship between eeriness and the human likeness of digitally created faces. *Computers in Human Behavior*, *29*(3), 759–771. https://doi.org/10.1016/j.chb.2012.11.021

Chen, Y.-Q., Lin, F.-A., Yang, T.-Y., Yeh, S.-C., Wu, E. H.-K., Poole, J. M., & Shao, C. (2021). A VR-based training and intelligent assessment system integrated with multimodal sensing for children with autism spectrum disorder. *2021 IEEE 3rd Eurasia Conference on IOT, Communication and Engineering (ECICE)*, 191–195. https://doi.org/10.1109/ECICE52819.2021.9645737

Clay, V., König, P., & König, S. (n.d.). Eye tracking in virtual reality. *Journal of Eye Movement Research*, *12*(1), 10.16910/jemr.12.1.3. https://doi.org/10.16910/jemr.12.1.3

Czech, H. (2018). Hans asperger, national socialism, and "race hygiene" in nazi-era vienna. *Molecular Autism*, *9*(1), 29. https://doi.org/10.1186/s13229-018-0208-6

de Melo, C. M., Carnevale, P. J., Read, S. J., & Gratch, J. (2014). Reading people's minds from emotion expressions in interdependent decision making. *Journal of Personality and Social Psychology*, *106*(1), 73–88. https://doi.org/10.1037/a0034251

Diagnostic and statistical manual of mental disorders: DSM-5 (5th ed). (2013). American psychiatric association.

Diel, A., Weigelt, S., & Macdorman, K. F. (2021). A meta-analysis of the uncanny valley's independent and dependent variables. *J. Hum.-Robot Interact.*, *11*(1), 1:1–1:33. https://doi.org/10.1145/3470742

Eye tracking [Varjo for developers]. (2024, September 23). Retrieved September 26, 2024, from https://developer.varjo.com/docs/unreal/ue4/eye-tracking-with-unreal4

Eye tracking on meta quest pro — meta store. (n.d.). Retrieved September 26, 2024, from https://www.meta.com/en-gb/help/quest/articles/getting-started/getting-started-with-quest-pro/eye-tracking/

Feng, S., Wang, X., Wang, Q., Fang, J., Wu, Y., Yi, L., & Wei, K. (2018). The uncanny valley effect in typically developing children and its absence in children with autism spectrum disorders [Publisher: Public Library of Science]. *PLOS ONE*, *13*(11), e0206343. https://doi.org/10.1371/journal.pone.0206343

Frith, C. D., & Corcoran, R. (1996). Exploring 'theory of mind' in people with schizophrenia. *Psychological Medicine*, *26*(3), 521–530. https://doi.org/10.1017/S0033291700035601

Generated photos — unique, worry-free model photos. (n.d.). Retrieved September 28, 2024, from https://generated.photos

GPU shark - lightweight and free GPU monitoring tool for NVIDIA GeForce and AMD/ATI radeon graphics cards — oZone3d.net. (n.d.). Retrieved September 30, 2024, from https://www.ozone3d.net/gpushark/#techdata

Gratch, J., Wang, N., Gerten, J., Fast, E., & Duffy, R. (2007). Creating rapport with virtual agents [ISSN: 0302-9743, 1611-3349 Series Title: Lecture Notes in Computer Science]. In C. Pelachaud, J.-C. Martin, E. André, G. Chollet, K. Karpouzis, & D. Pelé (Eds.), *Intelligent virtual agents* (pp. 125–138, Vol. 4722). Springer Berlin Heidelberg. https://doi.org/10.1007/978-3-540-74997-4_12

Happé, F. G. E. (1994). An advanced test of theory of mind: Understanding of story characters' thoughts and feelings by able autistic, mentally handicapped, and normal children and adults. *Journal of Autism and Developmental Disorders*, *24*(2), 129–154. https://doi.org/10.1007/BF02172093

Hart, S. G., & Field, M. (n.d.). NASA-TASK LOAD INDEX (NASA-TLX); 20 YEARS LATER.

Hepperle, D., Ödell, H., & Wölfel, M. (2020). Differences in the uncanny valley between head-mounted displays and monitors [ISSN: 2642-3596]. *2020 International Conference on Cyberworlds (CW)*, 41–48. https://doi.org/10.1109/CW49994.2020.00014

How do eye trackers work? — a tech-savvy walk-through. (n.d.). Retrieved September 26, 2024, from https://www.tobii.com/resource-center/learn-articles/how-do-eye-trackers-work

ICD-11 for mortality and morbidity statistics. (n.d.). Retrieved June 26, 2024, from https://icd.who.int/browse/2024-01/mms/en#437815624

Introducing our open mixed reality ecosystem [Meta]. (2024, April 22). Retrieved September 28, 2024, from https://about.fb.com/news/2024/04/introducing-our-open-mixed-reality-ecosystem/

Janiszewski, M., Uotinen, L., Szydlowska, M., Munukka, H., Dong, J., & Rinne, M. (2021). Visualization of 3d rock mass properties in underground tunnels using extended reality. *IOP Conference Series Earth and Environmental Science*, *703*, 012046. https://doi.org/10.1088/1755-1315/703/1/012046

Jaramillo, I. (n.d.). Do autistic individuals experience the uncanny valley phenomenon?: The role of theory of mind in human-robot interaction.

Kelly, C., Bernardet, U., & Kessler, K. (2020). A neuro-VR toolbox for assessment and intervention in autism: Brain responses to non-verbal, gaze and proxemics behaviour in virtual humans. *2020 IEEE Conference on Virtual Reality and 3D User Interfaces Abstracts and Workshops (VRW)*, 565–566. https://doi.org/10.1109/VRW50115.2020.00134

L, W. (1981). Asperger's syndrome: A clinical account [Publisher: Psychol Med]. *Psychological medicine*, *11*(1). https://doi.org/10.1017/s0033291700053332

Lord, C., Rutter, M., Dilavore, P. C., Risi, S., Gotham, K., Bishop, S. L., Luyster, R. J., & Guthrie, W. (2016). *ADOS-2: Autism Diagnostic Observation Schedule* [Google-Books-ID: brBdrgEACAAJ]. Hogrefe.

MacDorman, K. F., Srinivas, P., & Patel, H. (2013). The uncanny valley does not interfere with level 1 visual perspective taking. *Computers in Human Behavior*, *29*(4), 1671–1685. https://doi.org/10.1016/j.chb.2013.01.051

Maddox, B. B., Brodkin, E. S., Calkins, M. E., Shea, K., Mullan, K., Hostager, J., Mandell, D. S., & Miller, J. S. (2017). The accuracy of the ADOS-2 in identifying autism among adults with complex psychiatric conditions. *Journal of autism and developmental disorders*, *47*(9), 2703–2709. https://doi.org/10.1007/s10803-017-3188-z

Matplotlib — visualization with python. (n.d.). Retrieved September 29, 2024, from https://matplotlib.org/

Measuring workload – test science. (2023, April 20). Retrieved September 25, 2024, from https://testscience.org/measuring-workload/

Meta quest developer hub. (n.d.). Retrieved September 30, 2024, from https://developers.meta.com/horizon/documentation/unreal/ts-odh/

Meta quest pro (part 2) – block diagrams & teardown of headset and controller [KGOnTech]. (2023, March 15). Retrieved September 2, 2024, from https://kguttag.com/2023/03/15/meta-quest-pro-part-2-block-diagrams-teardown-of-headset-and-controller/

Meta quest pro: Premium mixed reality — meta store. (n.d.). Retrieved September 24, 2024, from https://www.meta.com/ie/quest/quest-pro/tech-specs/#tech-specs

MetaHuman — realistic person creator [Unreal engine]. (n.d.). Retrieved September 28, 2024, from https://www.unrealengine.com/en-US/metahuman

MetaHuman creator the starting point of the metaverse. (n.d.). Retrieved September 17, 2024, from https://ieeexplore.ieee.org/abstract/document/9603558

Mori, M., MacDorman, K. F., & Kageki, N. (2012). The uncanny valley [from the field] [Conference Name: IEEE Robotics & Automation Magazine]. *IEEE Robotics & Automation Magazine*, *19*(2), 98–100. https://doi.org/10.1109/MRA.2012.2192811

Murphy, D. (2006). Theory of mind in asperger's syndrome, schizophrenia and personality disordered forensic patients [Publisher: Routledge _eprint: https://doi.org/10.1080/13546800444000182]. *Cognitive Neuropsychiatry*, *11*(2), 99–111. https://doi.org/10.1080/13546800444000182

Murphy, D., & Radley, J. (2023). Autism spectrum disorder: Assessment and therapeutic approaches within forensic settings. In E. Chaplin, J. M. McCarthy, & R. T. Alexander (Eds.), *Forensic aspects of neurodevelopmental disorders: A clinician's guide* (pp. 137–147). Cambridge University Press. https://doi.org/10.1017/9781108955522.012

NumPy documentation — NumPy v2.1 manual. (n.d.). Retrieved August 28, 2024, from https://numpy.org/doc/stable/index.html#

The OpenXR™ specification. (n.d.). Retrieved September 23, 2024, from https://registry.khronos.org/OpenXR/specs/1.0/html/xrspec.html#XR_EXT_eye_gaze_interaction

OVM6211 [OMNIVISION]. (n.d.). Retrieved September 26, 2024, from https://www.ovt.com/products/ovm6211/

Pandas documentation — pandas 2.2.2 documentation. (n.d.). Retrieved August 28, 2024, from http://pandas.pydata.org/pandas-docs/stable/index.html

Pramuditya, S. (2023, January 4). *Heatmap plot of XYZ data from file with python / numpy / matplotlib (2d colored heatmap)* [Hey, what's going on?]. Retrieved September 21, 2024, from https://syeilendrapramuditya.wordpress.com/2023/01/04/heatmap-plot-of-xyz-data-from-file-with-python-numpy-matplotlib-2d-colored-heatmap/

Premack, D., & Woodruff, G. (1978). Does the chimpanzee have a theory of mind? *Behavioral and Brain Sciences*, *1*(4), 515–526. https://doi.org/10.1017/S0140525X00076512

Pylab_examples example code: Multipage_pdf.py — matplotlib 2.0.2 documentation. (n.d.). Retrieved September 21, 2024, from https://matplotlib.org/2.0.2/examples/pylab_examples/multipage_pdf.html

Rama's extra blueprint nodes for UE5, no c++ required! - programming & scripting / blueprint [Epic developer community forums] [Section: Development]. (2021, May 27). Retrieved September 26, 2024, from https://forums.unrealengine.com/t/ramas-extra-blueprint-nodes-for-ue5-no-c-required/231476

Rivero-Contreras, M., Saldaña, D., & Micai, M. (2023). An introduction to theory of mind: Fundamental concepts and issues. In T. Lopez-Soto, A. Garcia-Lopez, & F. J. Salguero-Lamillar (Eds.), *The theory of mind under scrutiny: Psychopathology, neuroscience, philosophy of mind and artificial intelligence* (pp. 11–33). Springer Nature Switzerland. https://doi.org/10.1007/978-3-031-46742-4_2

Sacks, O. (2011, June 16). *An anthropologist on mars* [Google-Books-ID: kucOxvoPjKwC]. Pan Macmillan.

Schmidtmann, G., Jennings, B. J., Sandra, D. A., Pollock, J., & Gold, I. (2020). The McGill face database: Validation and insights into the recognition of facial expressions of complex mental states [Publisher: SAGE Publications Ltd STM]. *Perception*, *49*(3), 310–329. https://doi.org/10.1177/0301006620901671

Stimuli & software [Schmidtmann lab - vision science]. (n.d.). Retrieved September 28, 2024, from http://www.gunnar-schmidtmann.com/stimuli-software

Stuart, N., Whitehouse, A., Palermo, R., Bothe, E., & Badcock, N. (2023). Eye gaze in autism spectrum disorder: A review of neural evidence for the eye avoidance hypothesis. *Journal of Autism and Developmental Disorders*, *53*(5), 1884–1905. https://doi.org/10.1007/s10803-022-05443-z

Tkinter — python interface to tcl/tk [Python documentation]. (n.d.). Retrieved September 29, 2024, from https://docs.python.org/3/library/tkinter.html

Ugwitz, P., Kvarda, O., Juříková, Z., Šašinka, Č., & Tamm, S. (2022). Eye-tracking in interactive virtual environments: Implementation and evaluation [Number: 3 Publisher: Multidisciplinary Digital Publishing Institute]. *Applied Sciences*, *12*(3), 1027. https://doi.org/10.3390/app12031027

Unreal engine 5.4 is here! find out what's new. [Unreal engine]. (n.d.). Retrieved September 26, 2024, from https://www.unrealengine.com/en-US/blog/unreal-engine-5-4-is-now-available

Unreal engine 5.4.x for meta quest VR — community tutorial [Epic games developer]. (2024, June 25). Retrieved September 28, 2024, from https://dev.epicgames.com/community/learning/tutorials/y4vB/unreal-engine-5-4-x-for-meta-quest-vr

Unreal.EyeTrackerGazeData — unreal python 5.3 (experimental) documentation. (n.d.). Retrieved August 26, 2024, from https://dev.epicgames.com/documentation/en-us/unreal-engine/python-api/class/EyeTrackerGazeData?application_version=5.3#unreal.EyeTrackerGazeData

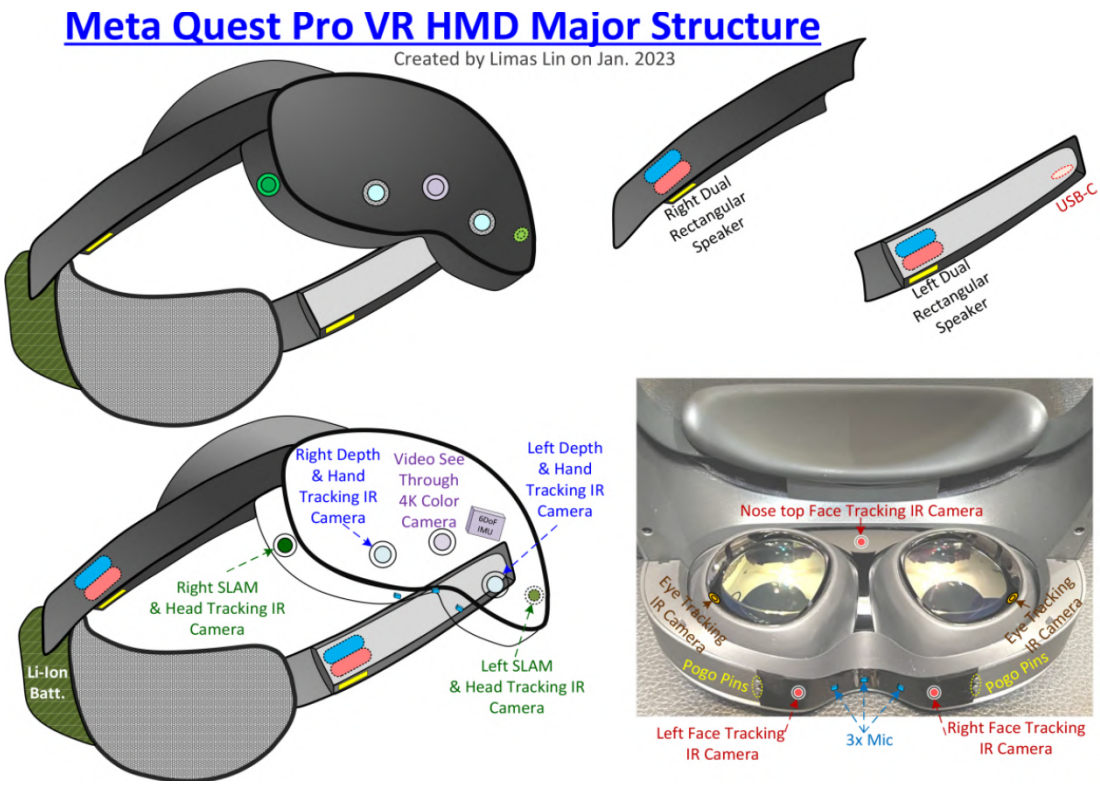
What is the waterfall model? - definition and guide [Software quality]. (n.d.). Retrieved September 26, 2024, from https://www.techtarget.com/searchsoftwarequality/definition/waterfall-model

Yam, J., Gong, T., & Xu, H. (2024). A stimulus exposure of 50 ms elicits the uncanny valley effect. *Heliyon*, *10*(6), e27977. https://doi.org/10.1016/j.heliyon.2024.e27977

