14/06/2026

Graduation Report

VR Simulation
Dementia-Friendly Society

Andreas Degtjarow

540076

Graduation Teacher - **Mark Boerrigter**

Alzheimervrijwilligers Supervisor - **Jan Timmers**

Table of Contents

1. Document Introduction 1
2. Project Overview 2
   2.1 Project Objectives 2
   2.2 Stakeholders 2
   2.3 Scope 3
   2.4 Deliverables 3
   2.5 Project Methodology 4
   2.6 Testing Methodology 5
3. Process Documentation 5
   3.1 Research and Approach 5
   3.2 Prototyping and Testing Iterations 7
      3.2.1 Post Processing Manager 7
      3.2.2 Physics-based VR Character & Grabbable Objects 9
      3.2.3 Localization 11
      3.2.4 UI Components 12
      3.2.5 AI Character 13
      3.2.6 Character Facial Control 15
      3.2.7 Task System 18
   3.3 Final Product 18
   3.4 Product Implementation 19
4. Recommendations 19
      4.1 Short Term 19
      4.2 Long-Term 19
5. Reflection 20
   5.1 Process Reflection 20
      5.1.1 What went well 20
      5.1.2 What did not go well 20
   5.2 Personal Reflection 21
      5.2.1 What went well 21
      5.2.2 What did not go well 21
6. References 22
7. Appendices 23
   7.1 Appendix A – Visual Representation of Iterations 23

7.1.1 Iteration 1 – Core Functionalities 23
7.1.2 Iteration 2 – Implementation for Open Days 24
7.1.3 Iteration 3 – Character Behavior & Story Implementation 26
7.1.4 Iteration 4 – Polishing & Wrap Up 28
7.2 Appendix B – UML Diagram for Old & New Project 30
7.3 Appendix C – A/B Testing Teleportation and Smooth 30
7.3.1 About them 30
7.3.2 Teleportation Results 31
7.3.3 Smooth Movement Results 32
7.3.4 UI Interactions Results & General Notes 33
7.3.5 AI Conversation Results 34
7.3.6 Character Expression Results 35
7.4 Appendix D – AI Service Cost & Performance Comparison 36
7.4.1 Diagrams for Comparison 36
7.4.2 Values examined during Speed Comparison 37
7.5 Appendix E – Test Results of Final Product Version 38
7.5.1 About them 38
7.5.2 Movement Results 39
7.5.3 UI Interactions & Guidance 40
7.5.4 AI Conversation Results 41
7.5.5 Character Expression Results 42
7.5.6 User Experience 43
7.6 Appendix F – Project Guidance Document 44
7.7 Appendix G – Contributions Showcase Video 49

1. Document Introduction

This report covers the technical development process of “VR Simulation – Dementia-Friendly Society”, which was an internship group-project during the graduation semester of Creative Media and Game Technologies. Working together with the Dutch non-profit organization Alzheimervrijwilligers as the client, this **Virtual Reality (VR)** product was to contribute to a more dementia-friendly society, by making the audience have a meeting and a conversation with a fictional family member, struggling with dementia.

The report will first establish an overview of the project, giving insights into details such as objectives, scope and methodology. It will then transition into the development phase, showing the different iterations that have been undergone, the details of feature implementations, as well as the resulting product. In the end, the report closes off with a recommendation for future development, as well as a reflection.

Being the only engineer in a group of five students, technical topics such as editor tool programming, AI integration, and VR physics implementations, are going to be the most prevalent subjects covered in the chapters to follow.

2. Project Overview

To lay the groundwork for the upcoming process chapter, this section will present the goals of the project and its deliverables, as well as who is involved. It will also include the scope of what is and what is not possible to implement in the given time frame, while also covering the methodologies applied for both project development and its testing phases.

2.1 Project Objectives

The project is a VR simulation developed inside Unity, that lets the user experience a meeting with a person with dementia. Within a friendly, realistic, and typically Dutch environment, they would get to have a conversation with them, getting to know their struggles, and feelings in day-to-day life. Next to raising awareness of dementia, the goal is to make a technically advanced simulation, which contains all the necessary features required, and can run standalone on the Meta Quest 3.

The target audience are people from all age groups, who are curious and open to using VR as a tool for social education. Whether the project was successful will be measured by having a final standalone build of the project inside the VR headset, containing features such as dynamic conversations, physics-interactions, and a fully functional game loop. By gathering feedback on the realism and feel of the interactions, it will also be possible to evaluate the amount of success of the implementation, based on how positive results are. Whether the client is satisfied with the product's technical state will naturally play a role in answering this question as well.

2.2 Stakeholders

The non-profit Dutch organization Alzheimervrijwilligers acts as the client, providing guidance and ideas for further development. They share the interest of encouraging a more dementia-friendly society, aiming to provide this VR product to any person and organization interested. Especially since the product is more technically advanced, they hope to also reach a younger audience, while still including other age groups. Though the development team hold creative freedom when it comes to the story and the interactions, these are communicated with Jan Timmers, who is the supervisor of the organization for this project. All their needs and requests need to be met, and as such, the supervisor needs to approve the implementations, to confirm them as an official part of the product. When conflicts arise, such as clashing visions or unfit interactions, it's on the developer's side to justify those decisions, before the client meets a final verdict of whether to include it or not.

The development team consists of five students, all of which are working on it as part of their bachelor's graduation internship. It is made up of two artists, two designers, and one engineer. The artists are responsible for giving the project a realistic look, by creating the environment, the character, and setting up the lighting within the simulated scenes. The designers will develop the narrative story and build the scenes, also taking care of the sounds and its implementation, to mimic a suburban Dutch household. The engineer is responsible for programming requested interactions and features, also making sure these are easy to utilize, while ensuring performance requirements for VR applications are met.

During the process, the graduation teacher Mark Boerrigter will offer academic guidance, ensuring the written report remains of high quality, and leaves no doubt in terms of documentation clarity.

2.3 Scope

The focus will lie on creating a single scene with simple interactions such as walking, picking up objects, and short interactive dialogues. The dialogue will be handled using AI integration, utilizing API keys from websites such as Groq, to receive responses, and process them within Unity. What will not be included are extensive interactions, such as exploring the house further than necessary, interacting with furniture components, and making items usable. Furthermore, the AI will only be used in a deterministic way, as a branching story is out-of-scope for this project. The project will also only be tailored for the use of a Meta Quest 3 and possibly Meta Quest 2, due to its easy switch between either hand-tracking or controllers, to show hands within the simulation. Making the project fully compatible with more devices will not be feasible.

All these constraints were set, based on the lack of experience with such elements within Unity, while also considering the given timeframe of 20 weeks.

2.4 Deliverables

The main deliverable is a standalone VR product running on the Meta Quest 3, which is suitable to be shown to the client's stakeholders. This includes fully functional systems that can be used for various implementations and configurations regarding gameplay content, coupled with research and documentation to make the process relatable. Additionally, the Unity project itself will be held in a well-documented state, in which it is able to be passed on to another group, for continued development.

To have set priorities for implementations within the project, the **MoSCoW** approach was chosen. This was done to reach the **Minimum Viable Product (MVP)**, and can be defined as follows:

Must Have

- Performance of the application must meet at least 60 **Frames Per Second (FPS)**
- Run standalone on the Meta Quest 3
- Functional and appropriate locomotion for turning and moving within VR
- Physics hands for the user within VR, including physics interactions (e.g. grabbing objects)
- Task system to be able to implement the content, including the story implementation itself
- Localization for both English and Dutch language

Should Have

- Manager with which it is possible to save and trigger different **Post-Processing (PP)** settings, based on emotions currently conveyed by the simulated dementia character
- VR interactable User Interfaces (UI)
- AI-driven dialogue
- Documentation for classes and the project setup

Could Have

- Audio- & prompt-based facial control for character (e.g. emotions, lip synchronization)
- Guidance indicators for simulation interactions during story (for beginners)

Won't Have

- Dynamically changing interactions (non-deterministic simulation)
- Explorable & extensively interactable environment

Progress presentations, as well as results from playtests or technical tests will also be added as a supporting medium, showing the client, as well as the graduation teacher, that the product has achieved its desired effect. Especially the “must have” deliveries need to be complete, as these are the groundwork for the project to be deemed as playable.

2.5 Project Methodology

For the process, a mix of double diamond and scrum will be used to be on track with the planning, while also being informed about each team members’ progress. Every task will include its own small research phase to figure out what already exists and what needs to be implemented additionally, moving on to the development phase of the solution afterwards. In detail, the first two phases of the double diamond approach will be researching what exactly needs to be implemented and how it would function on a high-level. The last two phases would consist of detailed research for implementation, as well as logic development, to be able to test and finish the feature **(*Figure 1*)**. Furthermore, daily standups in form of a shared excel sheet will strengthen team communication and awareness of current developments in other sectors. These steps will be noted down inside Trello, to have an accessible overview of all tasks and their current state.

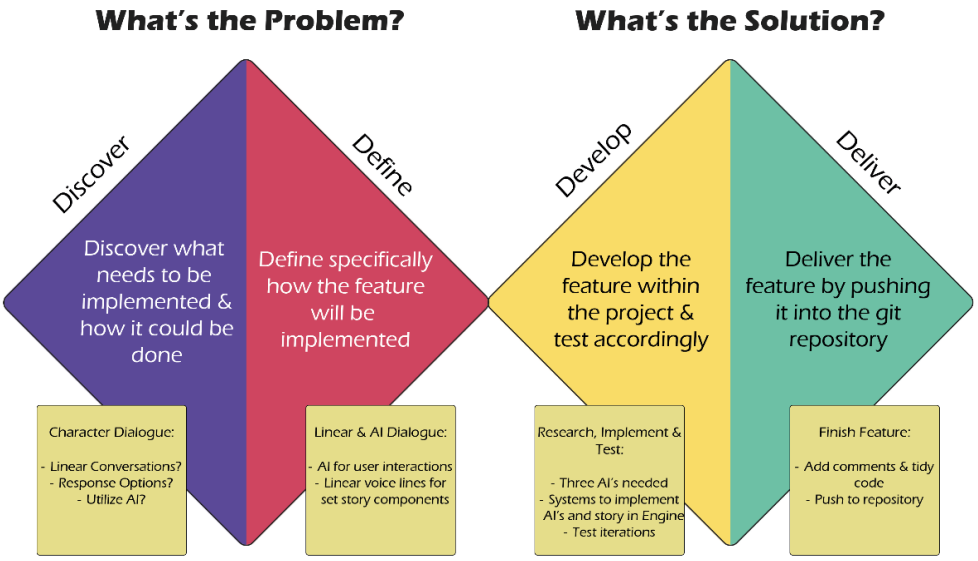


Figure 1 Double-Diamond Approach with Example Notes

Sprints will be used to indicate the current progression timeline. Every week, a new sprint starts, carrying over leftover tasks and adding new ones. This will help to organize and separate the development into their distinct sections.

Development will be made within Unity, utilizing GitHub for version control. Playtests with users will be conducted whenever a prototype version is finished, and technical tests will be done to confirm the success of new implementations. Though one playtest per sprint would be optimal, it is not a necessity, prioritizing quality of the tested prototype instead of quantity of results. Technical tests should be done throughout the development phase and documented, if results are worth mentioning to the client. Graduation circles and client meetings will also offer plenty of opportunities to gather feedback and ideas for improvement.