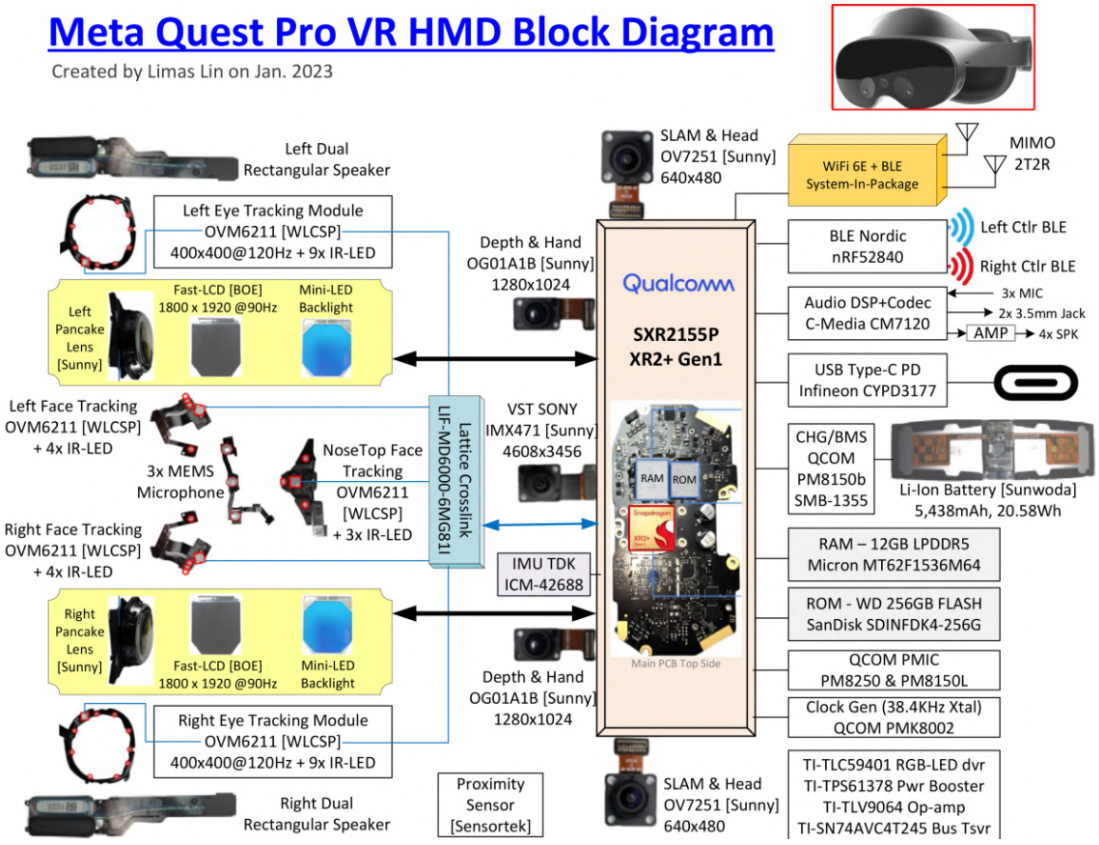
Appendix A

Meta Quest Pro Structure and Block Diagrams

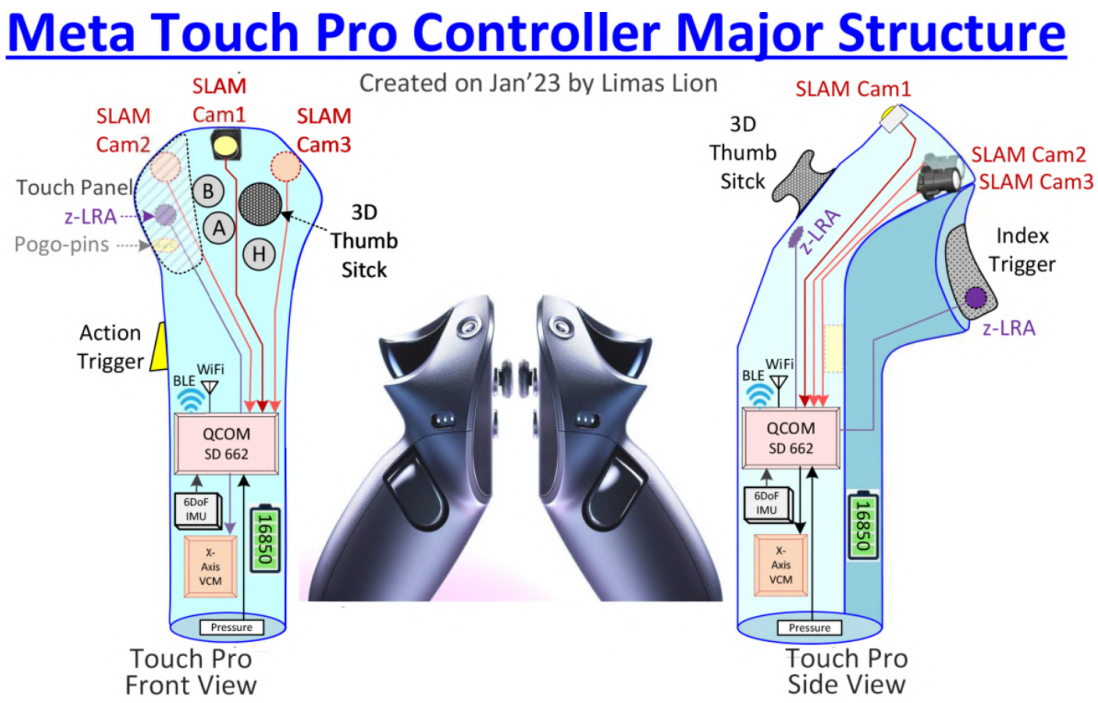
A.1 Meta Quest Pro Major Structure



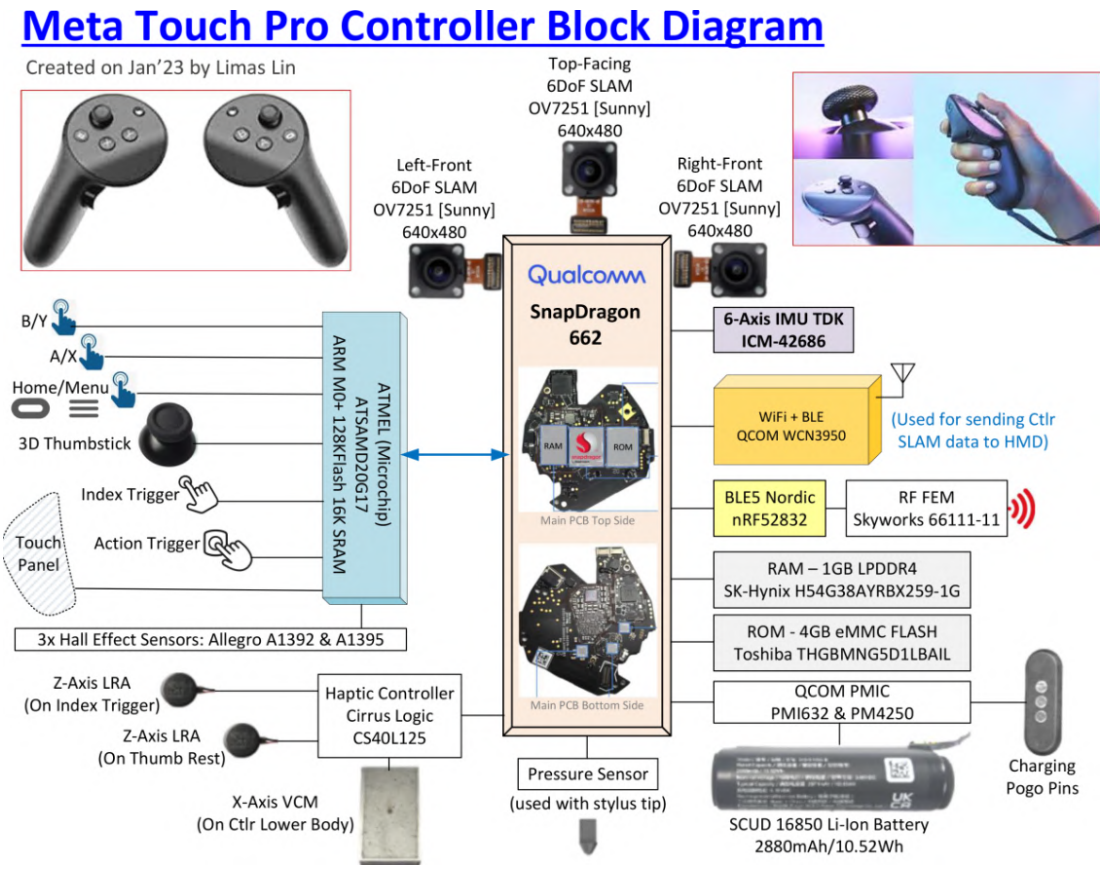
A.2 Meta Quest Pro Major HMD Block Diagram



A.3 Meta Quest Pro Controller Major Structure



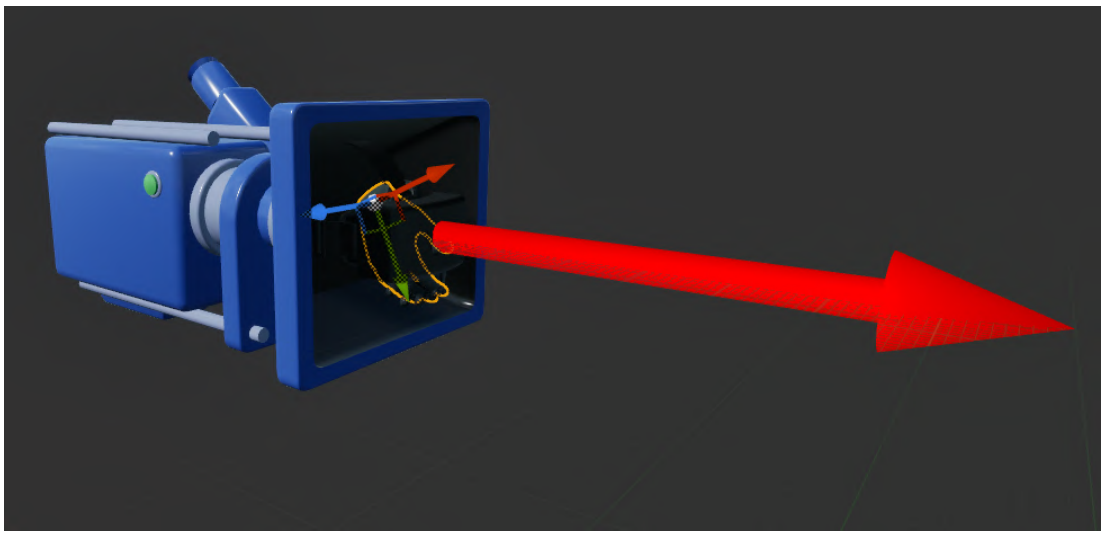
A.4 Meta Quest Pro Controller Block Diagram