Combining the mentioned methodologies supports the development of a project with a tight schedule, as additions and iterations can be made quickly, without too much effort being spent on keeping the overview organized. This way, more time can be spent on the development of the project instead. As the methodologies used are flexible, transitioning into using a different system will not pose any difficulties, in case of emerging conflicts.

2.6 Testing Methodology

Two playtests will be conducted using different prototypes, with people of any age who preferably have not yet experienced the project before or have not done so in a span of at least two months. They will consist of a gameplay session, followed by a questionnaire to gather feedback about the feel, realism, intuitiveness, technical performance, and the bug robustness of the current state of the simulation.

Purely technical prototypes will be tested without any outside testers whenever possible, to save time and resources. After features have been put together, usability testing will be conducted to gain insights on how to further improve the experience, while also being able to detect bugs. During these playtests, notes will be taken by the overseeing team members, together with a questionnaire which will be given out to the participant at the end of the session. Questions about performance, motion sickness, intuitiveness and realism of the implemented features will be part of this questionnaire. Though two such playtests are the aimed for amount, more might be conducted, based on the need of the project, and the availability of new and testable interactions.

The technical results will be used to tweak values within the simulation, change the behavior of interactions, and pinpoint bugs. By measuring the results in the questionnaire, it will also become clear whether the project succeeded in creating a technically advanced application that meets expectations for a VR product.

3. Process Documentation

The following chapters cover the preparational steps and research done, as well as the development processes of the most important technical aspects of the project. Decisions will be justified and backed by data mixed with supportive images, whenever applicable. The number of iterations and their visual representations can be found in the Appendix (**Appendix A**).

3.1 Research and Approach

Since the project was worked on for one semester priorly without any engineer, the first step was to analyze the already existing code and logic structure. For this purpose, a UML diagram was created, to gain a clearer overview of the current classes, as well as their way of communication and interaction **(Appendix B)**.

During this process, it became clear that the old logic followed no clear planning and was most likely implemented using AI. The classes existed within their own space, never utilizing already existing definitions of datasets or structs. Additionally, it never applied any programming principles, to make communication more efficient and less repetitive.

An example of this would be a class, which defined seven other classes within its script **(**Error! Reference source not found.**)**. Though these were used in the context of the logic, it is not recommended to do so, as it usually shows that those classes can be declared at another place, leading to less cramped scripts.

Figure 2 Collection of classes, cramped within one script with other logic

To make the base of the project tidy and ordered, a new structure needed to be created. A UML overview was supposed to give a general direction of how the scripts needed to be set up, what exactly needed to be developed, as well as give a guideline on how to connect the logic to the overall structure. It features design patterns such as Event Bus and Scriptable Objects, efficiently reuses defined datatypes whenever needed, and holds summarized classes, which were previously defined using multiple definitions. After presenting it to the other team members and getting feedback, the result was a tidy, modular, and efficient class diagram, that essentially achieved the same features as the previous version of the project ***(Figure 3)***.

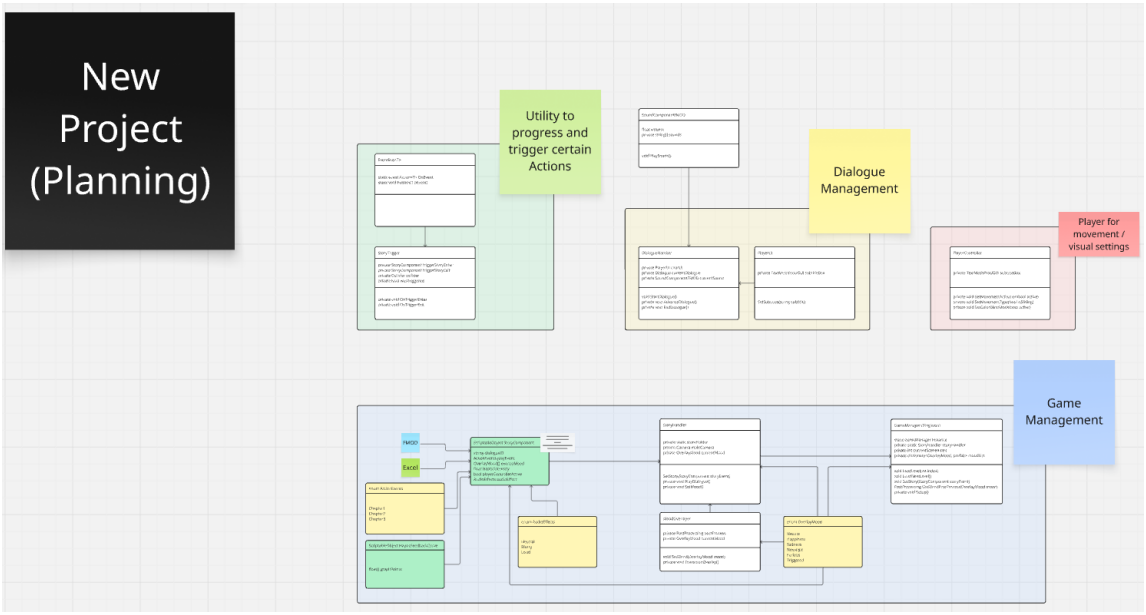


Figure 3 New project structure

After having completed the planning, the next step was to create a new Unity project, to start the development phase. Though an older Unity project already existed for this application, it was decided to create it anew, as the former Unity version had known security issues concerning potential data-leaks. This security risk wasn't a direct threat to the functionality of the application but was still better to avoid.

After getting hints from engineering teachers from the study, the default Unity VR Template was used, to have project settings optimized for VR usage. Additionally, the Meta XR All-In-One (MXR) plugin was imported, to make use of building blocks specifically tailored for the Meta Quest, simplifying development of core functionalities. With this, it was possible to add a versatile VR character containing features such as hand-tracking and customizable locomotion, grabbable scene objects, and interactable displays.

A git repository was also set up, to have version-control for the project, and make it easily accessible. From this point on, every team member worked on their own separate branches, and on their individual part of the implementations.

Further research conducted on feature implementations will be covered in detail in the following chapters.

3.2 Prototyping and Testing Iterations

3.2.1 Post Processing Manager

As the client wanted to have a product, which left a positive impact on the user's view on the topic of dementia, special attention needed to be paid to conveying the right emotions. In the previous version of the project, this was done by using a **Post Processing Manager (PPM)**, changing the visuals of the lenses by using PP. As it was the most refined feature within the old project, and as this implementation would ideally be done in the early stages of development, to give enough time for testing and applying modifications, it was decided to be developed immediately.

Different moods would be visualized by PP, changing the color of the current lens, and possibly adding different types of distortions as well. While the old system was built within a single script, holding all the settings for the moods locally, a more modular approach would result in a more intuitive, reusable, and organized system, also ensuring that risk of data loss is kept at a minimum. As such, a system was developed, which made use of **Scriptable Objects (SO)** and editor scripts, to make working with this tool as simple as possible.

The system itself consisted of four components: a prefab holding PP configurations, an SO holding a collection of PP prefabs sorted into moods, another SO to configure the percentages of individual moods active state, and the PPM itself, responsible for handling all those components ***(Figure 4)***. As PP can be very performance-heavy and is especially prone to having a negative impact within VR, due to the image being rendered twice instead of once, technical testing needed to be done to evaluate which settings can be used at little to no cost. After acquiring a list of recommended settings for URP mobile from a manual about the pipeline's details, and asking for guidance from an engineering teacher from the XR Lab within the university, the most useful and stable settings were chosen, leaving 10 out of the 18 settings recommended (Unity Technologies, Introduction to Universal Render Pipeline for advanced Unity creators, 2024). Majority of these settings were color-based and hold no additional cost in performance (Unity Technologies, Optimize for better performance, 2026).

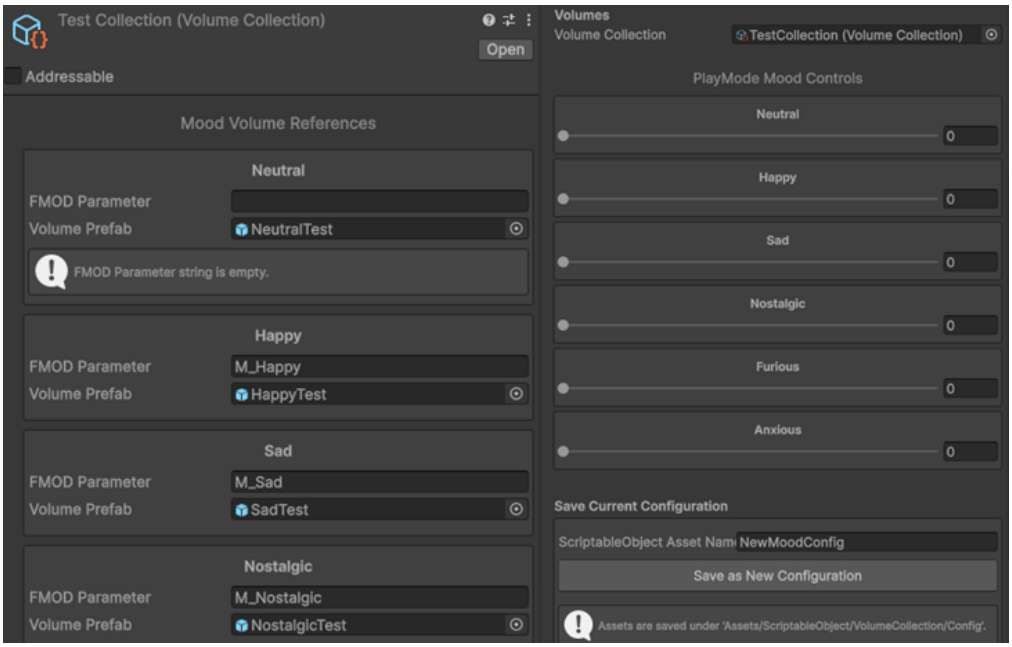


Figure 4 Left: SO Mood Mix Percentages - Right: SO Collection of Mood Prefabs

Because the project was supposed to run on the headset standalone, the application needed to be built onto the headset first, before determining the impact on performance. One version was built with PP, and the other without PP. Using the latest and fullest scene we had, the technical test showed that performance was not impacted in any way, despite having all settings active at once. As such it was concluded that the settings used within the PP prefab could be adjusted, without the risk of loss in performance (*Figure 5*).

Figure 5 Left has no PP – Right has all PP settings activated – with no difference in performance

To make the code for the PPM be more efficient and intuitive to handle, custom datatypes had to be created using SO's combined with editor scripts (***Figure 4***). These were straightforward to implement and were directly used within the PPM logic. After those preparational steps, development of the PPM could start. It needed to be able to interpret the references saved within the new datatypes, and transition between the active states. Additionally, sound logic using the FMOD system had to be developed, as the background music would be directly tied to the current mood visuals.

The most challenging part of this was scaling the weights of the PP correctly. Unity has its own priority and sorting system when having multiple active PP volumes, based on their priority and weights. With this, lower priority volumes are applied first, and higher priorities afterwards. As these are changing the same parameters, however, the effects of the lower priority volume are fully or partially overwritten by the one applied next. To make this process more intuitive for others, where setting two volumes at 100% activeness results in both being applied equally, calculations needed to be done to map the weights accordingly.