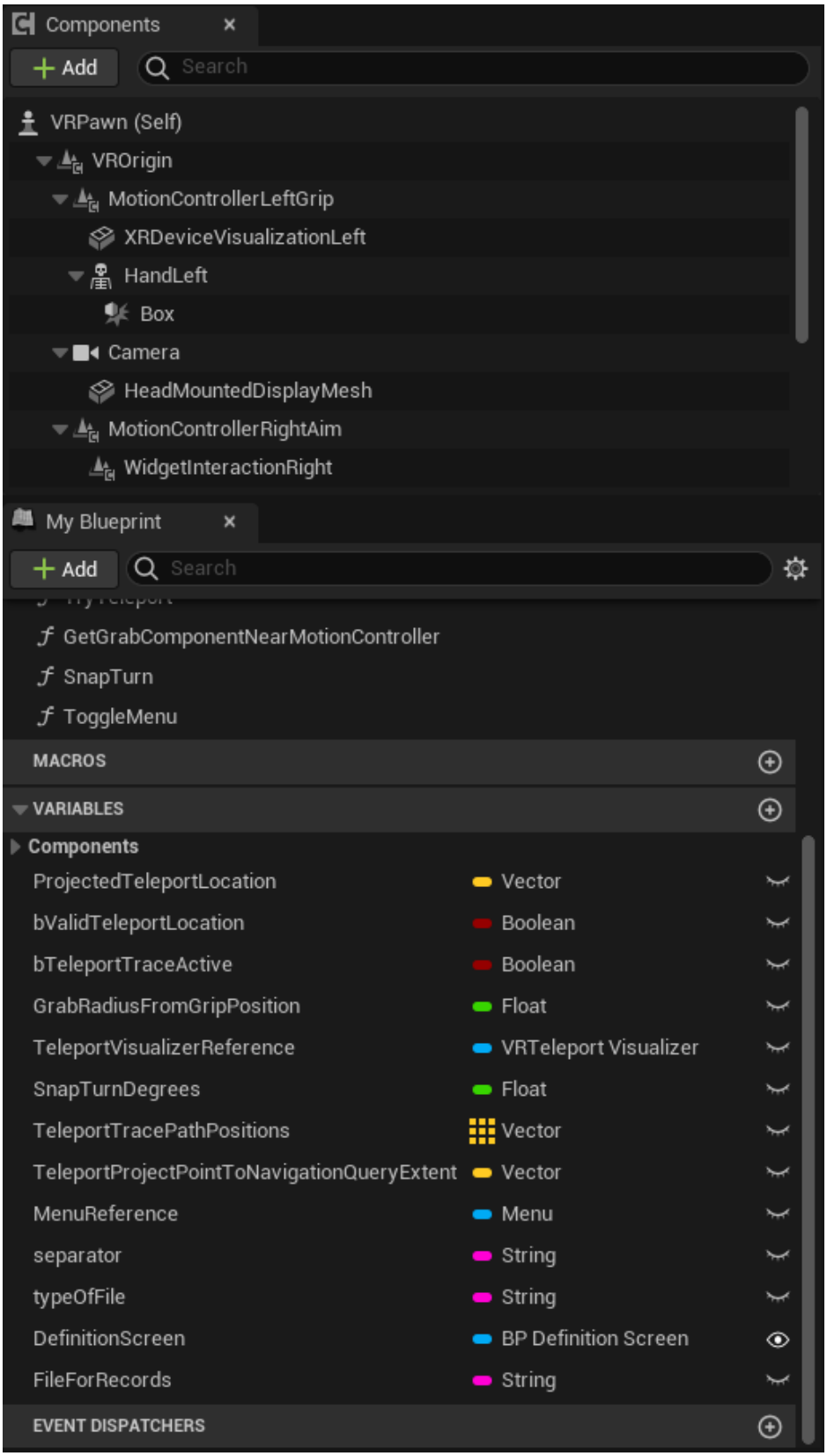
Appendix B

VR Pawn

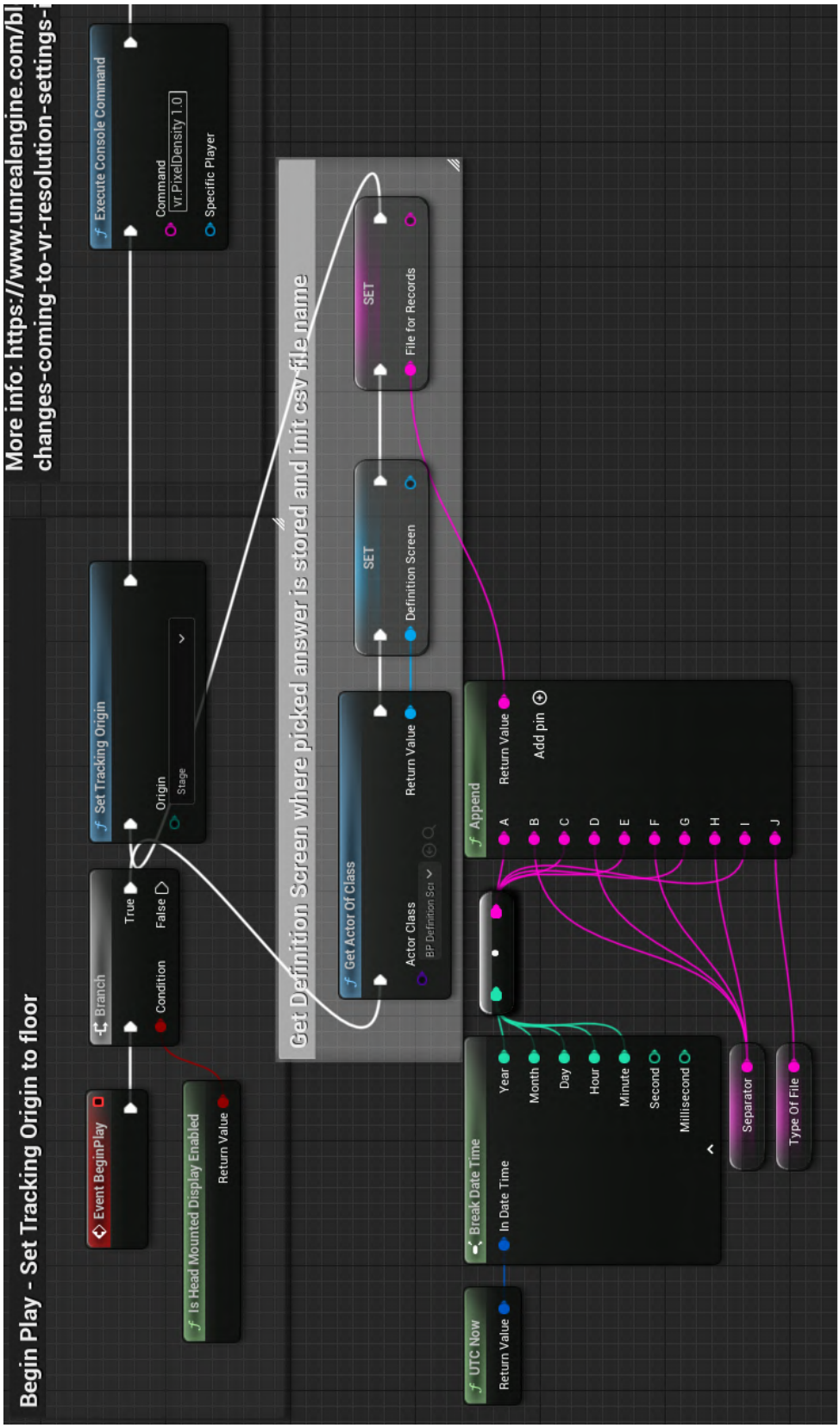
B.1 VR Pawn 3D object



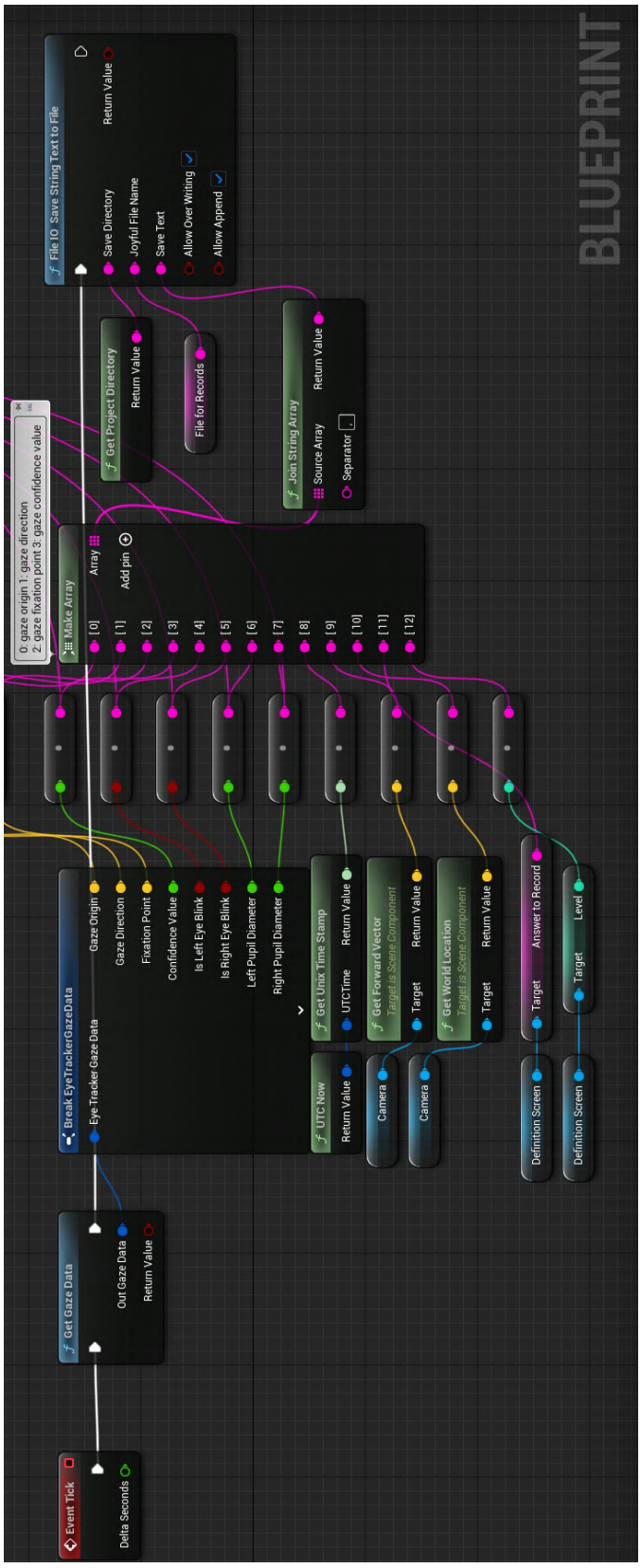
B.2 VR Pawn Components and Variables



B.3 VR Pawn Event BeginPlay

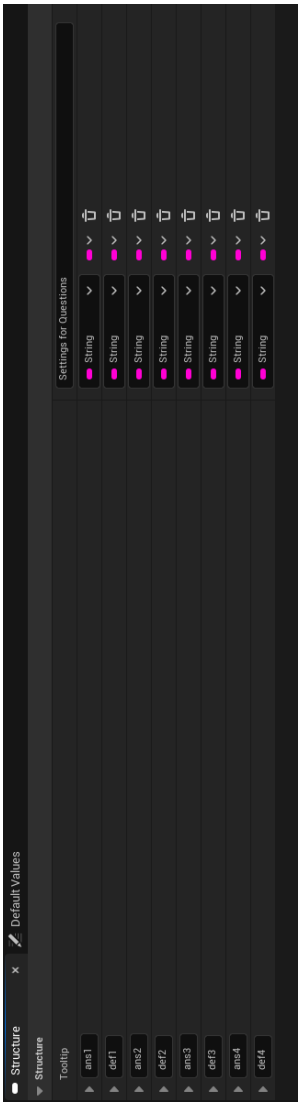


B.4 VR Pawn Event Tick



Appendix C

Question Settings Structure



Appendix D

questions.py

```
1  import pandas as pd
2
3
4  QUESTIONS = [
5      ["jealous", "panicked", "arrogant", "hateful"],
6      ["playful", "comforting", "irritated", "bored"],
7      ["terrified", "upset", "arrogant", "annoyed"],
8      ["joking", "flustered", "desire", "convinced"],
9      ["joking", "insisting", "amused", "relaxed"],
10     ["irritated", "sarcastic", "worried", "friendly"],
11     ["aghast", "fantasizing", "impatient", "alarmed"],
12     ["apologetic", "friendly", "uneasy", "dispirited"],
13     ["despondent", "relieved", "shy", "excited"],
14     ["annoyed", "hostile", "horrified", "preoccupied"],
15     ["cautious", "insisting", "bored", "aghast"],
16     ["terrified", "amused", "regretful", "flirtatious"],
17     ["indifferent", "embarrassed", "sceptical", "dispirited"],
18     ["decisive", "anticipating", "threatening", "shy"],
19     ["irritated", "disappointed", "depressed", "accusing"],
20     ["contemplative", "flustered", "encouraging", "amused"],
21     ["irritated", "thoughtful", "encouraging", "sympathetic"],
22     ["doubtful", "affectionate", "playful", "aghast"],
23     ["decisive", "amused", "aghast", "bored"],
24     ["arrogant", "grateful", "sarcastic", "tentative"],
25     ["dominant", "friendly", "guilty", "horrified"],
26     ["embarrassed", "fantasizing", "confused", "panicked"],
27     ["preoccupied", "grateful", "insisting", "imploring"],
28     ["contented", "apologetic", "defiant", "curious"],
29     ["pensive", "irritated", "excited", "hostile"],
```

```
30      ["panicked", "incredulous", "despondent", "interested"],
31      ["alarmed", "shy", "hostile", "anxious"],
32      ["joking", "cautious", "arrogant", "reassuring"],
33      ["interested", "joking", "affectionate", "contented"],
34      ["impatient", "aghast", "irritated", "reflective"],
35      ["grateful", "flirtatious", "hostile", "disappointed"],
36      ["ashamed", "confident", "joking", "dispirited"],
37      ["serious", "ashamed", "bewildered", "alarmed"],
38      ["embarrassed", "guilty", "fantasizing", "concerned"],
39      ["aghast", "baffled", "distrustful", "terrified"],
40      ["puzzled", "nervous", "insisting", "contemplative"],
41      ["ashamed", "nervous", "suspicious", "indecisive"]
42  ]
43
44  CORRECT_ANSWERS = [
45      "panicked",
46      "playful",
47      "upset",
48      "desire",
49      "insisting",
50      "worried",
51      "fantasizing",
52      "uneasy",
53      "despondent",
54      "preoccupied",
55      "cautious",
56      "regretful",
57      "sceptical",
58      "anticipating",
59      "accusing",
60      "contemplative",
61      "thoughtful",
62      "doubtful",
63      "decisive",
64      "tentative",
65      "friendly",
66      "fantasizing",
67      "preoccupied",
68      "defiant",
69      "pensive",
70      "interested",
71      "hostile",
```

```
72      "cautious",
73      "interested",
74      "reflective",
75      "flirtatious",
76      "confident",
77      "serious",
78      "concerned",
79      "distrustful",
80      "nervous",
81      "suspicious"
82  ]
83
84  DEFINITIONS = [
85      ["ACCUSING", "blaming", "The policeman was accusing the man of
            stealing a wallet."],
86      ["AFFECTIONATE", "showing fondness towards someone", "Most
            mothers are affectionate to their babies by giving them lots
            of kisses and cuddles."],
87      ["AGHAST", "horrified, astonished, alarmed", "Jane was aghast
            when she discovered her house had been burgled."],
88      ["ALARMED", "fearful, worried, filled with anxiety", "Claire was
            alarmed when she thought she was being followed home."],
89      ["AMUSED", "finding something funny", "I was amused by a funny
            joke someone told me."],
90      ["ANNOYED", "irritated, displeased", "Jack was annoyed when he
            found out he had missed the last bus home."],
91      ["ANTICIPATING", "expecting", "At the start of the football
            match, the fans were anticipating a quick goal."],
92      ["ANXIOUS", "worried, tense, uneasy", "The student was feeling
            anxious before taking her final exams."],
93      ["APOLOGETIC", "feeling sorry", "The waiter was very apologetic
            when he spilt soup all over the customer."],
94      ["ARROGANT", "conceited, self-important, having a big opinion of
            oneself", "The arrogant man thought he knew more about
            politics than everyone else in the room."],
95      ["ASHAMED", "overcome with shame or guilt", "The boy felt
            ashamed when his mother discovered him stealing money from
            her purse."],
96      ["ASSERTIVE", "confident, dominant, sure of oneself", "The
            assertive woman demanded that the shop give her a refund."],
97      ["BAFFLED", "confused, puzzled, dumbfounded", "The detectives
            were completely baffled by the murder case."],
```

```
98      ["BEWILDERED", "utterly confused, puzzled, dazed", "The child
            was bewildered when visiting the big city for the first time.
            "],
99      ["CAUTIOUS", "careful, wary", "Sarah was always a bit cautious
            when talking to someone she did not know."],
100     ["COMFORTING", "consoling, compassionate", "The nurse was
            comforting the wounded soldier."],
101     ["CONCERNED", "worried, troubled", "The doctor was concerned
            when his patient took a turn for the worse."],
102     ["CONFIDENT", "self-assured, believing in oneself", "The tennis
            player was feeling very confident about winning his match."],
103     ["CONFUSED", "puzzled, perplexed", "Lizzie was so confused by
            the directions given to her, she got lost."],
104     ["CONTEMPLATIVE", "reflective, thoughtful, considering", "John
            was in a contemplative mood on the eve of his 60th birthday."
            ],
105     ["CONTENTED", "satisfied", "After a nice walk and a good meal,
            David felt very contented."],
106     ["CONVINCED", "certain, absolutely positive", "Richard was
            convinced he had come to the right decision."],
107     ["CURIOUS", "inquisitive, inquiring, prying", "Louise was
            curious about the strange shaped parcel."],
108     ["DECIDING", "making your mind up", "The man was deciding whom
            to vote for in the election."],
109     ["DECISIVE", "already made your mind up", "Jane looked very
            decisive as she walked into the polling station."],
110     ["DEFIANT", "insolent, bold, don t care what anyone else
            thinks", "The animal protester remained defiant even after
            being sent to prison."],
111     ["DEPRESSED", "miserable", "George was depressed when he didn't
            receive any birthday cards."],
112     ["DESIRE", "passion, lust, longing for", "Kate had a strong
            desire for chocolate."],
113     ["DESPONDENT", "gloomy, despairing, without hope", "Gary was
            despondent when he did not get the job he wanted."],
114     ["DISAPPOINTED", "displeased, disgruntled", "Manchester United
            fans were disappointed not to win the Championship."],
115     ["DISPIRITED", "glum, miserable, low", "Adam was dispirited when
            he failed his exams."],
116     ["DISTRUSTFUL", "suspicious, doubtful, wary", "The old woman was
            distrustful of the stranger at her door."],
```

```
117     ["DOMINANT", "commanding, bossy", "The sergeant major looked
            dominant as he inspected the new recruits."],
118     ["DOUBTFUL", "dubious, suspicious, not really believing", "Mary
            was doubtful that her son was telling the truth."],
119     ["DUBIOUS", "doubtful, suspicious", "Peter was dubious when
            offered a surprisingly cheap television in a pub."],
120     ["EAGER", "keen", "On Christmas morning, the children were eager
            to open their presents."],
121     ["EARNEST", "having a serious intention", "Harry was very
            earnest about his religious beliefs."],
122     ["EMBARRASSED", "ashamed", "After forgetting a colleague's name,
            Jenny felt very embarrassed."],
123     ["ENCOURAGING", "hopeful, heartening, supporting", "All the
            parents were encouraging their children in the school sports
            day."],
124     ["ENTERTAINED", "absorbed and amused or pleased by something", "
            I was very entertained by the magician."],
125     ["ENTHUSIASTIC", "very eager, keen", "Susan felt very
            enthusiastic about her new fitness plan."],
126     ["FANTASIZING", "daydreaming", "Emma was fantasizing about being
            a film star."],
127     ["FASCINATED", "captivated, really interested", "At the seaside,
            the children were fascinated by the creatures in the rock
            pools."],
128     ["FEARFUL", "terrified, worried", "In the dark streets, the
            women felt fearful."],
129     ["FLIRTATIOUS", "brazen, saucy, teasing, playful", "Connie was
            accused of being flirtatious when she winked at a stranger at
            a party."],
130     ["FLUSTERED", "confused, nervous and upset", "Sarah felt a bit
            flustered when she realised how late she was for the meeting
            and that she had forgotten an important document."],
131     ["FRIENDLY", "sociable, amiable", "The friendly girl showed the
            tourists the way to the town centre."],
132     ["GRATEFUL", "thankful", "Kelly was very grateful for the
            kindness shown by the stranger."],
133     ["GUILTY", "feeling sorry for doing something wrong", "Charlie
            felt guilty about having an affair."],
134     ["HATEFUL", "showing intense dislike", "The two sisters were
            hateful to each other and always fighting."],
135     ["HOPEFUL", "optimistic", "Larry was hopeful that the post would
            bring good news."],
```

```
136     ["HORRIFIED", "terrified, appalled", "The man was horrified to
            discover that his new wife was already married."],
137     ["HOSTILE", "unfriendly", "The two neighbours were hostile
            towards each other because of an argument about loud music."
            ],
138     ["IMPATIENT", "restless, wanting something to happen soon", "
            Jane grew increasingly impatient as she waited for her friend
            who was already 20 minutes late."],
139     ["IMPLORING", "begging, pleading", "Nicola looked imploring as
            she tried to persuade her dad to lend her the car."],
140     ["INCREDULOUS", "not believing", "Simon was incredulous when he
            heard that he had won the lottery."],
141     ["INDECISIVE", "unsure, hesitant, unable to make your mind up",
            "Tammy was so indecisive that she couldn't even decide what
            to have for lunch."],
142     ["INDIFFERENT", "disinterested, unresponsive, don't care", "
            Terry was completely indifferent as to whether they went to
            the cinema or the pub."],
143     ["INSISTING", "demanding, persisting, maintaining", "After a
            work outing, Frank was insisting he paid the bill for
            everyone."],
144     ["INSULTING", "rude, offensive", "The football crowd was
            insulting the referee after he gave a penalty."],
145     ["INTERESTED", "inquiring, curious", "After seeing Jurassic Park
            , Hugh grew very interested in dinosaurs."],
146     ["INTRIGUED", "very curious, very interested", "A mystery phone
            call intrigued Zoe."],
147     ["IRRITATED", "exasperated, annoyed", "Frances was irritated by
            all the junk mail she received."],
148     ["JEALOUS", "envious", "Tony was jealous of all the taller,
            better-looking boys in his class."],
149     ["JOKING", "being funny, playful", "Gary was always joking with
            his friends."],
150     ["NERVOUS", "apprehensive, tense, worried", "Just before her job
            interview, Alice felt very nervous."],
151     ["OFFENDED", "insulted, wounded, having hurt feelings", "When
            someone made a joke about her weight, Martha felt very
            offended."],
152     ["PANICKED", "distraught, feeling of terror or anxiety", "On
            waking to find the house on fire, the whole family was
            panicked."],
```

```
153     ["PENSIVE", "thinking about something slightly worrying", "Susie
            looked pensive on the way to meeting her boyfriend's parents
            for the first time."],
154     ["PERPLEXED", "bewildered, puzzled, confused", "Frank was
            perplexed by the disappearance of his garden gnomes."],
155     ["PLAYFUL", "full of high spirits and fun", "Neil was feeling
            playful at his birthday party."],
156     ["PREOCCUPIED", "absorbed, engrossed in one's own thoughts", "
            Worrying about her mother's illness made Debbie preoccupied
            at work."],
157     ["PUZZLED", "perplexed, bewildered, confused", "After doing the
            crossword for an hour, June was still puzzled by one clue."],
158     ["REASSURING", "supporting, encouraging, giving someone
            confidence", "Andy tried to look reassuring as he told his
            wife that her new dress did suit her."],
159     ["REFLECTIVE", "contemplative, thoughtful", "George was in a
            reflective mood as he thought about what he'd done with his
            life."],
160     ["REGRETFUL", "sorry", "Lee was always regretful that he had
            never travelled when he was younger."],
161     ["RELAXED", "taking it easy, calm, carefree", "On holiday, Pam
            felt happy and relaxed."],
162     ["RELIEVED", "freed from worry or anxiety", "At the restaurant,
            Ray was relieved to find that he had not forgotten his wallet
            ."],
163     ["RESENTFUL", "bitter, hostile", "The businessman felt very
            resentful towards his younger colleague who had been promoted
            above him."],
164     ["SARCASTIC", "cynical, mocking, scornful", "The comedian made a
            sarcastic comment when someone came into the theatre late."
            ],
165     ["SATISFIED", "content, fulfilled", "Steve felt very satisfied
            after he had got his new flat just how he wanted it."],
166     ["SCEPTICAL", "doubtful, suspicious, mistrusting", "Patrick
            looked sceptical as someone read out his horoscope to him."],
167     ["SERIOUS", "solemn, grave", "The bank manager looked serious as
            he refused Nigel an overdraft."],
168     ["STERN", "severe, strict, firm", "The teacher looked very stern
            as he told the class off."],
169     ["SUSPICIOUS", "disbelieving, suspecting, doubting", "After Sam
            had lost his wallet for the second time at work, he grew
            suspicious of one of his colleagues."],
```

```
170     ["SYMPATHETIC", "kind, compassionate", "The nurse looked
            sympathetic as she told the patient the bad news."],
171     ["TENTATIVE", "hesitant, uncertain, cautious", "Andrew felt a
            bit tentative as he went into the room full of strangers."],
172     ["TERRIFIED", "alarmed, fearful", "The boy was terrified when he
            thought he saw a ghost."],
173     ["THOUGHTFUL", "thinking about something", "Phil looked
            thoughtful as he sat waiting for the girlfriend he was about
            to finish with."],
174     ["THREATENING", "menacing, intimidating", "The large, drunken
            man was acting in a very threatening way."],
175     ["UNEASY", "unsettled, apprehensive, troubled", "Karen felt
            slightly uneasy about accepting a lift from the man she had
            only met that day."],
176     ["UPSET", "agitated, worried, uneasy", "The man was very upset
            when his mother died."],
177     ["WORRIED", "anxious, fretful, troubled", "When her cat went
            missing, the girl was very worried."]
178 ]
179
180 for i in range(len(DEFINITIONS)):
181     DEFINITIONS[i][0]=DEFINITIONS[i][0].lower() # Strings are
            immutable in python
182
183 for i in range(len(DEFINITIONS)):
184     DEFINITIONS[i][1]=DEFINITIONS[i][1]+"<br>"
185     #print(DEFINITIONS[i][1])
186
187 #print(len("Sarah felt a bit flustered when she realised how late
        she was for the meeting a")) # => 79
188
189 for i in range(len(DEFINITIONS)):
190     # Add a <br> tag if the definition is to long to display on one
            line
191     if len(DEFINITIONS[i][2]) > 79:
192         # Find the index of the first space before the 79th
                character
193         index = 79
194         while DEFINITIONS[i][2][index] != "-":
195             index -= 1
196         # Replace the space with "<br>"
```