To achieve this, the goal was to always have a maximum total weight of one. For this purpose, a factor was calculated, with which all the volume percentages were multiplied, resulting in the exact weight values needed (***Figure 6***). Later, when the FMOD system was set in place for sounds, the SO's were extended to hold the background music reference connected to the mood, which were then read and triggered within the PPM.

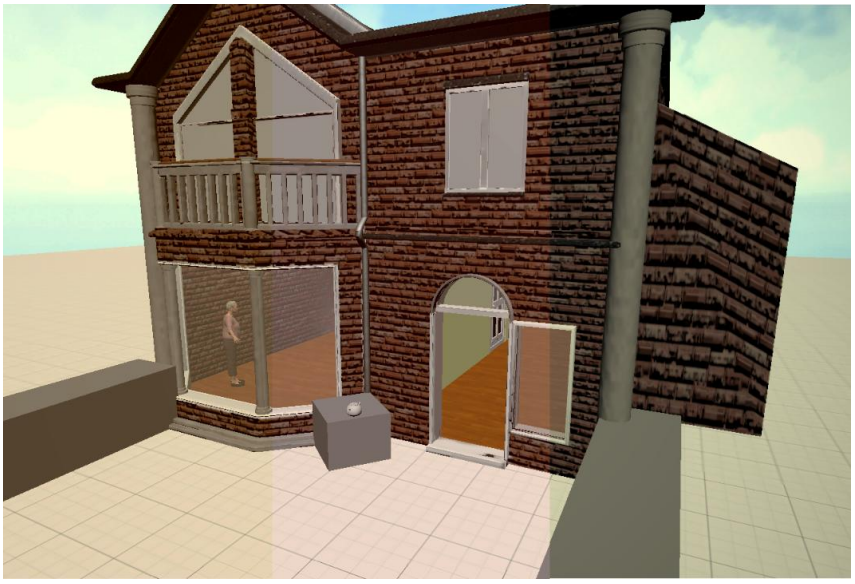


Figure 6 Mixing PP with the custom logic - Right = Nostalgic | Middle = Happy | Left = Both mixed at 100%

After distributing this tool and implementation to the designers, and receiving feedback on its ease of usage, it was concluded that it lacked the possibility of actively seeing one's changes when configuring the PP settings. As such, the manager and the editor scripts were extended, to also support applying the settings during runtime. This enabled quicker and easier iterations for the designers, when this tool.

3.2.2 Physics-based VR Character & Grabbable Objects

The client wanted to have a realistic environment within VR. Not looking at models, textures and the environment, it also meant having a character which can move, collide and interact with its surroundings. The previously imported MXR plugin already held a configurable VR Character specifically tailored for the Meta Quest 3, and as such, was taken as basis. It had features for different locomotion such as teleportation and slide movement, had responsive hand and controller models to visualize the hands within the simulation, and was able to swap between the use of controllers and hand tracking during runtime. Additionally, objects could be easily modified to be grabbable. However, this basis lacked any sort of physics support for the hands, and though a component for physics based grabbable objects existed, it only supported collisions with other objects that held a **Rigidbody (RB)**.

Because of this, multiple iterations of different physics implementations were undergone, most of which failed due to the details and complexity of the VR Character logic. For example, simply adding or modifying an RB didn't work, as hand colliders consisted of multiple capsule colliders with their own individual RBs attached. These RBs were already tailored to meet the requirements of the scripts controlling the hand movement and interactions, often causing errors when changed. The capsules themselves were created and configured dynamically using a script, which also added another layer of complexity, when testing different approaches. Analyzing and adjusting this complex structure was not feasible in such short time, which is why a proxy approach was taken instead, utilizing the existing implementations, instead of inventing new ones.

In this approach, a custom physics hand follows the movement of the default hand, essentially having both physics and non-physics versions implemented for the character. The default hands would be made invisible but still active, to only show a single pair to the user. In case the physics hands shown reach a certain distance from the real hands position, for example when blocked by a surface, the default hands will be shown additionally, to give the user a sense of orientation. This is also a well-established way of implementation, as this approach was also present in the development of physics hands for popular VR games such as Half-Life: Alyx (HalfLifeAlyxTeam, 2020).

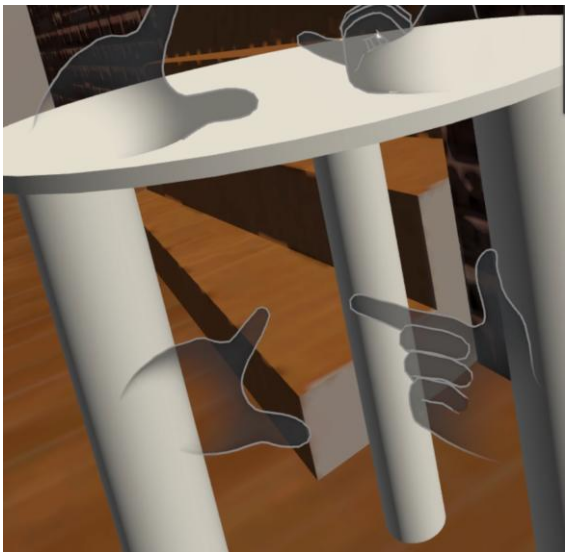


Figure 7 Physics hands are blocked by the table, and beneath, the real hands are shown

The first step to implement this was to create the proper colliders for the hands, with correct settings for physics interactions. The MXR plugin had a separate script called "HandPhysicsCapsules" for instantiating such colliders, and by creating a new script with the same logic, leaving out the pieces tied to RBs, it was ready to be used for the mentioned approach. To mimic the movement of the actual hand, the final position of the invisible default hand was applied as velocity, using a custom script. This made it possible to account for any possible collisions that would happen throughout the hands motion. The physics hand itself held an RB, where collision detection was set to "Continuous Dynamic", to also cover hectic movement, as well as to "Interpolate", to make the motions smoother. Additionally, the solver iterations for collision detection and resolving were increased inside the physics hand script, to improve accuracy. Doing so within the project settings would have led to an increased performance usage for all physics objects, and as such, was only done from within the physics hands script itself.

As this physics movement is applied using FixedUpdate, unlike the default hand movement which happens inside the Update function, simply applying the velocity towards the current position may lead to

noticeable delays in the hand's movement, when making the player move at the same time. This is due to Update being tied to the frames per second, unlike FixedUpdate which has a constant interval instead, leading to an asynchronous resolving of actions (Unity Technologies, Fixed Updates, 2026). To reduce the stuttering, the players' movement velocity or end position after teleportation was used for prediction, when calculating and applying the necessary velocity for the hand motion.

The same proxy-follow approach and RB configurations were applied to the grabbable physics objects, which were placed in the scene for interactions and tasks. As these would not be in use as often as the hands, their collision settings for the RB were left to be "Discrete", and the collision detection and resolving iterations were left the same. Setting it to "Interpolate" was important still however, as the objects had to move smoothly as well.

Being able to switch between different types of locomotion was also worked on, as users might react differently in terms of motion sickness, based on the way they move in the simulation. As the base character lacked any default way to achieve this, further investigation of the building blocks and sample scenes was done. By importing and analyzing the sample scene about locomotion using the MXR plugin, it was possible to pinpoint the exact component within the character hierarchy, responsible for the characters' locomotion. By toggling their active states based on if they were needed, the movement type was able to be switched within runtime, to either teleportation, slide movement, smooth turn, and snap turn. After user testing, however, this feature was in doubt, as most participants felt motion sick after using the slide movement. As those who did not experience motion sickness also did not hold any preference in the type of locomotion, teleportation and snap rotation were temporarily set as default, and options to tweak this were removed, to keep the number of visible settings at a minimum. This decision would later be reverted, based on the results of an A/B testing, where one version only allowed teleportation, and another only slide movement (**Appendix C**).

Additionally, a first-person character was implemented, which was able to be controlled using normal keyboard and mouse, to allow easier application testing. The character was equipped with the usual movement, as well as a physics interactor, with which it was possible to pick up and drag around an item. This approach has significantly helped reduce time for testing certain implementations, as VR headsets are quick to discharge, and putting them on for each small addition was time-consuming.

A/B Testing: Teleportation and Smooth Movement

Though the testing session during open days indicated that most people would prefer to have teleportation for movement, due to increased chances of motion sickness, one more A/B testing was conducted, to have more visual replies to make such decision for locomotion **(Appendix C)**. Results were surprising, as they gave an impression of there being not much of a difference between using either teleportation or smooth movement. Though the teleportation variant had slightly better replies, it was not in such high amounts, where excluding a movement type would have been reasonable. As such, it was decided to be added once more to the available options for the user.

3.2.3 Localization

Since the client wanted to have the product be available in both English and Dutch, as it was supposed to be offered mainly in the Netherlands at first, but also potentially internationally, setting up a localization system early into the project was essential. Unity had a well-established plugin for this called "Localization", which provided all functionalities needed to easily switch sentences into different languages. By creating tables within their system, UI-components were easily able to be translated, by additionally adding a component of type "LocalizeEvent" and attaching the fitting key value **(*Figure 8*)**.

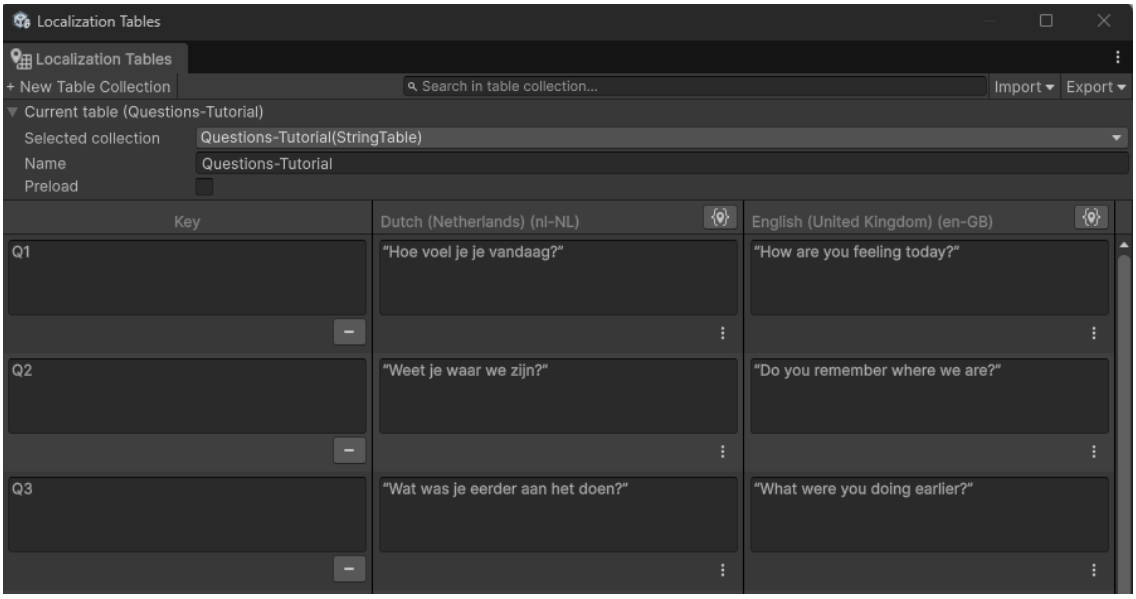


Figure 8 Localization Table with keys, with Dutch and English as available language

A Localization Manager script was also created, to be able to change these settings within runtime. This was important to have, as the application was supposed to run constantly, for various users with different language expertise to experience. Within the logic, only the selected language within the "LocalizationSettings" class had to be changed, as the engine and plugin already took care of replacing all the objects, holding the mentioned components. Additional configurations done for localization to other features such as AI, will be covered in their own sections.

3.2.4 UI Components

As the client wanted to have a user-friendly product, which was easy to understand for beginners as well, proper UI needed to be implemented, together with intuitive controls.

The MXR plugin offered a lot of different components for UI interactions, and by using a prefab called "FlatUnityCanvas", a technical basis was already set, with which it was possible to change the style and functionalities using Unity components. Doing this instead of setting up an object from scratch was more time efficient and safe, as finding and configuring components was as complex as the VR character previously. With this, it was possible to use the index finger to press and activate the UI buttons. Though hands with ray casts were also available and easily configurable, this feature was mostly removed, as internal technical testing has shown that buttons would function inconsistently within the build version.

An interactive menu was also developed, which made it possible to open the UI, by making the palm of the hand face the user's camera. This was achieved by mapping the current rotation to a value between zero and one, setting the visibility of the canvas, and toggling the active state of the interactive components **(*Figure 9*)**.

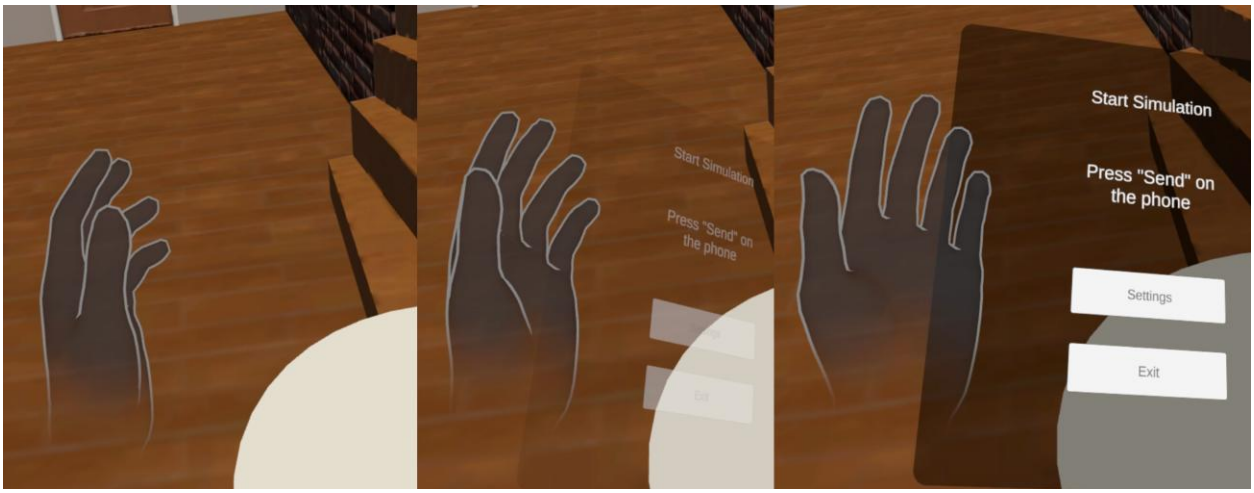


Figure 9 The handheld menu becoming more visible, the more the hand turns towards the face

During the A/B testing for locomotion, Dutch participants were additionally encouraged to use the handheld menu, to switch the localization in the simulation. Results showed that it was easy to switch the language using the UI, and as such, the implementation was marked as a finished **(Appendix C)**.

Only changes made to different iterations were the size and position of the handheld menu. Putting it on the center of the hand turned out to be irritating, as the hand itself would have to stretch uncomfortably, to be able to read and interact with the menu. Having it smaller was also not an option, as the lenses of the Meta Quest 3 proved to be blurry when reading smaller text **(Appendix C)**.

3.2.5 AI Character

The main feature the client wanted to have was a character which was able to talk about and explain what it feels like living with dementia. As such, some form of dialogue needed to be implemented.

Though the project was initially planned to have linear dialogues between the character with dementia and the user, a suggestion from the head of XR Lab mentioned that this feature might be better powered by AI instead, to be able to hold dynamic conversations in real time. This featured many obstacles though, as usage of AI in development environments is often tied to costs, and many different providers exist, all with their own advantages and disadvantages. Additionally, holding a conversation required three AI's, all responsible for their own part of the dialogue: **Speech-To-Text (STT)** to pick up what the user says, a **Large-Language-Model (LLM)** to create a fitting response to the prompt, and **Text-To-Speech (TTS)** to make the character respond with a voice.

Choosing a service

Following an AI- and VR-experienced teacher's advice, to use OpenAI to integrate this feature into our project, this provider was picked for the first iteration. Since it offered a deal, with which it was possible to use the LLM and STT with either no or little costs for low-scale development, it was also favorable from a financial point of view. Analyzing available services for TTS, Inworld stood out as it offered language support for both English and Dutch, had realistic intonation with adjustable emotions, and was paired to comparably low costs. Additionally, it offered a free service with which it was possible to create an AI voice, using recordings of a real person's speech. This would be useful for the simulation, as the client requested a very specific accent for the character.

Acquiring an API key from both OpenAI and Inworld was essential, to be able to use the AI models. However, directly implementing this key into the application would make the service account prone to misuse, as the key could be obtained by reverse-engineering the simulation, and be potentially used for unintended purposes. As such, this key was not directly embedded into the code of the simulation, but instead saved locally onto the device, and acquired by code when needed.

Speech-To-Text & Large-Language-Model

To implement STT, the first step was to start recording audio bytes, using the prioritized microphone by the current system. Since the project used FMOD for all the sounds, getting the microphone using the default Unity functions was not an option, as mixing both systems can be the cause of potential loss in performance. As such, FMOD functions were used instead to start and stop the recording, and configurations were applied to set the maximum amount of time available, for formulating a question or a response for example. To implement the AI, a tutorial was followed which used a plugin called “OpenAI-Unity”, that supported the usage of most features offered by OpenAI, including TTS and LLM (Altundas, 2023). As such getting a transcript of the audio created could be done by utilizing the plugin function “CreateAudioTranscriptionsRequest”, only needing to additionally create a function, with which it was possible to convert the audio bytes into a “.wav” file. Putting it all together resulted in the transcription of what was said, making it possible to get a response from the LLM.

Doing so could be done quickly, by using the plugin function “CreateChatCompletionRequest”. To prepare the AI to be able to roleplay as the character that was needed for the simulation, a custom initial prompt needed to be created, tailored to her backstory, habits, and containing safety checks to prevent undesired answers. This task was given to the designers, and the result was then used for further development. Furthermore, to mitigate unintended loss of memory during conversations, the initial prompt and previously sent messages would be attached to every message sent to the LLM. In the end, the AI would return a text, which could then be used for TTS.

Text-To-Speech

As no plugin existed with which it was possible to get TTS requests from Inworld, a custom script needed to be developed, to access the service from within the project. Taking an example of the class OpenAIAPI” from the OpenAI plugin, the “UnityWebRequest” library was used to establish a connection to the service, replacing details with the code provided by Inworld’s API documentation, and translating it into Unity C# code (Inworld, 2026).

To make it also function properly with the intended emotions, however, preparational steps needed to be taken for the initial prompt of the LLM, as the TTS needed very specific tags to use different emotions. Furthermore, each TTS request was only able to use a single emotion, which meant that a text with varying tones in between needed to be put together with multiple individual TTS results. To minimize latency when waiting for the requests and prevent freezing frames, the individual requests were made to be sent out simultaneously, and put together within a byte array, using a background thread.

After converting the resulting bytes into a “.wav” file, the next step was to play a dynamically changing sound from within FMOD. For this, a programmer sound needed to be added to the FMOD system, which would then be used and changed from within the script. With this final addition, the AI dialogue feature could be put together, and be tested for its functionality, after which it was completed.

Switching OpenAI STT & LLM to Groq

Following a showcase of the project done during open-days, vocal responses of visitors implicated that the latency between the user talking and the character responding considerably harmed the realism of the experience. This prompted further research into what was causing the latency, and how this could be

mitigated. By printing out the time between every step of the AI dialogues chain (***Figure 10***), it became clear that the LLM had a significantly higher latency, compared to the other STT and TTS **(Appendix D)**.

Figure 10 Example Debug Log used to analyse latency of every step of the AI logic

After looking into alternatives for OpenAI, the provider Groq stood out, as it specialized in low-latency services, paired with noticeable free-tier offers, and as such, was tested as well **(Appendix D)**. Implementation was quick to do, as their API was built the same way as OpenAI, leading to only a few references needing to be changed. The same technical tests used for testing the OpenAI version showed a decrease in latency of more than half of the initial time, leading to more fluent conversations **(Appendix D)**.

Connecting AI to localization

Since the logic controlling this interaction was custom made, localization had to be applied manually within the script, upon detecting a change in settings. Specifically, this meant adjusting the voice model and STT recognition, as well as adapting the initial prompt for the LLM, to reflect the language change in their response. By subscribing to the “HandleLocalizationChanged” event from the LocalizationSettings class, all mentioned changes were able to be applied when necessary, using a custom function to replace all string entries as necessary.

Dialogue Testing

To see how realistic and natural the flow of the conversation felt, additional steps have been added for the A/B testing. Though results were mostly neutral or positive, some participants experienced it quite negatively **(Appendix C)**. The quality of the AI’s response would vary a lot, where for some it was very talkative and interactive, while for others it was responding to unrelated things, repeating itself, and sometimes saying incomprehensible words. Slight adjustments to the initial prompt and code were made, to potentially take care of some of the issues, but the inconsistent nature of the problem indicated that it was most likely related to the AI model itself. To this end, there was nothing that could have been done during the time span of the project, and as such, it will be mentioned once more in the recommendation section of this report.

3.2.6 Character Facial Control

As the client wanted to have a realistic-looking product together with a person with dementia, with whom you could have a conversation, the character needed to have realistic motions. Utilizing the components created by the character artist, this specifically meant lip synchronization, eye movement, and emotions through their facial expressions, to make the character feel more alive.

Lip Synchronization

To make the character move their lips during the conversations, visemes were added to the model by the character artist, which were able to be controlled using blend shapes. As audio bytes were generated dynamically using AI, the lip synchronization had to be able to adjust based on what was said. As such, the “Oculus Lipsync Unity” package was imported, which provided classes and functions for this specific

purpose. Though the package was deprecated, it still provided logic which could be used within custom scripts, while also offering plenty of documentation and guides (Meta Platforms, 2021).

By taking an example of the “OVRLipSyncContext” class and the “ProcessAudioSamplesRaw” function provided by the package, a clear vision was set, with which it was possible to convert raw audio samples into viseme data. Special attention needed to be paid to converting the audio bytes into the appropriate structure though, as the function itself only took audio data of type short. Since a short can hold twice as much as a byte, the first step was to divide the amount of total available entries of the initial audio format by two, to not use more memory than necessary. After copying the data into the appropriate structure, the audio samples were processed in steps of 1024 frames, which was approximately equal to 0.02 seconds of audio played. Each frame was then evaluated using the ”ProcessFrame” function provided by the package, to know the exact percentages in which different visemes needed to be set.