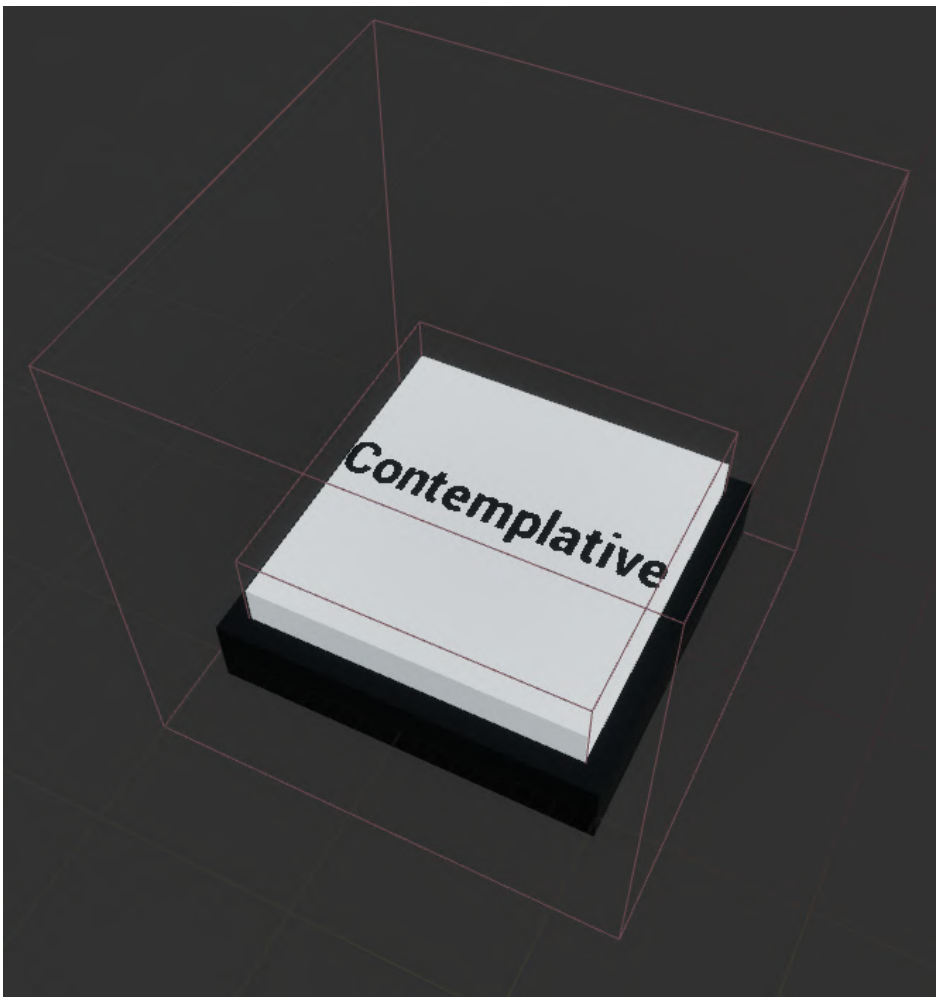
```
197             DEFINITIONS[i][2] = DEFINITIONS[i][2][:index] + "<br>" +
                    DEFINITIONS[i][2][index+1:]
198
199 # Prepare Columns for the DataTable
200 RowName=[
201         "0",
202         "1",
203         "2",
204         "3",
205         "4",
206         "5",
207         "6",
208         "7",
209         "8",
210         "9",
211         "10",
212         "11",
213         "12",
214         "13",
215         "14",
216         "15",
217         "16",
218         "17",
219         "18",
220         "19",
221         "20",
222         "21",
223         "22",
224         "23",
225         "24",
226         "25",
227         "26",
228         "27",
229         "28",
230         "29",
231         "30",
232         "31",
233         "32",
234         "33",
235         "34",
236         "35",
237         "36"
```

```
238 ]
239
240 ans1=[i[0] for i in QUESTIONS]
241 ans2=[i[1] for i in QUESTIONS]
242 ans3=[i[2] for i in QUESTIONS]
243 ans4=[i[3] for i in QUESTIONS]
244
245 def1=[]
246 def2=[]
247 def3=[]
248 def4=[]
249
250 # for each ans, append the corresponding definition
251 def createDefinitons(ans,wordDef,output):
252     for i in range(len(ans)):
253         for j in range(len(wordDef)):
254             if ans[i] == wordDef[j][0]:
255                 output.append(wordDef[j][1]+wordDef[j][2]+"<br>")
256                 break
257             if j == len(wordDef)-1:
258                 output.append("Word not defined<br>")
259
260 createDefinitons(ans1,DEFINITIONS,def1)
261 createDefinitons(ans2,DEFINITIONS,def2)
262 createDefinitons(ans3,DEFINITIONS,def3)
263 createDefinitons(ans4,DEFINITIONS,def4)
264
265 # DataTable for Unreal Engine 5
266 data = {
267     "Row Name": RowName,
268     "ans1": ans1,
269     "def1": def1,
270     "ans2": ans2,
271     "def2": def2,
272     "ans3": ans3,
273     "def3": def3,
274     "ans4": ans4,
275     "def4": def4,
276 }
277
278 DataTable = pd.DataFrame(data)
279 DataTable.to_csv("McGill_Database/mcgill_data.csv", index=False)
```