Setting the blend shapes according to the visemes was straightforward, as this only required the reference of the Skinned Mesh Renderer holding the sliders, and mapped indexes according to the default order of visemes ***(Figure 11)***. By reading the current time frame in which the audio is currently playing from within FMOD, the exact viseme data was able to be extracted, multiplied by a hundred to fit the system within Unity, and applied to the blend shapes accordingly.

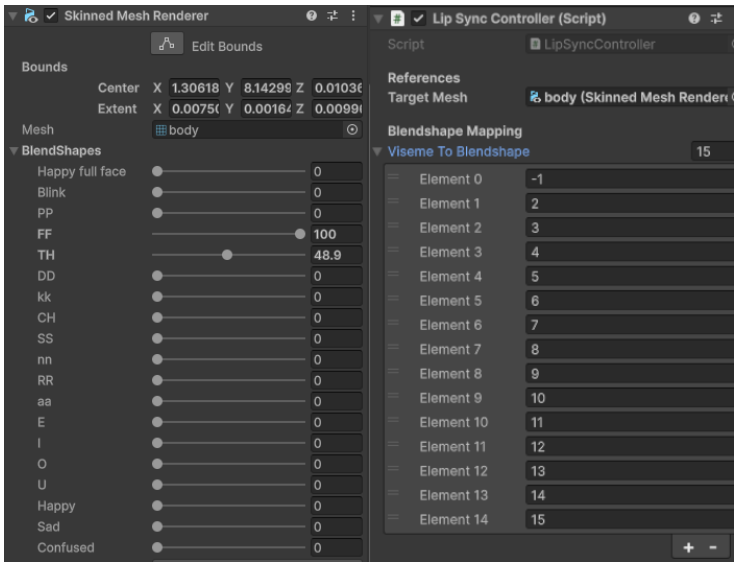


Figure 11 Left: Blendshapes for visemes, emotions, and blinking | Right: Blendshapes mapped accordingly

When running internal testing with the team, including the character artist, it was noticeable that the lips never looked the way they should with the letter currently pronounced. Following the hint of the artist, that the visemes needed to hit the value of 100, to show the exact letter, and that only one viseme should be active at the same time, the implementation was adjusted to reflect this exact logic. After another internal testing, however, the results turned out worse than in the previous iteration. With the guidance of an artist teacher, it was concluded that the visemes should be left as they were previously, and that having multiple visemes active was the desired outcome. As such, the first iteration was reverted to once more, and after changing the shapes of the visemes themselves, the quality of the implementation improved well enough, to be able to move onto the next feature.

Facial Expressions

Next to the blend shapes for lip synchronization, the character also contained emotional expressions, that could be controlled in the same manner. Since differentiating between emotions within certain sections of

the AI's response was already implemented for the use of TTS, the existing structure was utilized to additionally change the expressions of the character. By writing a new script, which specialized in applying emotional expressions, based on the mood currently used during the speech, this script was finalized. Additionally, the logic that was created to manage moods on a visual level using PP was also connected, to tie this system to the expressions the character is showing.

To further increase realism, another script was created, which specialized in controlling the eye movement of the character. It was responsible for making the character blink using randomized intervals, and switching between looking at the user, and purely looking down.

Since blinking was controlled using blend shapes once again, a transition needed to be created. To make it realistic, however, it needed to be separated into three different sections with their own intervals, namely closing, closed, and opening. Implementation was straight forward, and by doing internal testing to achieve fitting intervals for the eye pauses in milliseconds, the character became more alive. Afterwards, the script was extended, to also hold the logic to occasionally rotate both left and right eye towards the user's face. By calculating the look rotation, multiplying it with the initial 90° rotation existing on the model, as well as with the inverse to make the end rotation be in the local space of the character, the logic was finalized and ready to be used **(*Figure 12*)**.

The same timer-based logic was also used to make the character breath, utilizing another blend shape.

Figure 12 Character looking at the user from the side with a happy expression

Testing Character Impressions

Similar to the AI voice interaction, questions about the characters expressions were asked, to get insights into the quality of the eye-following and blinking implementation, as well as the lip synchronization during the dialogue **(Appendix C)**. Looking at the feedback, the lip synchronization seemed to be the most consistent with their positive responses, while the eyes and the expression were mixed. Some experienced the character looking into one's eyes as "creepy and unnatural", while others did not seem too bothered by it. In terms of expression, most of the feedback was unrelated to the speed at which it was transitioning between the emotions, and as such, was out of range for improvements in terms of programming. Though the behavior of the eyes could be improved by researching into typical eye movement during conversations and translating it into logic, time was short and results were good enough for now. As such, it will be mentioned in the recommendation section to come.

3.2.7 Task System

As the client wanted the product to be quick to experience, the story and guidance planned by the designers needed to be implemented. For this purpose, a base class for tasks was created, together with a Task Manager responsible for their order of execution. The tasks themselves were equipped with a description of what was supposed to be done, displayed on the handheld menu during runtime, as well as all necessary references needed for this specific section of the simulation (*Figure 13*). By implementing the logic needed to trigger the systems previously covered, next to playing animations and certain phrases for the character, the content for this product was able to be developed and finalized.

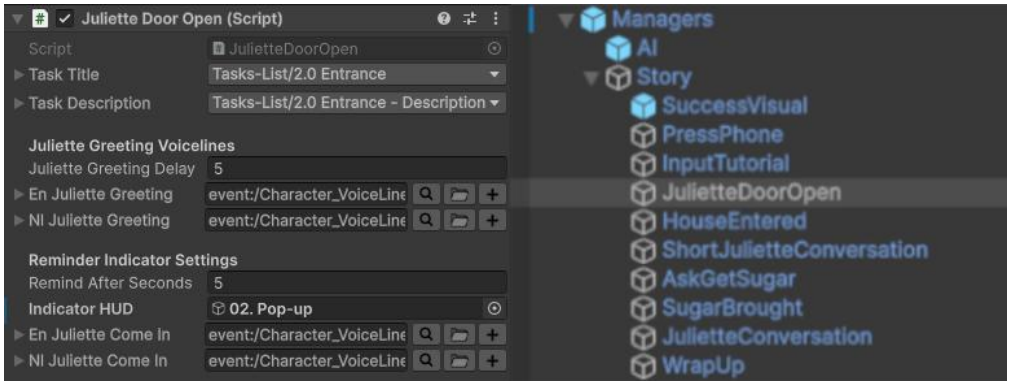


Figure 13 Left: Inspector with references to localization and phrases | Right: Task Components

Test results during the final week of the project have shown that users were mostly satisfied with this story implementation and progression (Appendix E). Though overall satisfaction with the simulations experience can be improved still, the interactions themselves were already intuitive, as can be seen on the high average regarding that question.

3.3 Final Product

The final product is a fully playable, standalone application built on the Meta Quest 3. It holds one scene where all the features mentioned in the development phase are implemented and connected, to form a tutorial, the story and the main experience of the project.

For basic VR interactions, this includes swappable locomotion for both smooth and teleportation movement, physics-based interactions for hands and objects, and UI components which could be intuitively activated and interacted with by using the hand motions and the index finger. On the technical side, it consistently holds 72 FPS during runtime and has easy to switch localization in both English and Dutch.

Implemented systems include the Post Processing Manager for ambience based on moods, AI driven dialogue, and character expressions such as breathing, lip and eye movement, and change in emotions based on the characters response. The task system implemented, which utilized and combined all features, was also used to implement guidance indicators and time-based reminders, to support inexperienced users. All this helped to form the game loop, which defines the final product.

Additionally, the project itself was kept in a tidy and well-documented state, from which the client can continue development. Scripts hold summaries and comments, which help to understand their purpose and their way of functioning, and supporting documents were created for instructions on how to utilize the established systems within Unity, as well as describe how they function on a high-level (**Appendix F**).

As such, all Must Have, Should Have, and Could Have requirements mentioned in the Deliverables section were reached, resulting in a product which is more than what was expected for an MVP. A video showcasing the product and describing the implementations can be found in the Appendix (**Appendix G**).

Results from testing the final product reflect the positive impact of the iterations undergone, especially when comparing them to the results from the previous version. This confirms the successful implementation of all features requested by the client (**Appendix F**).

3.4 Product Implementation

The final product was handed over to the client as an Android application, suitable to be launched on the Meta Quest 3. Additionally, the project itself and supporting documents describing how it is set up and can be developed further, were added as well. The executable was used for a presentation for the clients and their stakeholders, and the project and its documentation would become useful for a new team of developers for this project. A meeting was also held together with the next group of developers, to give them an in-person overview of the project. Though all the core functionalities were implemented, more polishing for certain features such as the AI and the story implementation would be needed, to be able to consider the product market-ready, regarding the technical side. This currently falls out of scope due to time constraints and will be covered in more detail in the upcoming Recommendation section.

In case the product receives time for further development, it could be used for showcases during public conventions, to raise people's awareness about dementia. Whether it would be distributed for free or be available for purchase would depend on the client's wishes and needs.

4. Recommendations

This section will cover recommendations which can be followed to further develop the project. It will distinguish between short-term development, which can be taken on without much preparation, and long-term, which will require more in-depth research and planning.

4.1 Short Term

Though guidance was put in place for interactions, testing has shown that some features are not as intuitive as intended. Specifically, this concerns pressing the send button on the phone using the index finger and sitting down on the chair. By implementing additional indicators or possibly changing the interactions, this can be done within either one or two sprints.

Additionally, real human behavior during conversations could be researched by designers and engineers, to increase the quality of the characters' idle state. This would include looking into eye and head movement for example, translating it into code and implementing it into the project, which should take one sprint.

4.2 Long-Term

Since the AI market is changing rapidly, the current services used might either change their pricing or become unavailable. A good example of this is the newest price change to Inworld, which was used for TTS, where the costs have risen by 400% in mid-development. As such, it is recommended to investigate the possibility of hosting an AI on a machine, and make it available through a tightly controlled network, to save costs during development and distribution. This would significantly reduce the risk of logic getting deprecated quickly, and mitigate the necessity of frequent maintenance, even after the project is deemed

as finished. Opportunities in this would also be the configurability of self-hosted AI, which could possibly help prevent inconsistent quality in responses, previously experienced during testing.

Holding more frequent testing sessions outside the typical study environment is also recommended, as meetings with the client have shown that elderly people need a lot more guidance than younger people with more technical experience. This could for example contribute to an increasingly refined story and task implementation, leading to a more positive experience overall.

5. Reflection

The following section will reflect on the graduation project. It will first cover the positives and negatives in the process of development itself, following up with a personal reflection on how my professional competencies grew and improved, as well as what I would like to improve next time.

5.1 Process Reflection

5.1.1 What went well

Since the projects beginning, all technical requirements and how they could be implemented were organized in a structured and reasonable manner, preventing misunderstanding and potential over-scoping. This has led to a significantly faster workflow, as well as clear communication with other team members, in terms of what is and is not feasible. This was due to early research done on the previous project structure, and the planning that followed afterwards, to show what the new structure should potentially look like using UML diagrams. To repeat this for upcoming projects, strict planning and research phases will be continued, especially in the beginning, to prevent potentially having to rework parts of the implementations.

Additionally, the code was documented and made adaptable to various use-cases, which saved time especially towards the end. This was due to a well-planned workflow, as was previously mentioned, while keeping code easily understandable in its functionality. To keep this up, the same principles will be applied for all upcoming scripts and logic as well.

5.1.2 What did not go well

More testing should have been done, especially with people outside the university. Because of this, some implementations might have to be changed when development continues, based on the results gathered. This was due to being fixed on only wanting to test, when having a fully functional implementation with a complete testing environment, resulting in rarer than intended sessions overall. To prevent this, simple tests will be held instead, where functionality will be tested as is, with no specific environment prepared. This approach would save time, while also contributing to more result-based iterations and development. To additionally gather more feedback from non-students, or students with less technical experience, tests would be held outside the usual comfort zone and familiar workspace, taking the laptop and the necessary devices to another place for example.

Though task updates and standups were held regularly, and everyone had access to the project, it was difficult for other team members to come up with specific tasks for themselves, to prepare for the next development phases. This resulted in delays for certain implementations, as their necessary preparations were only discovered, as soon as they were required. To prevent this, a meeting about future tasks for each role will be held right after the standup as well, to not only inform about individual tasks, but also recommend other roles what is high in priority and might be needed for their part of the development. By

making each team member think about the other roles, it will also contribute to a more equally distributed overview of the project.

5.2 Personal Reflection

5.2.1 What went well

During development, it was easy for me to adapt myself to new and difficult ideas for features. Even if previous developments would be scrapped with a new idea's addition, I was able to immediately switch, with no frustration building up. Having grown confident in my technical expertise, I am able to confidently estimate whether something makes sense to add and is feasible. This has helped me to iterate through different implementations more quickly, while also offering plenty of new experience for myself, when tackling previously unknown problems such as usage of AI within Unity. To repeat this behavior, I will continue to be open and curious about new ideas for implementations, continuing to build my technical expertise by developing new and unique features.

When problems arose or time was running short, I was quick to inform the other team members about it, also suggesting certain behavioral changes to reach the final project state that was aimed for. In the time that I have been working on different projects, I have grown more confident in my ability to communicate well, resolving conflicts and misunderstandings without causing internal conflicts as a result. This has helped to motivate our group to work more deliberately on what was currently needed, resulting in the final product being more than just an MVP. To repeat this, I will continue to communicate openly, especially when it means pointing out and resolving conflicts.

5.2.2 What did not go well

Communication among the team turned out to be unclear occasionally. Sometimes, by leaving out certain technical details due to thinking they were obvious, the workflow of others was hindered to some extent, leading to them spending more time on assets than necessary. This was because of me struggling with estimating what is only known to engineers, and what is not. To prevent this in the future, I will spend more time explaining implementations in more detail, while also checking how the team members have understood that request, to correct misunderstandings as soon as possible.

Though I was always trying to stay productive and reach the goals I set for myself and the project, the amount of work I was able to finish varied by a lot. This was not only due to some tasks being more complex than others, but also due to a lack of proper breaks throughout the development. To prevent this, I will be more open towards the team members, when they suggest having a break together for example, and will also be stricter with myself, to have at least one day during the week, with no planned activities regarding work.

6. References

Altundas, S. (. (2023, January 8). *Making ChatGPT in Unity!* Retrieved from YouTube: https://www.youtube.com/watch?v=MQfVCY9qgEU

HalfLifeAlyxTeam. (2020, January 22). *r/HalfLife*. Retrieved from Reddit: https://www.reddit.com/r/HalfLife/comments/esen9b/were_developers_from_the_halflife_alyx_team_ask/

Inworld. (2026, May 4). *Realtime TTS API Quickstart: Get Audio in 3 Lines*. Retrieved from inworld.ai: https://inworld.ai/resources/tts-api-quickstart

Meta Platforms. (2021, May 24). *Oculus Lipsync Unity*. Retrieved from www.developers.meta.com: https://developers.meta.com/horizon/downloads/package/oculus-lipsync-unity/

Unity Technologies. (2024, October 17). *Introduction to Universal Render Pipeline for advanced Unity creators.* Unity Technologies. Retrieved from Unity: https://unity.com/resources/introduction-to-urp-advanced-creators-unity-6

Unity Technologies. (2026, May 10). *Fixed Updates*. Retrieved from www.docs.unity3d.com: https://docs.unity3d.com/6000.3/Documentation/Manual/fixed-updates.html

Unity Technologies. (2026, April 21). *Optimize for better performance*. Retrieved from www.docs.unity3d.com: https://docs.unity3d.com/Packages/com.unity.render-pipelines.universal@14.0/manual/optimize-for-better-performance.html

7. Appendices

7.1 Appendix A – Visual Representation of Iterations

7.1.1 Iteration 1 – Core Functionalities

Most core interactions were implemented during this iteration, namely physics hands and grabbables, locomotion, free hands compatibility, as well as the PPM. These were tested internally at first, to quickly be able to move forward to the next task and implementation. The scene used for this, was the VR template scene from Unity, only adding the components necessary for the current development phase (**Error! Reference source not found.**).

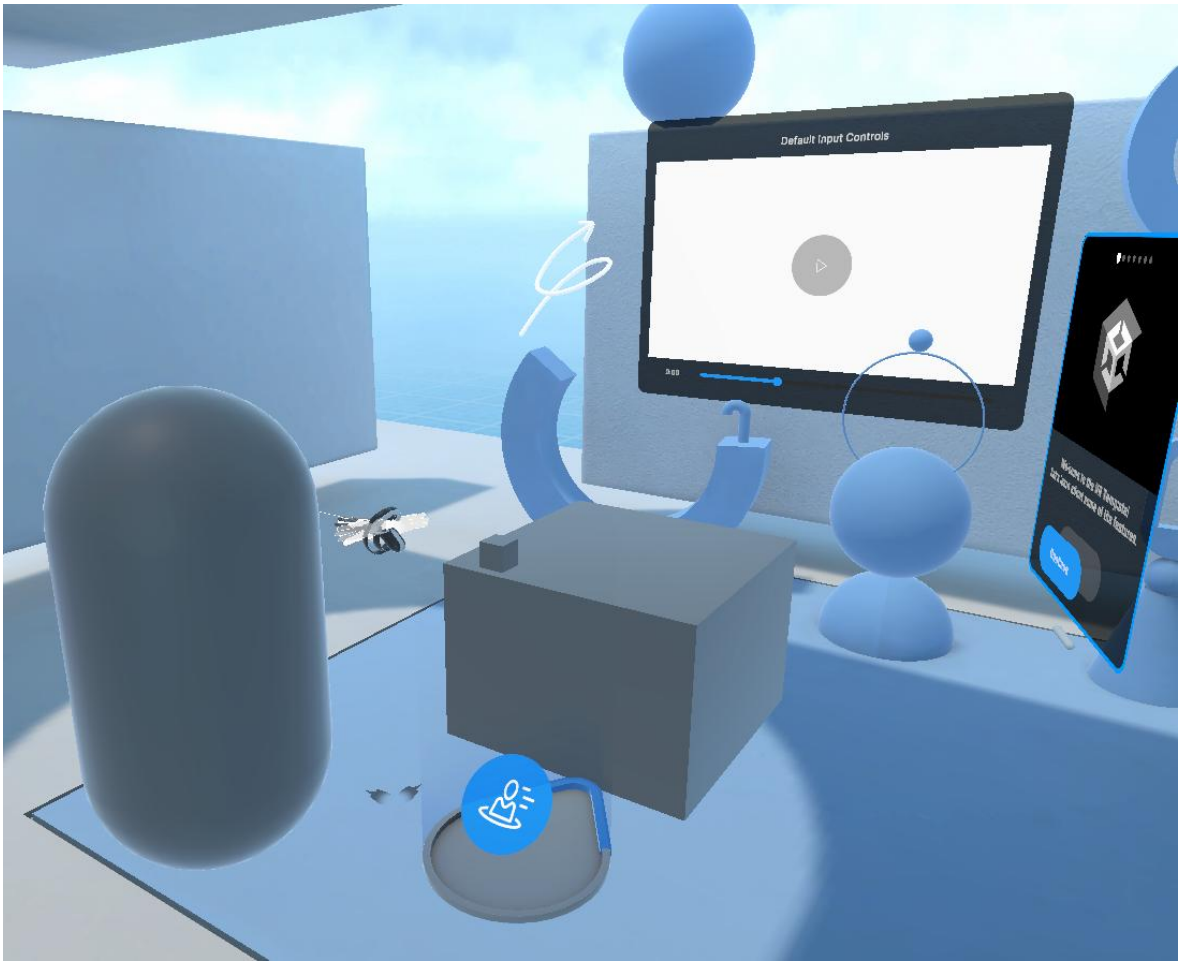


Figure 14 VR Template Scene with physics grabbable cube & functional VR Character

7.1.2 Iteration 2 – Implementation for Open Days

This iteration was used during the open-days and held the first version of the AI conversation, combined with the previously mentioned features. The user would appear outside (**Error! Reference source not found.**), would then have to move inside the house, optionally pick up the phone using their hands (**Error! Reference source not found.**), have a conversation with Juliette, and then go into the yellow zone, to experience the different moods both visually and audibly (**Error! Reference source not found.** & **Error! Reference source not found.**).

Figure 15 User outside the house - second iteration

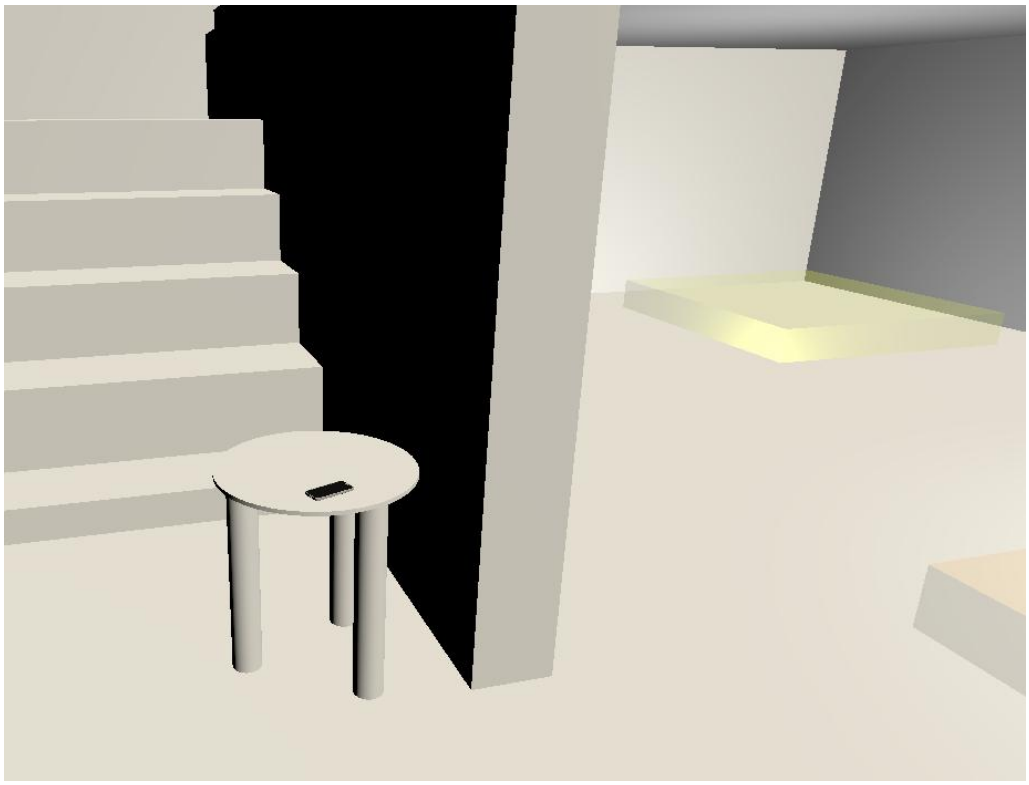


Figure 16 Physics grabbable phone on the table, and yellow zone in the background for mood activation

Figure 17 Juliette standing, ready for conversation. This version had no lip sync or facial behavior yet

7.1.3 Iteration 3 – Character Behavior & Story Implementation

During this iteration, the core functionalities were finished, and details such as lip synchronization, facial expressions based on the response, mood controls tied to the conversation, and story sequences were developed. The user would appear outside once more and be greeted by Juliettes voice to come in (**Error! Reference source not found.**). As animations were not yet available during this stage, her model would stand still in the corner of the room, and her voice would simply guide the player, as if she was moving as intended. As soon as the player entered, they could proceed to sit down and have a talk with the character (**Error! Reference source not found.**), bringing sugar to the table using the physics grabbable feature (**Error! Reference source not found.**). Afterwards, the conversation would continue, until Juliette would start humming, fading the screen to black, and restarting the scene.