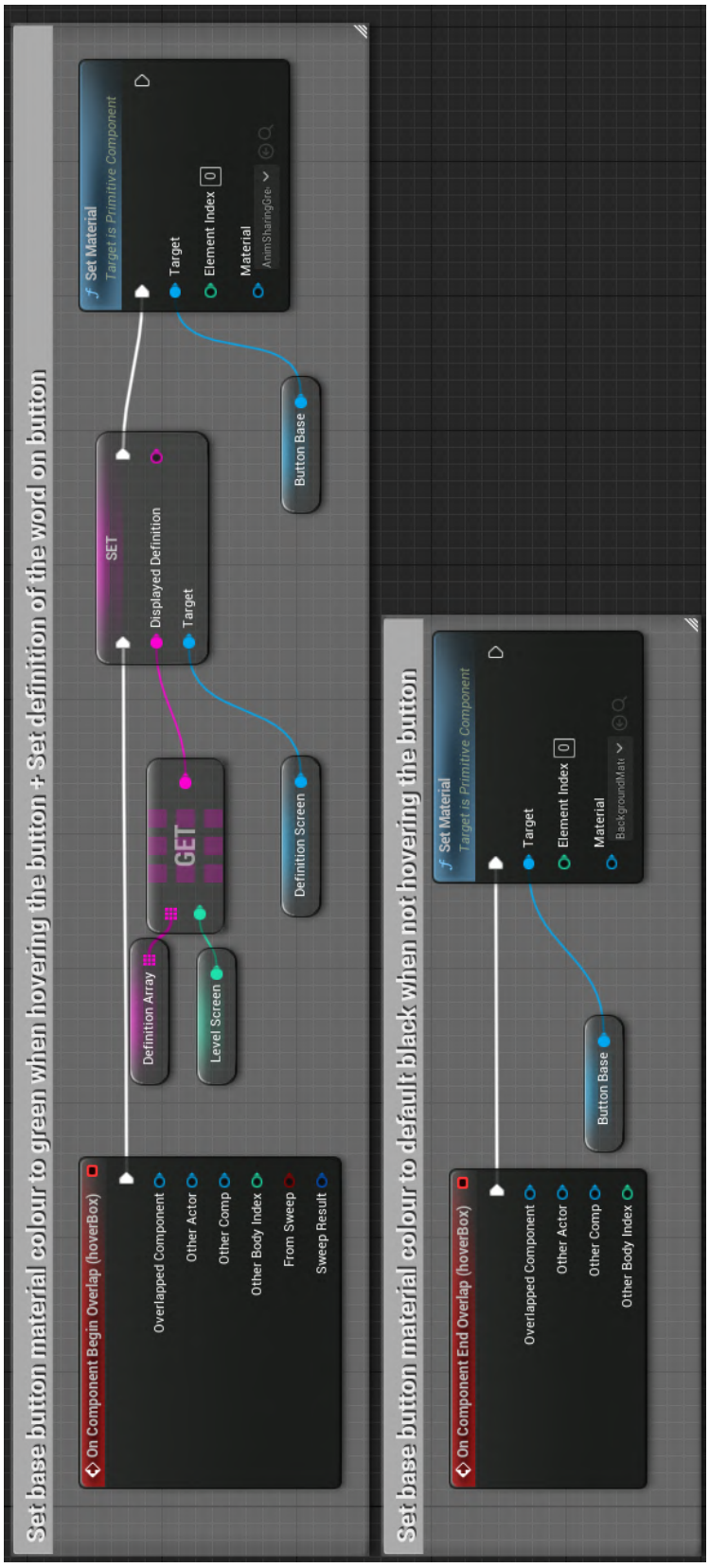
Appendix E

BP_Button

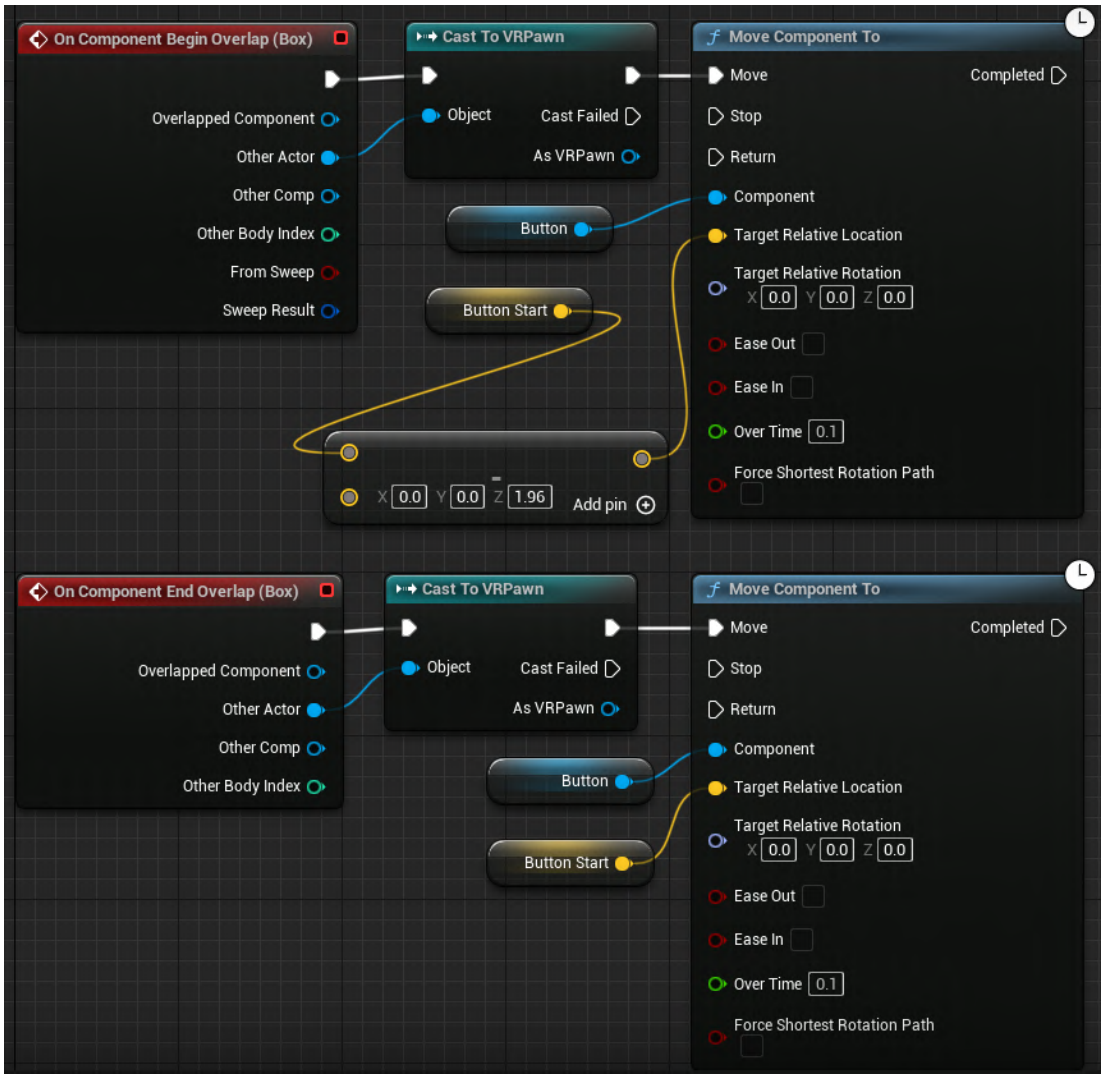
E.1 BP_Button 3D object



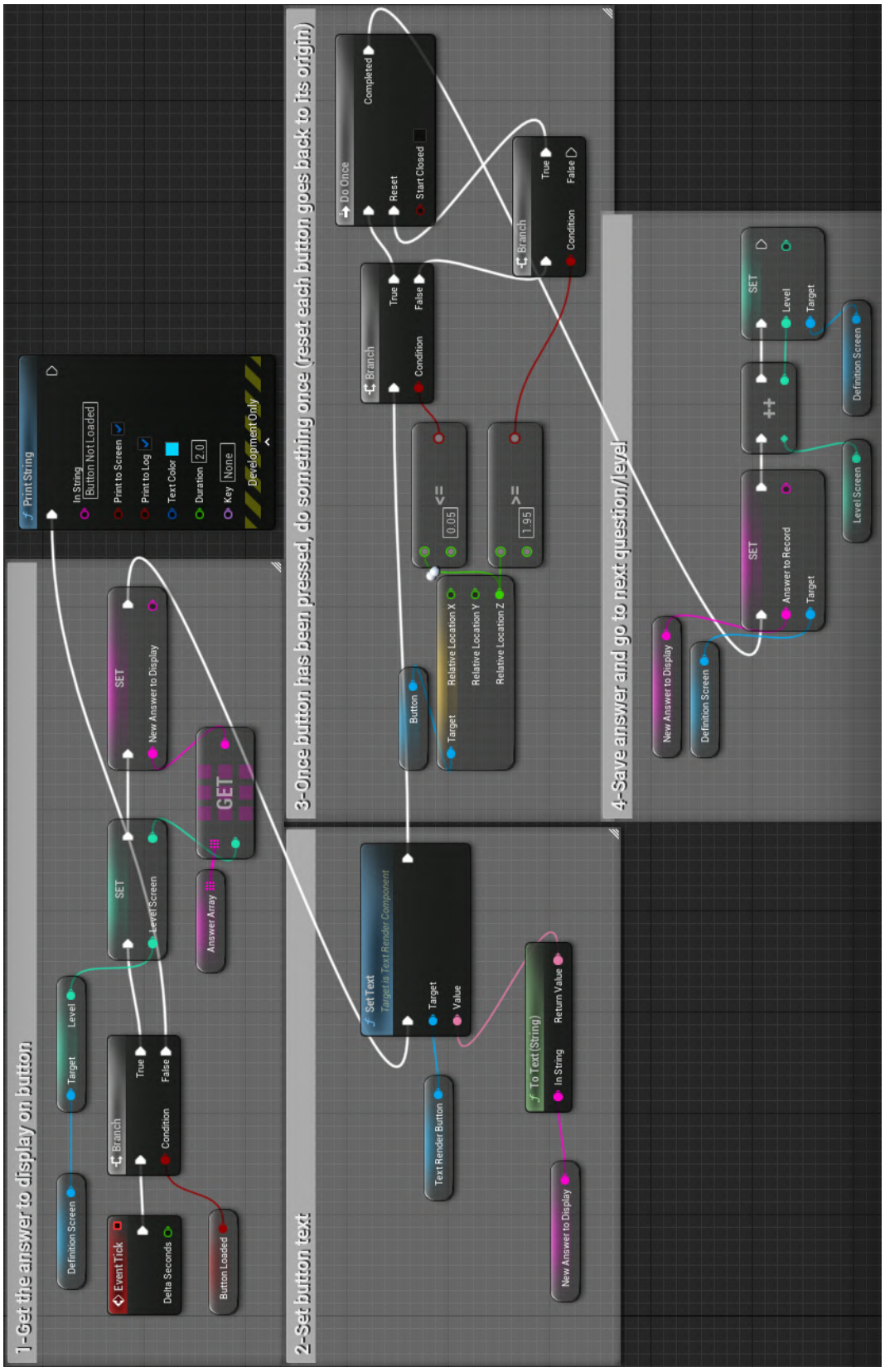
E.2 BP_Button hoverBox



E.3 BP_Button Box



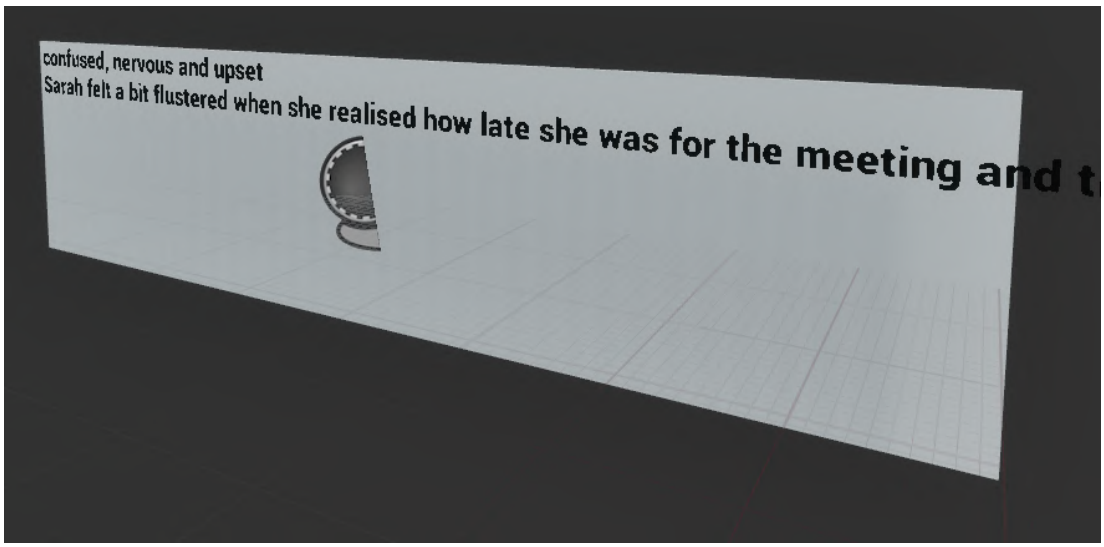
E.4 BP_Button Event Tick



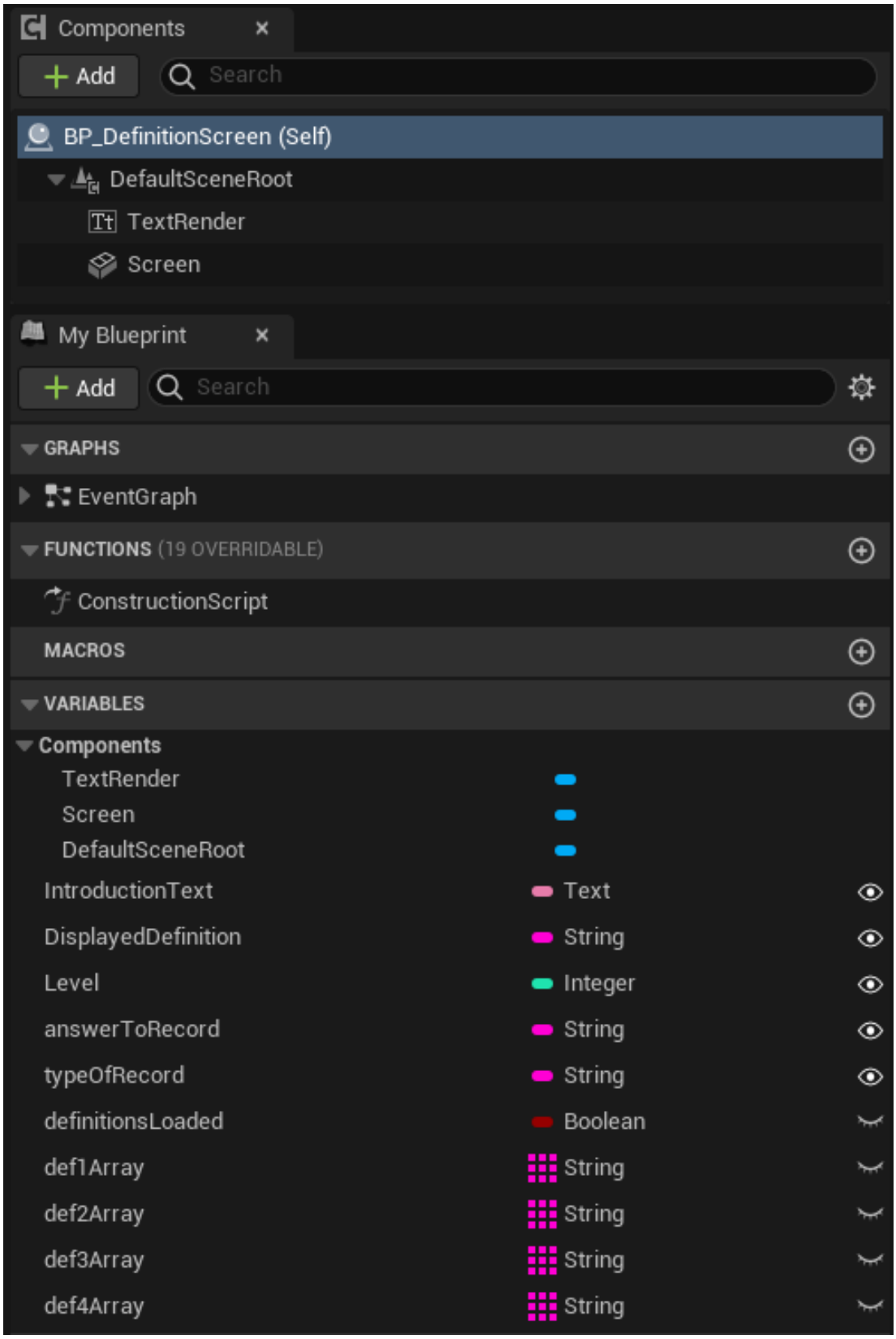
Appendix F

BP_DefinitionScreen

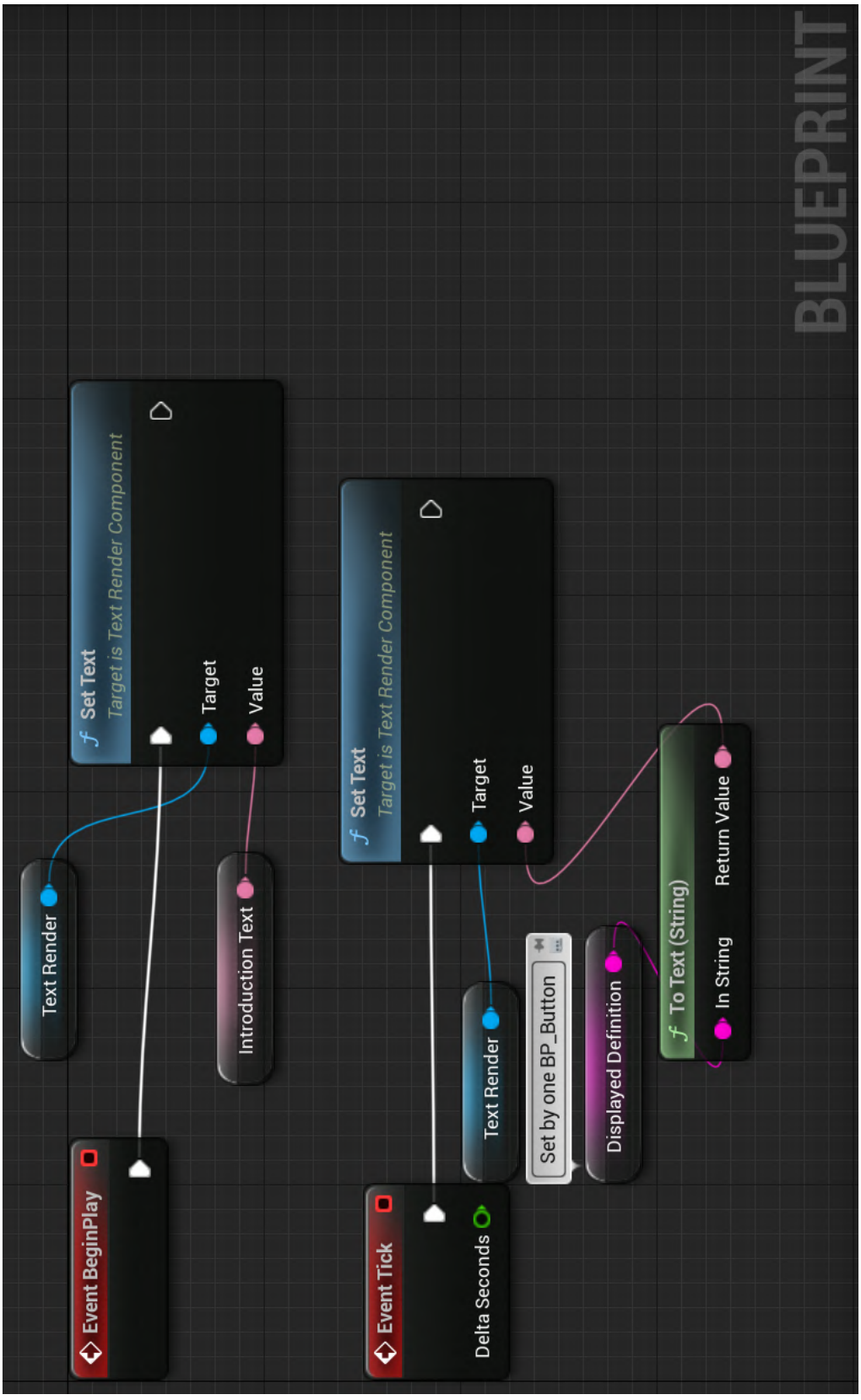
F.1 BP_DefinitionScreen 3D object



F.2 BP_DefinitionScreen Components and Variables



F.3 BP_DefinitionScreen Event Graph

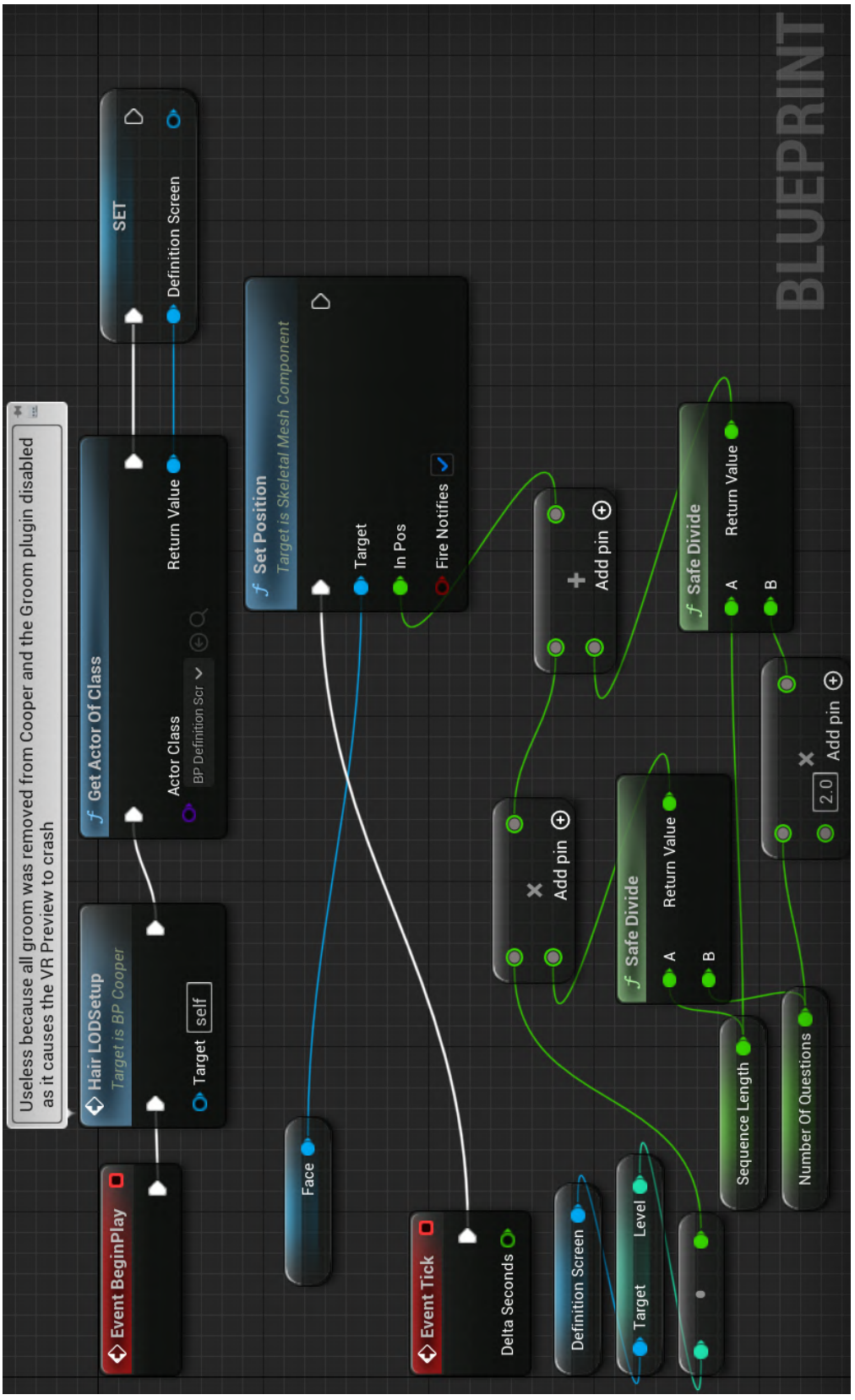


Appendix G

BP_Cooper

G.1 BP_Cooper 3D object

G.2 BP_Cooper Event Graph



Appendix H

animation.py

```
1  import os
2  import subprocess
3  import pptx
4
5  RMET_LIST=["Panicked", "Playful", "Upset", "Desire", "Insisting", "
       Worried", "Fantasizing", "Uneasy", "Despondent", "Preoccupied", "
       Cautious", "Regretful", "Sceptical", "Anticipating", "Accusing",
       "Contemplative", "Thoughtful", "Doubtful", "Decisive", "Tentative
       ", "Friendly", "Fantasizing", "Preoccupied", "Defiant", "Pensive"
       , "Interested", "Hostile", "Cautious", "Interested", "Reflective"
       , "Flirtatious", "Confident", "Serious", "Concerned", "
       Distrustful", "Nervous", "Suspicious"]
6
7  def getListJPG():
8      cwd = os.path.dirname(os.path.realpath(__file__))
9      path = cwd + '\inputPictures'
10     # Get the list of files in the pictures directory recursively
11     files = [os.path.join(dp, f) for dp, dn, filenames in os.walk(
           path) for f in filenames if os.path.splitext(f)[1] == '.jpg']
12     print("getListJPG done")
13     return files
14
15 def getRMETList(files):
16     # Filter the list of files to keep only the RMET files
17     RMETFiles = [f for f in files if any(word in f for word in
           RMET_LIST)]
18     # Put them in the RMETList order
19     RMETFiles = sorted(RMETFiles, key=lambda x: RMET_LIST.index([
           word for word in RMET_LIST if word in x][0]))
```

```
20
21      # List of all files with actor male
22      RMETFilesMale=[f for f in RMETFiles if any(word in f for word in
            ["ActorMale"])]
23      RMETFilesMale1=[f for f in RMETFilesMale if any(word in f for
            word in ["_1.jpg"])] # Frontal view
24      RMETFilesMale2=[f for f in RMETFilesMale if any(word in f for
            word in ["_2.jpg"])] # Profile view
25
26      # List of all files with actor female
27      RMETFilesFemale=[f for f in RMETFiles if any(word in f for word
            in ["ActorFemale"])]
28      RMETFilesFemale1=[f for f in RMETFilesFemale if any(word in f
            for word in ["_1.jpg"])] # Frontal view
29      RMETFilesFemale2=[f for f in RMETFilesFemale if any(word in f
            for word in ["_2.jpg"])] # Profile view
30
31      print("getRMET done")
32      return RMETFilesMale1, RMETFilesMale2, RMETFilesFemale1,
            RMETFilesFemale2
33
34  # jpeg => 60 frames of the same picture in mp4
35  def generateVideo(listOfFiles=[], actorOutput="NB"):
36      cwd = os.path.dirname(os.path.realpath(__file__))