User testing done during this iteration, was without the story elements implemented yet, as it was still in development. As such, the user testing after the open-days, only featured moving through the house, grabbing the object with no purpose, and talking to Juliette with her new facial controls for eyes, moods, and lips, until the maximum number of responses was reached.

Figure 18 User outside the house - third iteration

Figure 19 White cube representing the sugar to be brought. The zone beneath was used as indicator for users

Figure 20 Juliette standing in the room, waiting for the conversation to start

7.1.4 Iteration 4 – Polishing & Wrap Up

This was the final iteration of the graduation project, featuring all developed implementations covered in this report, together with all the available assets. As most of the remaining features were implemented in the previous iteration, focus was put on polishing the functionalities, tidying up the project, fixing remaining bugs, and creating proper documentation for further development by other groups.

Most assets were finalized or created during this time frame, and as such, animations and models were made to work with the created components during this iteration as well. With this, Juliette would open the door for the user, as soon as the tutorial was finished, greeting them the same way as before (**Error! Reference source not found.**). After walking down the hall, she would sit down and be able to have a short chat with the user. Afterwards, she would request the user to get the sugar, to place it on the table, and continue the conversation once more (**Error! Reference source not found.** & **Error! Reference source not found.**). As soon as that was completed, the screen would fade to black and restart the scene. User testing was also done in the last weeks of this iteration, to get impressions and find possibly remaining bugs.

Figure 21 User outside the house - fourth iteration

Figure 22 Sugar standing next to a lot of stroopwafels, with position indicator beneath & tutorial above

7.2 Appendix B – UML Diagram for Old & New Project

https://miro.com/app/board/uXjVGDFQDmk=/?share_link_id=274056444714

7.3 Appendix C – A/B Testing Teleportation and Smooth

This A/B testing was conducted in week 14 of development. It had 8 participants in total, which were distributed among A and B equally. All the participants were students.

1 disagrees with the question asked, while 5 completely agrees with it. Be careful when looking at questions such as experiencing motion sickness and preferring other types of movement, as low values mirror a positive outcome, different to most other questions.

7.3.1 About them

7.3.2 Teleportation Results

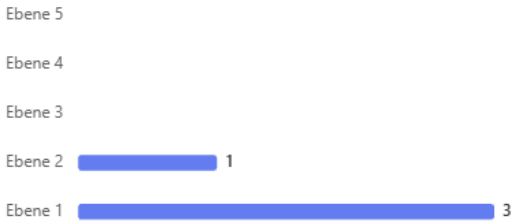
12. Did you experience any motion sickness?

Further information

1.25

Average rating

1 2 3 4 5



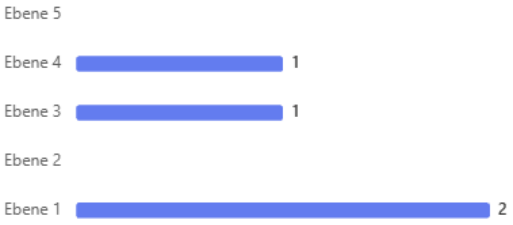
13. Would you have preferred a different type of movement/rotation?

Further information

2.25

Average rating

1 2 3 4 5



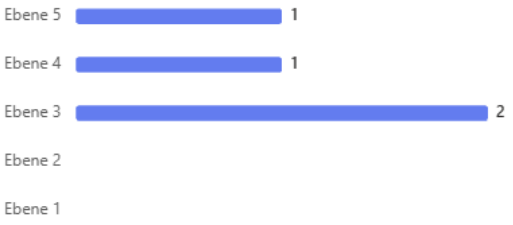
17. How much did you enjoy moving around?

Further information

3.75

Average rating

1 2 3 4 5



7.3.3 Smooth Movement Results

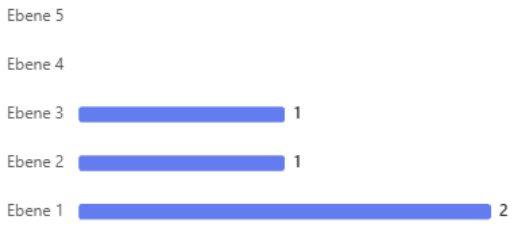
19. Did you experience any motion sickness?

Further information

1.75

Average rating

1 2 3 4 5



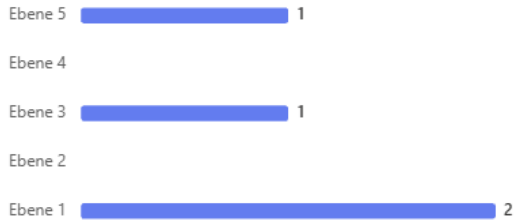
20. Would you have preferred a different type of movement/rotation?

Further information

2.50

Average rating

1 2 3 4 5



24. How much did you enjoy moving around?

Further information

3.50

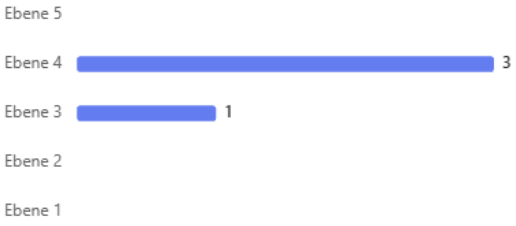
Average rating

1 2 3 4 5

7.3.4 UI Interactions Results & General Notes

11. Was it easy to start the simulation?

Further information



40. How easy was it to switch from English to Dutch?

Further information

45. Is there anything you would like to add as a note?

6 Reply

ID ↑	Name	Reply
1	anonymous	Make it so the thing you need to grab at the start cant fly away
2	anonymous	the phones resolution made the text unreadable with the headset on. It ws not immediately appearent if you are ecording sound or not. maybe some feedback like an icon in the UI would work. the phone may be glued to your character´s hand so they can move it closer to their face.
3	anonymous	good luck, y'all
4	anonymous	Some simple outer world not to confuse some players, when they look out of the window. Otherwise i liked the simulation a lot, i think it developed pretty much from my previous testing.
5	anonymous	I felt very confused talking to the grandma. She always talked about her own stuff, not really answering my questions. She mention Zutphen a lot, probably because i am supposed to come from it? The Whatsapp was unsharp because of the glasses, so i was unable to read it. I do feel that something is off though with the grandma, basically her having a condition.
6	anonymous	Already gave my pointers in the other explanations

7.3.5 AI Conversation Results

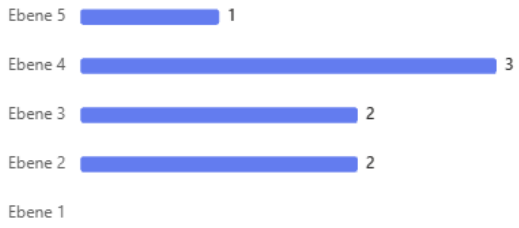
32. How fluent did the conversation feel?

Further information

3.38

Average rating

1 2 3 4 5



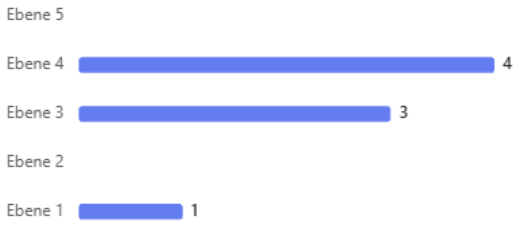
33. Did the voice sound realistic?

Further information

3.25

Average rating

1 2 3 4 5



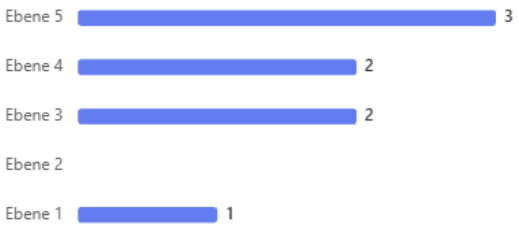
35. Was the quality in word pronunciation consistent?

Further information

3.75

Average rating

1 2 3 4 5



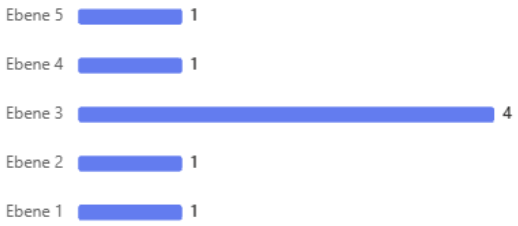
7.3.6 Character Expression Results

26. Did the characters eye contact make the interaction feel more personal?

Further information

3.00
Average rating

1 2 3 4 5

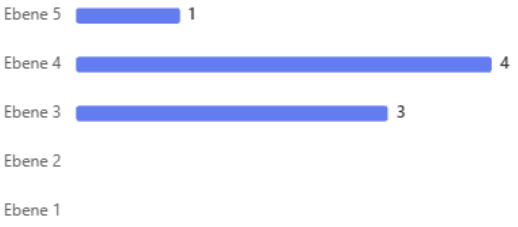


27. Did the character's mouth movement match the speech?

Further information

3.75
Average rating

1 2 3 4 5

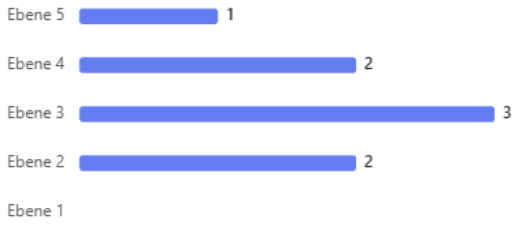


28. Did the character's facial expression match the emotions she was communicating?

Further information

3.25
Average rating

1 2 3 4 5



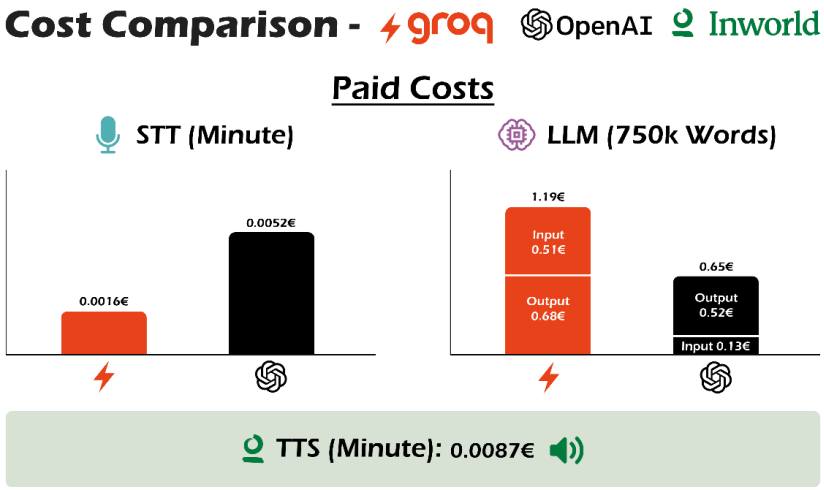
31 . Was there anything unnatural or distracting about it?