37      path = cwd + '\outputVideos\\' + actorOutput # Output directory
38      if not os.path.exists(path):
39          os.makedirs(path)
40      for f in listOfFiles:
41          # get the name of the file without the extension
42          emotion = os.path.splitext(os.path.basename(f))[0]
43          # ffmpeg command to generate the video, replace 1st argument
                with the path to ffmpeg.exe on your computer (would be
                better on linux but you get the ideq)
44          subprocess.call(['C://Users/ysp1/AppData/Local/Microsoft/
                WinGet/Packages/Gyan.FFmpeg_Microsoft.Winget.
                Source_8wekyb3d8bbwe/ffmpeg-7.0.2-full_build/bin/ffmpeg.
                exe', '-r', '1/1', '-i', f, '-vf', 'scale=1920:1080:
                force_original_aspect_ratio=decrease', '-c:v', 'libx264',
                 '-vf', 'fps=60', '-pix_fmt', 'yuv420p', path + '\\' +
                name + '_' + actorOutput + '.mp4'])
45      print(actorOutput + "::gererateVideo done")
46
```

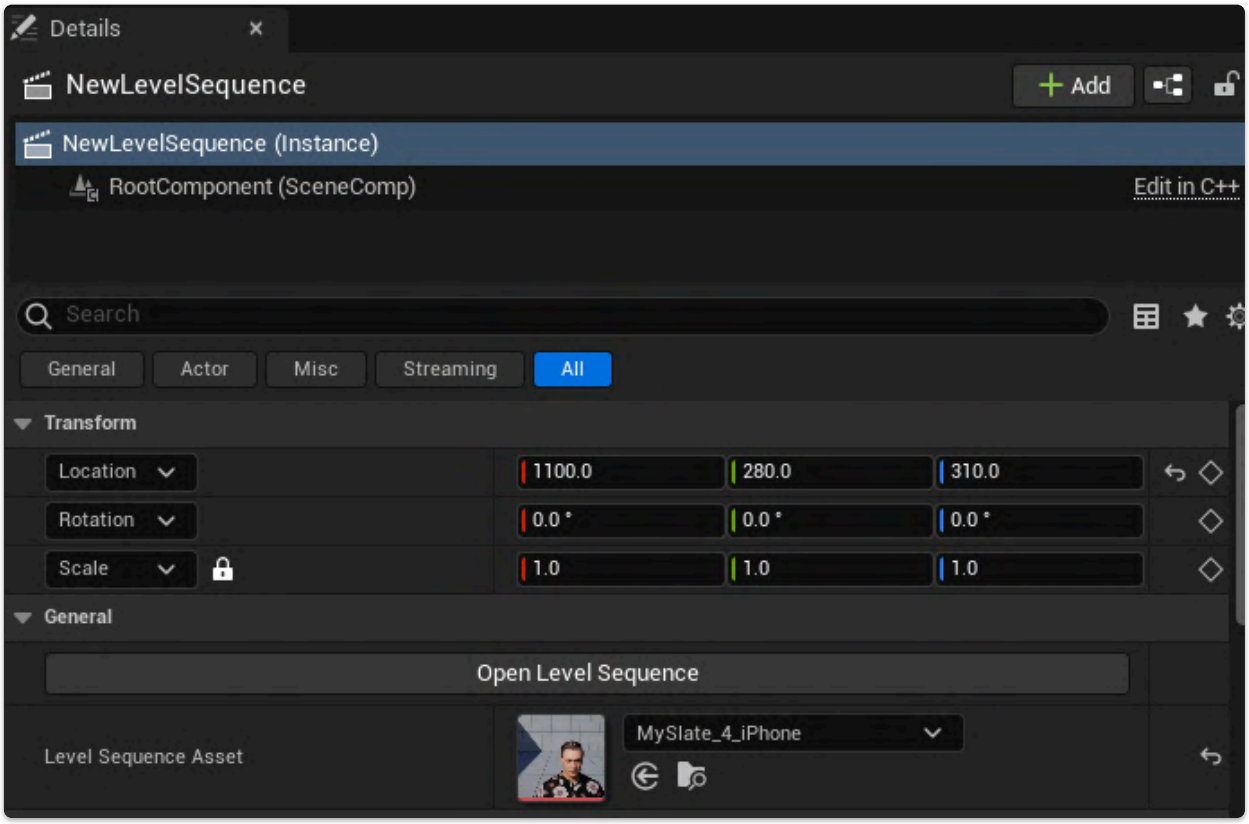
```
47 def generateSlideShow(listOfFiles=[], actorOutput="NB"):
48     cwd = os.path.dirname(os.path.realpath(__file__))
49     path = cwd + '\outputSlides\\' # Output directory
50     if not os.path.exists(path):
51         os.makedirs(path)
52     prs = pptx.Presentation()
53 
54     # 2 slides: 1st with the text, 2nd with the picture
55     for f in listOfFiles:
56         #Text
57         slide_layout = prs.slide_layouts[0]
58         emotion = os.path.splitext(os.path.basename(f))[0]
59         slide = prs.slides.add_slide(slide_layout)
60         title = slide.shapes.title
61         title.text = emotion
62         # Picture
63         slide_layout = prs.slide_layouts[6]
64         slide = prs.slides.add_slide(slide_layout)
65         slide.shapes.add_picture(f, 0, pptx.util.Pt(25), None, pptx.
               util.Pt(480)) # Don't use int a arguments but pptx.util.
               Length https://python-pptx.readthedocs.io/en/latest/api/
               util.html
66 
67     prs.save(path + '\\' + actorOutput + '.pptx')
68     print(actorOutput + "::generateSlideShow done")
69 
70 files=getListJPG()
71 RMETListMale1, RMETListMale2, RMETListFemale1, RMETListFemale2=
       getRMETList(files)
72 
73 #generateVideo(RMETListMale1, "Male_1")
74 #generateVideo(RMETListFemale1, "Female_1")
75 
76 generateSlideShow(RMETListMale1, "Male_1")
77 generateSlideShow(RMETListFemale1, "Female_1")
```

Appendix I

Create an animation asset for a MetaHuman with Sequencer in Unreal Engine 5

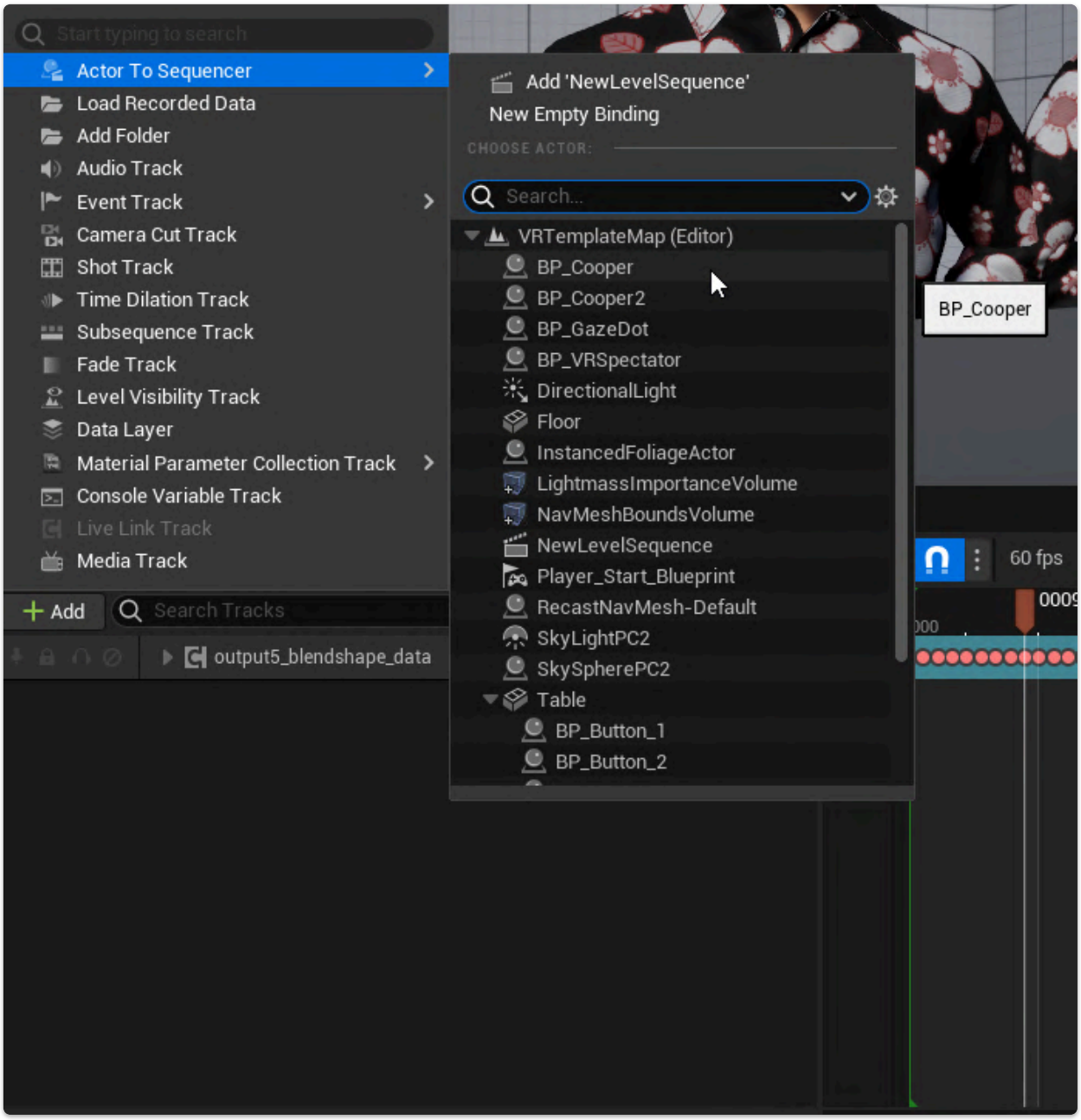
Steps :

- Install and activate LiveLinkFaceImporter plugin
- Drag and drop the generated CSV from the LiveLink Face App into the content drawer to generate a level sequence
- Select it as the Level Sequence Asset

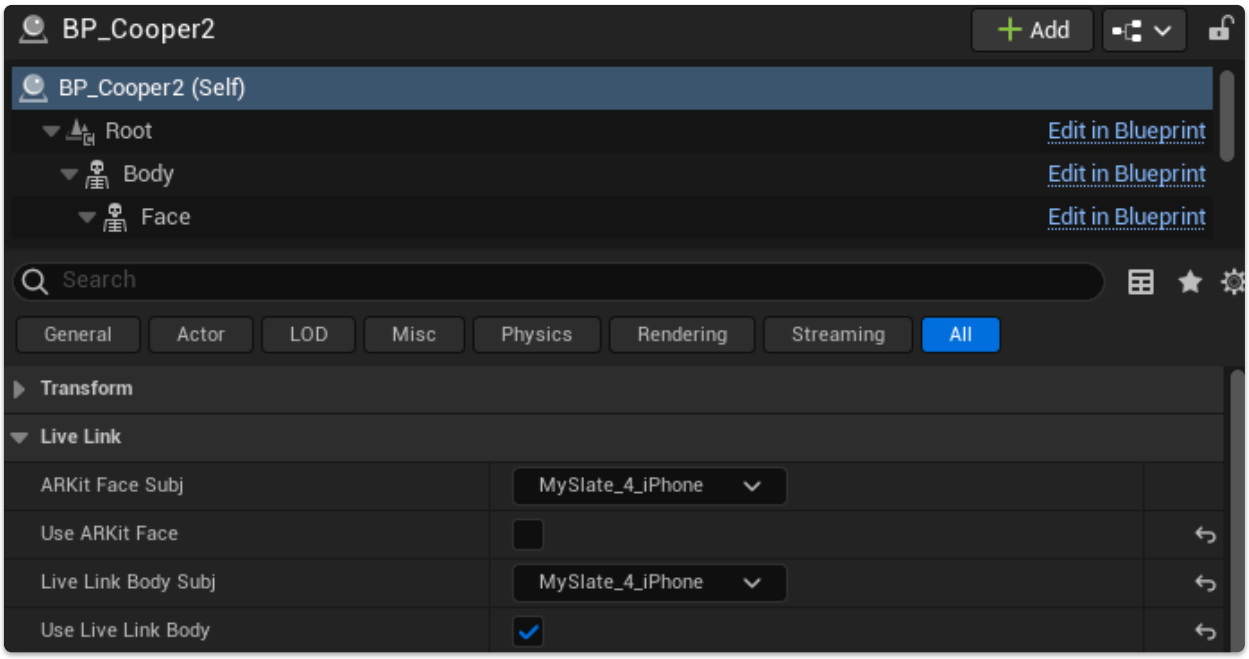


- Open Level Sequence

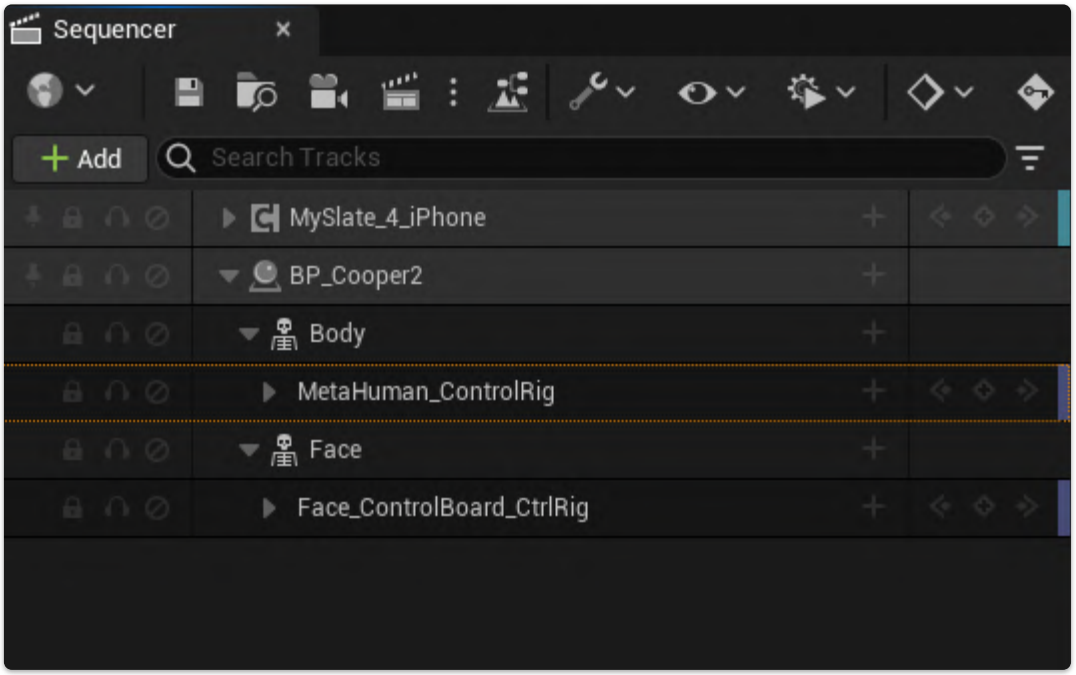
- Add your MetaHuman Actor

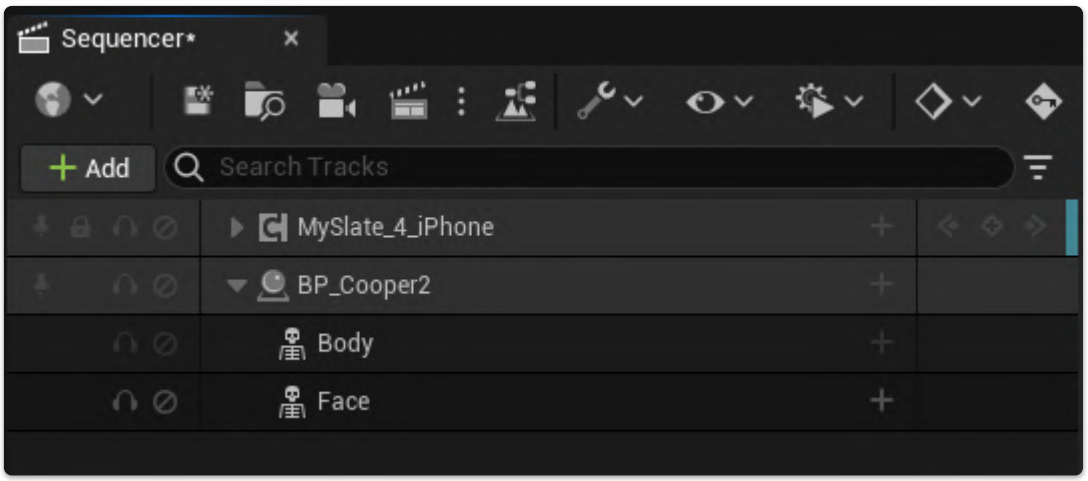


- Select your Level Sequence Asset (there should be only one) and be sure to check only 'Use Live Link Body'



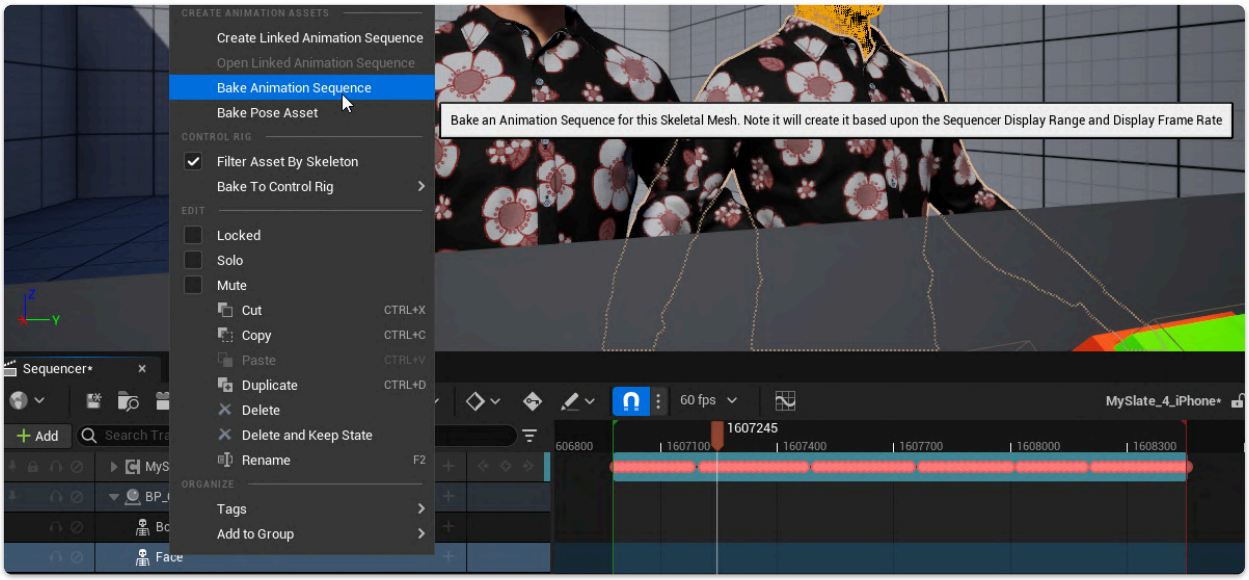
- Delete both control rig under Body and Face



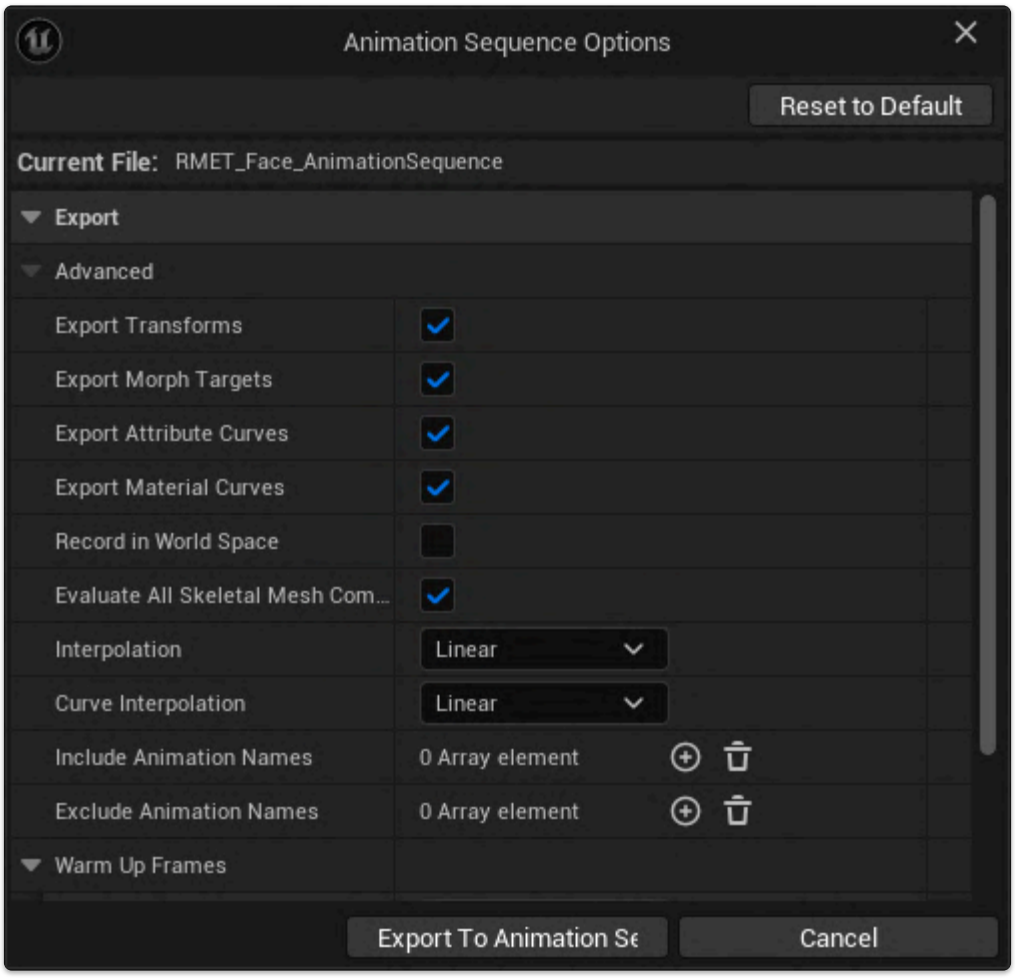


- Be sure to have Animation Mode set as 'Use Custom Mode'

- Bake an animation sequence for the Face (right click on Face in Sequencer)

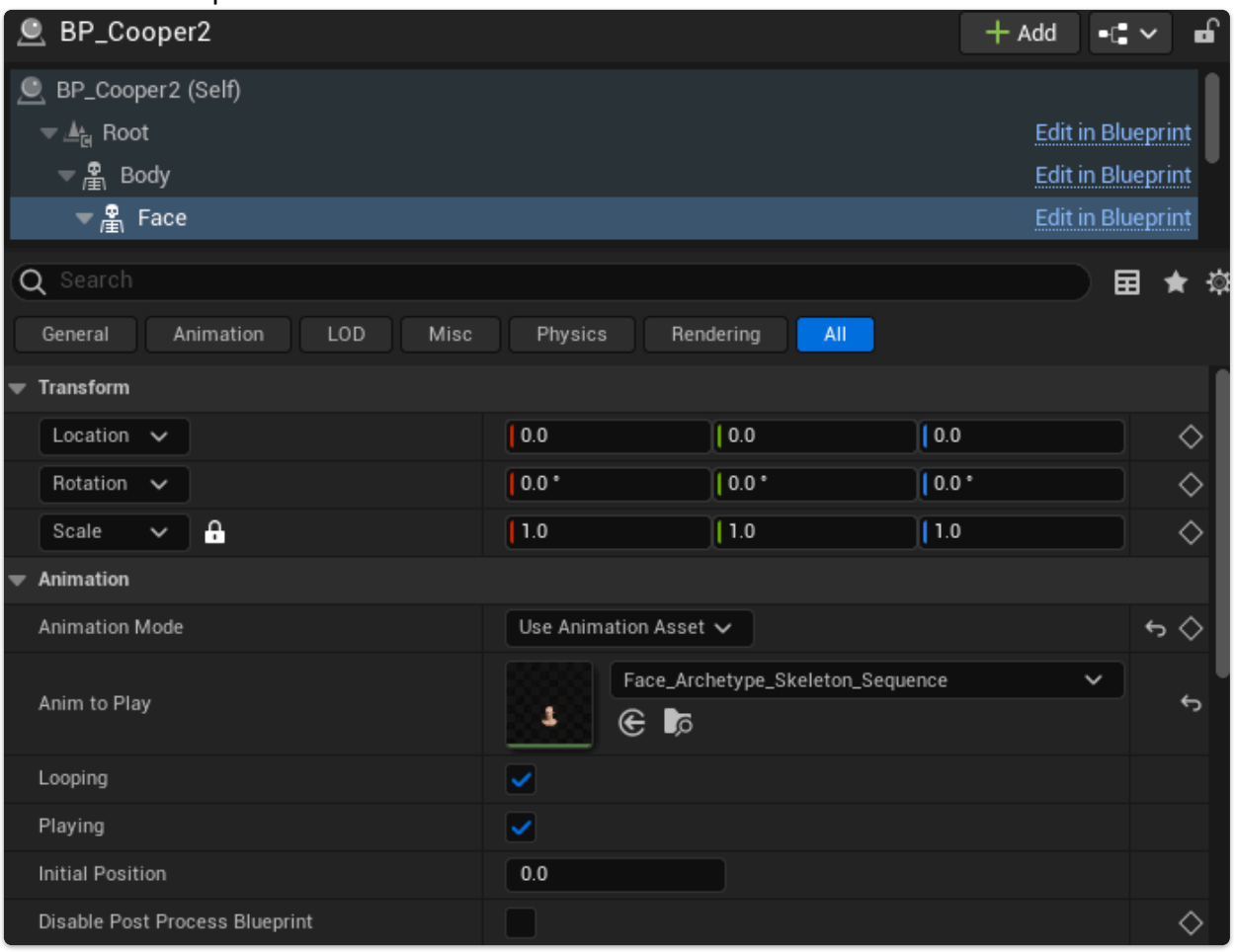


- Keep default Animation Sequences Options



- Go back to your MetaHuman actor and change the Animation Mode the Face Skeletal Mesh Component to 'Use Animation Asset' and select your newly created

Animation Sequence



- You can delete your MetaHuman Actor from the Sequencer
- If you want to change the animation of your MetaHuman, go to its blueprint and use the function Play Animation

- Other helpful functions :

Appendix J

eyeTrackingProcessing.py

```
1
2  import numpy as np
3  import pandas as pd
4  import tkinter as tk
5  from tkinter import filedialog
6  from matplotlib.backends.backend_pdf import PdfPages
7  import matplotlib.pyplot as plt
8
9  QUESTIONS = [
10     ["jealous", "panicked", "arrogant", "hateful"],
11     ["playful", "comforting", "irritated", "bored"],
12     ["terrified", "upset", "arrogant", "annoyed"],
13     ["joking", "flustered", "desire", "convinced"],
14     ["joking", "insisting", "amused", "relaxed"],
15     ["irritated", "sarcastic", "worried", "friendly"],
16     ["aghast", "fantasizing", "impatient", "alarmed"],
17     ["apologetic", "friendly", "uneasy", "dispirited"],
18     ["despondent", "relieved", "shy", "excited"],
19     ["annoyed", "hostile", "horrified", "preoccupied"],
20     ["cautious", "insisting", "bored", "aghast"],
21     ["terrified", "amused", "regretful", "flirtatious"],
22     ["indifferent", "embarrassed", "sceptical", "dispirited"],
23     ["decisive", "anticipating", "threatening", "shy"],
24     ["irritated", "disappointed", "depressed", "accusing"],
25     ["contemplative", "flustered", "encouraging", "amused"],
26     ["irritated", "thoughtful", "encouraging", "sympathetic"],
27     ["doubtful", "affectionate", "playful", "aghast"],
28     ["decisive", "amused", "aghast", "bored"],
29     ["arrogant", "grateful", "sarcastic", "tentative"],
```

```
30      ["dominant", "friendly", "guilty", "horrified"],
31      ["embarrassed", "fantasizing", "confused", "panicked"],
32      ["preoccupied", "grateful", "insisting", "imploring"],
33      ["contented", "apologetic", "defiant", "curious"],
34      ["pensive", "irritated", "excited", "hostile"],
35      ["panicked", "incredulous", "despondent", "interested"],
36      ["alarmed", "shy", "hostile", "anxious"],
37      ["joking", "cautious", "arrogant", "reassuring"],
38      ["interested", "joking", "affectionate", "contented"],
39      ["impatient", "aghast", "irritated", "reflective"],
40      ["grateful", "flirtatious", "hostile", "disappointed"],
41      ["ashamed", "confident", "joking", "dispirited"],
42      ["serious", "ashamed", "bewildered", "alarmed"],
43      ["embarrassed", "guilty", "fantasizing", "concerned"],
44      ["aghast", "baffled", "distrustful", "terrified"],
45      ["puzzled", "nervous", "insisting", "contemplative"],
46      ["ashamed", "nervous", "suspicious", "indecisive"]
47  ]
48
49  # convert to np.array
50  QUESTIONS = np.array(QUESTIONS)
51
52  CORRECT_ANSWERS = [
53      "panicked",
54      "playful",
55      "upset",
56      "desire",
57      "insisting",
58      "worried",
59      "fantasizing",
60      "uneasy",
61      "despondent",
62      "preoccupied",
63      "cautious",
64      "regretful",
65      "sceptical",
66      "anticipating",
67      "accusing",
68      "contemplative",
69      "thoughtful",
70      "doubtful",
71      "decisive",
```

```
72      "tentative",
73      "friendly",
74      "fantasizing",
75      "preoccupied",
76      "defiant",
77      "pensive",
78      "interested",
79      "hostile",
80      "cautious",
81      "interested",
82      "reflective",
83      "flirtatious",
84      "confident",
85      "serious",
86      "concerned",
87      "distrustful",
88      "nervous",
89      "suspicious"
90  ]
91
92  # convert to np.array
93  CORRECT_ANSWERS = np.array(CORRECT_ANSWERS)
94
95  def eyeTrackingProcessing(dataPath, reportAuthor="RMET-VR-Eye-
        Tracking-Processing-Software", reportPatient="Unknown"):
96      # Extracting the data from the csv file
97      data = pd.read_csv(dataPath)
98      # TODO: Add the question column to the data in Unreal
99      header = ["gazeOrigin","gazeDirection","gazeFixationPoint","
            gazeConfidenceValue","leftEyeBlink","rightEyeBlink","
            leftEyePupilDiameter","rightEyePupilDiameter","timestamp","
            cameraForwardVector","cameraOrigin","answer","question"]
100     data.columns = header
101
102     gazeOrigin = np.array(data["gazeOrigin"])
103     gazeDirection = np.array(data["gazeDirection"])
104     gazeFixationPoint = np.array(data["gazeFixationPoint"])
105     gazeConfidenceValue = np.array(data["gazeConfidenceValue"])
106     leftEyeBlink = np.array(data["leftEyeBlink"])
107     rightEyeBlink = np.array(data["rightEyeBlink"])
108     leftEyePupilDiameter = np.array(data["leftEyePupilDiameter"])
109     rightEyePupilDiameter = np.array(data["rightEyePupilDiameter"])
```

```
110     timestamp = np.array(data["timestamp"])
111     cameraOrigin = np.array(data["cameraOrigin"])
112     cameraForwardVector = np.array(data["cameraForwardVector"])
113
114     # cameraOrigin should be equal to gazeOrigin, however at a given
            frame t, cameraOrigin[t] = gazeOrigin[t-1]
115     # looking at the Blueprint in Unreal Engine, this 1 frame delay
            must be coming from the way cameraOrigin is read from another
            actor
116     # cameraForwardVector is pulled in the same way so I assume that
            it has the same delay
117     # Below is the code to correct this delay (shift the
            cameraOrigin and cameraForwardVector by 1 frame)
118     cameraOrigin = np.roll(cameraOrigin, -1)
119     cameraForwardVector = np.roll(cameraForwardVector, -1)
120
121
122     answer = np.array(data["answer"])
123     question = np.array(data["question"])
124
125
126
127     #EYE_TRACKING_START = 715
128     #DATA_PER_QUESTION = (len(answer) - EYE_TRACKING_START)//37
129     #print("Data per question: ", end="")
130     #print(DATA_PER_QUESTION)
131     # Replace [0:EYE_TRACKING_START] with 99
132     #answer[:EYE_TRACKING_START] = 99
133     #print("Answer type: ", end="")
134     #print(answer.dtype)
135     # Divide the data into 37 equal parts, each part has a number
136     #for i in range(0, 37):
137         #answer[EYE_TRACKING_START+1+i*DATA_PER_QUESTION:
                EYE_TRACKING_START+1+(i+1)*DATA_PER_QUESTION] = i
138         #print("Question n=", end=" ")
139         #print(i)
140         #print(question[EYE_TRACKING_START+1+i*DATA_PER_QUESTION:
                EYE_TRACKING_START+1+(i+1)*DATA_PER_QUESTION]) """
141     #print("Question shape: ", end="")
142     #print(answer.shape)
143     # Replace all the zeros after the last question with 99
144     #answer[EYE_TRACKING_START+1+37*DATA_PER_QUESTION:] = 99
```

```
145     #print(answer[EYE_TRACKING_START+1+37*DATA_PER_QUESTION:])
146     print(answer)
147
148     # Checking the data types
149     """ print(type(gazeOrigin[0])) # => string
150     print(type(gazeDirection[0])) # => string
151     print(type(gazeFixationPoint[0])) # => string
152     print(type(gazeConfidenceValue[0])) # => numpy.float64
153     print(type(leftEyeBlink[0])) # =>string
154     print(type(rightEyeBlink[0])) # => string
155     print(type(leftEyePupilDiameter[0])) # => numpy.float64
156     print(type(rightEyePupilDiameter[0])) # => numpy.float64
157     print(type(timestamp[0])) # => numpy.int64
158     print(type(cameraOrigin[0])) # => string
159     print(type(cameraForwardVector[0])) # => string
160     print(type(answer[0])) # => string
161     print(type(question[0])) # => numpy.int64
162     """
163
164
165     # Converting the string data to numpy array : gazeOrigin,
            gazeDirection, gazeFixationPoint => pGazeOrigin,
            pGazeDirection, pGazeFixationPoint (p stands for processed)
166
167     # gazeOrigin format: "X=0.000 Y=0.000 Z=0.000": string
168     # pGazeOrigin format : [0.000, 0.000, 0.000]: numpy.ndarray
169     pGazeOrigin = np.array([list(map(float, gazeOrigin[i].replace("X
            =", "").replace("Y=", "").replace("Z=", "").split())) for i
            in range(len(gazeOrigin))])
170     #print(type(processedGazeOrigin[0][0])) => numpy.float64
171
172     # gazeDirection format: "X=0.000 Y=0.000 Z=0.000": string
173     # pGazeDirection format : [0.000, 0.000, 0.000]: numpy.ndarray
174     pGazeDirection = np.array([list(map(float, gazeDirection[i].
            replace("X=", "").replace("Y=", "").replace("Z=", "").split()
            )) for i in range(len(gazeDirection))])
175
176     # gazeFixationPoint format: "X=0.000 Y=0.000 Z=0.000": string
177     # pGazeFixationPoint format : [0.000, 0.000, 0.000]: numpy.
            ndarray
```

```
178     pGazeFixationPoint = np.array([list(map(float, gazeFixationPoint
            [i].replace("X=", "").replace("Y=", "").replace("Z=", "").
            split())) for i in range(len(gazeFixationPoint))])
179
180     # Can't join the arrays because they have different shapes so I
            split them for each axis
181     pGazeOriginX = pGazeOrigin[:, 0]
182     pGazeOriginY = pGazeOrigin[:, 1]
183     pGazeOriginZ = pGazeOrigin[:, 2]
184
185     pGazeDirectionX = pGazeDirection[:, 0]
186     pGazeDirectionY = pGazeDirection[:, 1]
187     pGazeDirectionZ = pGazeDirection[:, 2]
188
189     pGazeFixationPointX = pGazeFixationPoint[:, 0]
190     pGazeFixationPointY = pGazeFixationPoint[:, 1]
191     pGazeFixationPointZ = pGazeFixationPoint[:, 2]
192
193     # Same operation for cameraOrigin and cameraForwardVector
194     pCameraOrigin = np.array([list(map(float, cameraOrigin[i].
            replace("X=", "").replace("Y=", "").replace("Z=", "").split()
            )) for i in range(len(cameraOrigin))])
195     pCameraForwardVector = np.array([list(map(float,
            cameraForwardVector[i].replace("X=", "").replace("Y=", "").
            replace("Z=", "").split())) for i in range(len(
            cameraForwardVector))])
196
197     pCameraOriginX = pCameraOrigin[:, 0]
198     pCameraOriginY = pCameraOrigin[:, 1]
199     pCameraOriginZ = pCameraOrigin[:, 2]
200
201     pCameraForwardVectorX = pCameraForwardVector[:, 0]
202     pCameraForwardVectorY = pCameraForwardVector[:, 1]
203     pCameraForwardVectorZ = pCameraForwardVector[:, 2]
204
205
```

```
206     pData = np.array([pGazeOriginX, pGazeOriginY, pGazeOriginZ,
            pGazeDirectionX, pGazeDirectionY, pGazeDirectionZ,
            pGazeFixationPointX, pGazeFixationPointY, pGazeFixationPointZ
            , gazeConfidenceValue, leftEyeBlink, rightEyeBlink,
            leftEyePupilDiameter, rightEyePupilDiameter, timestamp,
            pCameraForwardVectorX, pCameraForwardVectorY,
            pCameraForwardVectorZ, pCameraOriginX, pCameraOriginY,
            pCameraOriginZ, answer,question])
207     print("Shape of the data :", end="")
208     print(pData.shape)
209
210     # Data cleaning
211
212     # TypeError: ufunc 'isnan' not supported for the input types,
            and the inputs could not be safely coerced to any supported
            types according to the casting rule ''safe''
213     #pData = pData[:, ~np.isnan(pData).any(axis=0)]
214     #print("Shape after deleting all missing values : ", end="")
215     #print(pData.shape)
216
217     pData = pData[:, pData[9] >= 0.5]
218     print("Shape after deleting all rows where the confidence value
            is less than 0.5 :", end="")
219     print(pData.shape)
220
221     # left eye blink always false, hardware issue
222     #pData = pData[:, pData[10] == " false"] # Mistake in the
            blueprint, the separator is ", " instead of ","
223     #print("Shape after deleting all rows where the left eye blink
            is True : ", end="")
224     #print(pData.shape)
225
226     # right eye blink always false, hardware issue
227     #pData = pData[:, pData[11] == " false"] # Mistake in the
            blueprint, the separator is ", " instead of ","
228     #print("Shape after deleting all rows where the right eye blink
            is True : ", end="")
229     #print(pData.shape)
230
231
232     # Data stored as a list of 37 numpy arrays
233     questionsData = [np.array([]) for i in range(38)]
```

```
234     for i in range(0, 38):
235         questionsData[i] = pData[:, pData[22] == i]
236
237
238     # Get all the answers as unique values
239     answer = []
240     for i in range(0,38):
241         answer.append(questionsData[i][21][0])
242         # remove the " " at the beginning of the string
243         answer[i] = answer[i][1:]
244
245     # Eye tracking data for a given answer[i] are located in
            questionsData[i-1][21] so we need to change the whole answer
            column
246     for i in range(0,37):
247         questionsData[i][21] = np.full(questionsData[i][21].shape,
                answer[i+1])
248
249
250     # Here is the funny part. The Quest Pro doesn't give me a
            fixation point in my 3D space so I'll calculate the
            intersection
251     # between the vector V = gazeDirection  and the plane P defined
            by 4 points making a square on the same level as the
            MetaHuman face
252     # I do this for each question and plot the heatmap of the
            fixation points (ie the intersection points)
253
254
255     # for each frame, calculate the intersection point betwen a
            x_plane  (plane parralel to y and z) and the parametric
            equation of the line P defined by the gaze direction and the
            camera forward vector (V) passing by the gaze origin/camera
            origin (O)
256     # P(t) = (x0+tVx, y0+tVy, z0+tVz)
257     # then keep the y and z coordinates of the intersection point
            that are within the green square on the MetaHuman face in
            Unreal Engine
258     # point coordinates are approximated to 0.5
259     x_plane = [392.928,0,0]
260     intersectionPoints = [] # list of 37 lists of frames
261     for i in range(0,37):
```

```
262             frames=[]
263             for j in range (len(questionsData[i][0])):
264                 # Define the vector V
265                 gazeDirectionVector = questionsData[i][3:6, j]
266 
267                 V = gazeDirectionVector
268                 V = V.astype(np.float64)
269                 V=V.round(3)
270 
271                 # Get gazeOrigin
272                 gazeOriginX = (questionsData[i][0, j])
273                 gazeOriginY = (questionsData[i][1, j])
274                 gazeOriginZ = (questionsData[i][2, j])
275 
276                 t = (x_plane[0] - gazeOriginX)/V[0]
277                 y = gazeOriginY + t*V[1]
278                 z = gazeOriginZ + t*V[2]
279 
280                 intersectionPoint = np.array([x_plane[0],y,z])
281                 intersectionPoint = intersectionPoint.round(2)
282                 # round to nearest half
283                 if intersectionPoint[1]!=float('inf') and
                        intersectionPoint[2]!=float('inf') and
                        intersectionPoint[1]!=float('-inf') and
                        intersectionPoint[2]!=float('-inf'):
284                     intersectionPoint[1] = round(intersectionPoint[1]*2)
                            /2
285                     intersectionPoint[2] = round(intersectionPoint[2]*2)
                            /2
286 
287                 frames.append(intersectionPoint)
288             intersectionPoints.append(frames)
289 
290         #print("Intersection points: ", end="")
291         #print(intersectionPoints)
292 
293         # Plot the heatmap
294 
295         with PdfPages('RMET-VR-Results-'+reportPatient+'.pdf') as pdf:
296             greenSquareSideSize = 37.0
297             greenSquareLocation = [x_plane[0],0.0,161.0]
298             offset = greenSquareSideSize/2
```

```
299         topLeft = [greenSquareLocation[0], greenSquareLocation[1]-
                offset, greenSquareLocation[2]+offset]
300         topRight = [greenSquareLocation[0], greenSquareLocation[1]+
                offset, greenSquareLocation[2]+offset]
301         bottomLeft = [greenSquareLocation[0], greenSquareLocation[1]-
                offset, greenSquareLocation[2]-offset]
302         bottomRight = [greenSquareLocation[0], greenSquareLocation
                [1]+offset, greenSquareLocation[2]-offset]
303         # get a list of value [-greenSquareSideSize/2,
                greenSquareSideSize/2] for the x and y axis with a 0.5
                step
304         x_axis = np.arange(-greenSquareSideSize/2+
                greenSquareLocation[1], greenSquareSideSize/2 + 0.5+
                greenSquareLocation[1], 0.5)
305         y_axis = np.arange(-greenSquareSideSize/2+
                greenSquareLocation[2], greenSquareSideSize/2 + 0.5+
                greenSquareLocation[2], 0.5)
306
307         for i in range(0, 37):
308             x = np.array([])
309             y = np.array([])
310             for j in range(len(intersectionPoints[i])):
311                 x = np.append(x, intersectionPoints[i][j][1])
312                 y = np.append(y, intersectionPoints[i][j][2])
313
314             X,Y = np.meshgrid(x_axis, y_axis)
315
316             # count the number of points in each bin
317             Z = np.zeros_like(X)
318
319             #count everypoint that are at the same position as Z[l][
                    m]
320             for l in range(len(x_axis)):
321                 for m in range(len(y_axis)):
322                     for n in range(len(x)): # len(x) == len(y)
323                         if x[n] == x_axis[l] and y[n] == y_axis[
                                m]:
324                             Z[l][m] += 1
325
326             data=Z
327             plt.figure(figsize=(7,7), dpi=100)
328             #plt.contour(X, Y, Z, 7, linewidths = 0.5, colors = 'k')
```

```
329             #set an image background
330             plt.imshow(plt.imread("MetaHumanFace.png"), extent=[
                    bottomLeft[1],topRight[1],bottomLeft[2],topLeft[2]])
331
332             if data.size > 0:
333                 plt.imshow( data , cmap = 'jet' , interpolation = '
                        gaussian' , origin='lower', aspect='equal',
                        extent = [bottomLeft[1],topRight[1],bottomLeft
                        [2],topLeft[2]] , alpha=0.7)
334             plt.colorbar()
335             plt.xlabel("X")
336             plt.ylabel("Y")
337             if i == 0:
338                 plt.title("Control test" + "\n Answer from the
                        participant:"+answer[i+1] + "\n Correct answer:
                        "+CORRECT_ANSWERS[i] )
339             else:
340                 plt.title("Question "+str(i) + "\n Answer from the
                        participant:"+answer[i+1] + "\n Correct answer:
                        "+CORRECT_ANSWERS[i])
341             pdf.savefig()
342             plt.close()
343         # pdf metadata
344         d = pdf.infodict()
345         d['Author'] = reportAuthor
346
347 # GUI
348
349 root = tk.Tk()
350
351 root.title("RMET in VR Eye Tracking Data Processing")
352 welcome = tk.Label(root, text="Welcome to the RMET in VR Eye
        Tracking Data Processing software\nPlease follow the instructions
        below")
353 welcome.pack(pady=10)
354
355 labelPatient = tk.Label(root, text="Enter the name of the patient or
        its ID:")
356 labelPatient.pack()
357 entryPatient = tk.Entry(root)
358 entryPatient.pack(pady=10)
359
```

```
360 labelAuthor = tk.Label(root, text="Enter the name of the author of
        this report: ")
361 labelAuthor.pack()
362 entryAuthor = tk.Entry(root)
363 entryAuthor.pack(pady=10)
364
365 labelConsignes = tk.Label(root, text="Select the Eye Tracking data
        file (.csv) to generate a PDF report of the RMET in VR")
366 labelConsignes.pack(pady=10)
367
368 authorOfSoftware = "Yannis Peloutier, MScIM UCC 2024"
369 labelAuthorOfSoftware = tk.Label(root, text="Author of the software:
        "+authorOfSoftware)
370 labelAuthorOfSoftware.pack(side="bottom")
371
372 # Size of window
373 root.geometry("600x600")
374
375 def getResults():
376     filePath = filedialog.askopenfilename()
377     eyeTrackingProcessing(filePath,entryAuthor.get(),entryPatient.
            get())
378
379 buttonGetResults = tk.Button(root, text="Browse", command=getResults
        )
380 buttonGetResults.pack()
381
382 root.mainloop()
```

Appendix K

Performance logs

Appendix L

User testing results

Participant	Mental Demand	Physical Demand	Temporal demand	Performance	Effort	Frustration	I think that I would like to use this system frequently	I found the system unnecessarily complex	I thought the system was easy to use	I think that I would need the support of a technical person to be able to use this system	I found the various functions in this system were well integrated	I thought there was too much inconsistency in this system	I would imagine that most people would learn to use this system very quickly	I found the system very cumbersome to use	I felt very confident using the system	I needed to learn a lot of things before I could get going with this system	How authentic are the facial expressions and emotions displayed by the MetaHuman in front of you during the test ?
1	50	30	0	70	50	0	3	2	5	1	4	1	5	1	4	1	the eyes were a little dead inside and a lot of the facial expressions looked very similar. it also felt like cooper wasn't looking at me so that made it a little harder to tell but overall pretty authentic
2	0	0	0	90	90	0	3	1	5	1	4	1	5	2	5	1	The Facial expressions were fairly related with my daily life experiences so I haven't run into any difulties while navigating the application
3	90	10	0	40	90	10	3	1	5	1	5	4	5	1	3	1	Some smiles felt fake and some emotions / expressions felt very exaggurated and impossible to accomplish.I found it very difficult at times to relate the facial expression to the provided emotion. At times, I felt that an option appeared before the face it was supposed to be assigned to. TL;DR Not 100% Authentic. Somewhat unnatural at times
4	70	10	10	80	50	10	3	1	5	2	4	1	5	1	5	2	Pretty authentic, differences were subtle
5	70	80	40	60	40	30	4	1	5	1	5	4	5	1	5	1	bit robotic but was easily able to tell the expressions
6	60	10	0	100	50	0	1	1	5	1	5	1	5	1	5	1	Emotions that have very tell tale characteristics like smiling or squinting were easy. More subtle emotions like boredom, contemplative were more difficult
7	30	10	0	70	50	10	4	1	5	1	5	1	5	1	4	1	Pretty accurate at times.
8	50	30	0	60	50	40	5	1	5	1	5	1	5	1	5	1	The expressions were really good and easy to guess
9	30	0	0	50	60	50	5	1	5	1	4	1	5	2	4	1	The expressions seemed natural and authentic, but I often felt that the choices of emotions displayed on the buttons didn't match the expression