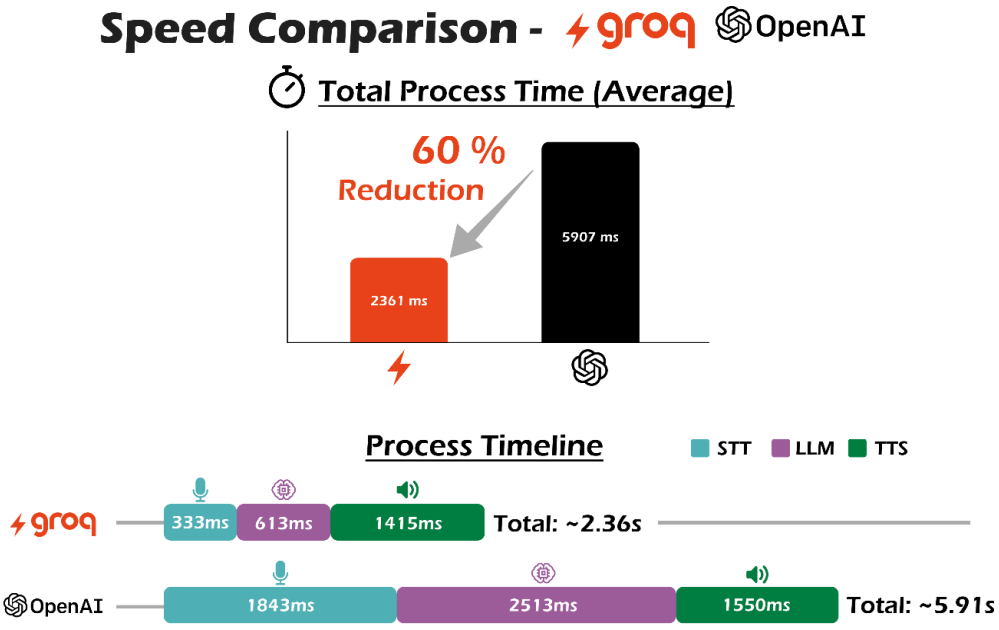
6 Answer

ID ↑	name	Answer
1	anonymous	Maybe make the character a bit smaller and turn its head toward the player/rotate towards the player
2	anonymous	besides the T-pose? not really
3	anonymous	The constant eye contact was making the experience too creepy and unnatural
4	anonymous	Ecxept of the T-pose, which is definitely going to be fixed, so I don't think it is a problem, there were no distracting actions or some unnatural behavior.
5	anonymous	her eye movement is not very realistic yet
6	anonymous	Obviously t-pose, eyes feel a bit dead

7.4 Appendix D – AI Service Cost & Performance Comparison

7.4.1 Diagrams for Comparison





7.4.2 Values examined during Speed Comparison

It's noteworthy that only STT & LLM were subject of change during tests, as no alternative for Inworld TTS could be found. All tests were done using the exact same STT prompt "The quick brown fox jumps over the lazy dog", as well as the same initial prompt for the LLM.

Groq Tests	STT (ms)	LLM (ms)	TTS (ms)
1	333	613	1415
2	294	587	1512
3	351	642	1344
4	318	599	1481
5	342	602	1372
Average	327,6	608,6	1424,8
Sum of Averages	2361		

OpenAI Tests	STT (ms)	LLM (ms)	TTS (ms)
1	1521	2104	1492
2	2134	2847	1365
3	1843	2513	1550
4	1698	2431	1524
5	2228	2901	1384
Average	1884,8	2559,2	1463
Sum of Averages	5907		

When looking at the data, tests with Groq services seem to be more reliable in consistency, regarding latency. Though there was a chance this result might only be a coincidence, as latencies can happen for every service at different points of time, this however, might also indicate that OpenAI is more prone to having higher latency, due to their popularity and its increased usage by other developers and users.

7.5 Appendix E – Test Results of Final Product Version

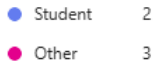
This testing session was conducted in week 18, using the final version of the project. It had 5 participants in total, three of which were no students.

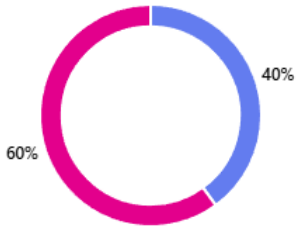
1 disagrees with the question asked, and 5 completely agrees with it. As was the case in the previous test results as well, be careful when looking at questions such as experiencing motion sickness and preferring other types of movement, as low values mirror a positive outcome, different to most other questions.

7.5.1 About them

2. Who are you?

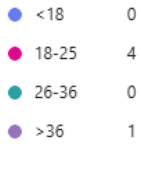
Weitere Informationen

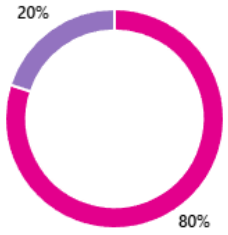




3. What is your age range?

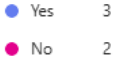
Weitere Informationen





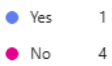
4. Have you experienced VR before?

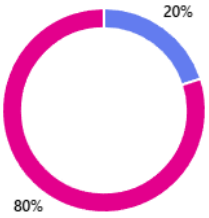
Weitere Informationen



5. Have you tested this project before?

Weitere Informationen

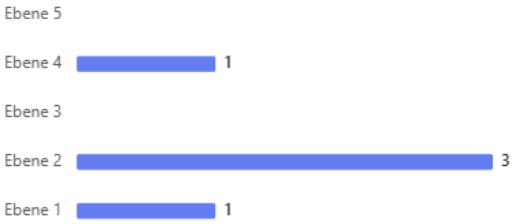




7.5.2 Movement Results

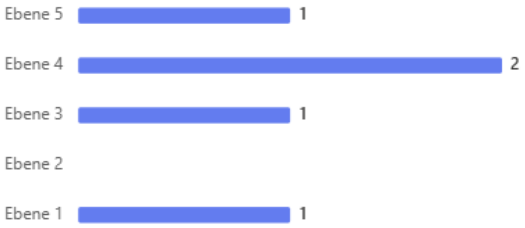
12. Did you experience any motion sickness?

Weitere Informationen



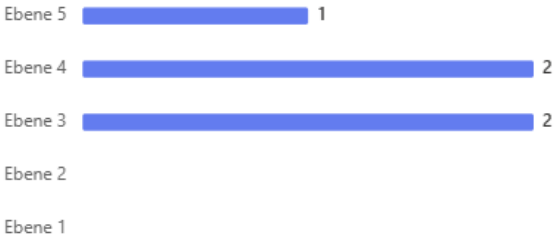
13. Would you have prefered a different type of movement/rotation?

Weitere Informationen



17. How much did you enjoy moving around?

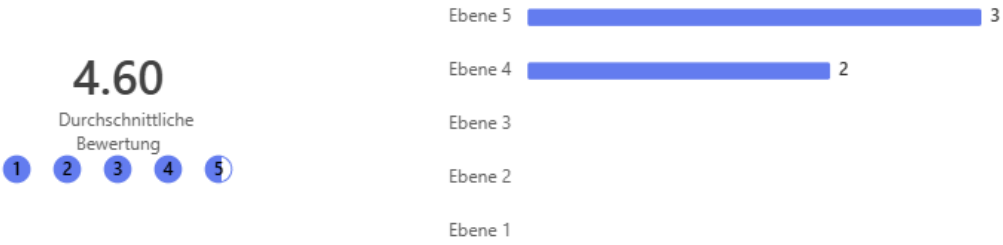
Weitere Informationen



7.5.3 UI Interactions & Guidance

11. Was it easy to start the simulation? Weitere Informationen

43. From 1 to 5, how confident did you feel using the controls after completing the video tutorials? Weitere Informationen

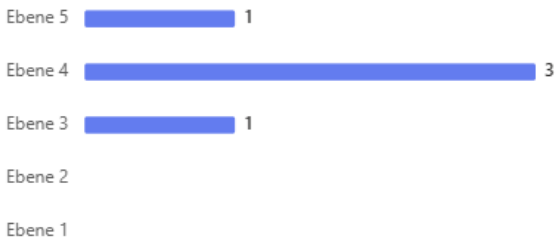


47. From 1 to 5, how intuitive and easy to use were the interactions and navigation throughout the experience? Weitere Informationen

7.5.4 AI Conversation Results

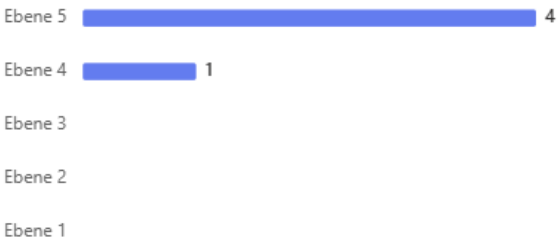
25. How fluent did the conversation feel?

Weitere Informationen



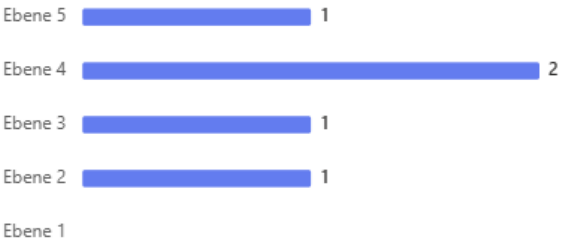
26. Did the voice sound realistic?

Weitere Informationen



28. Was the quality in word pronounciation consistent?

Weitere Informationen



7.5.5 Character Expression Results

19. Did the characters eye contact make the interaction feel more personal?

Weitere Informationen

20. Did the character's mouth movement match the speech?

Weitere Informationen

21. Did the character's facial expression match the emotions she was communicating?

Weitere Informationen

7.5.6 User Experience

47. From 1 to 5, how intuitive and easy to use were the interactions and navigation throughout the experience? Weitere Informationen

48. From 1 to 5, how satisfied are you with the overall user interface, onboarding process, and user experience of t he simulation? Weitere Informationen

7.6 Appendix F – Project Guidance Document

VR Dementia Awareness – Guidance

Spring Semester Graduation – 2026

Setup

Unity Modules and Version

Upon pulling the repository, Unity Hub should already indicate what version to download, to be able to open the project. In case it doesn't: Unity 6.3 LTS (6000.3.7f1).

Additionally, to be able to build the project, you will need to add the "Android Build Support" module to this Unity version during the install, together with the submodules "OpenJDK" and "Android SDK & NDK Tools" located directly beneath, in the Unity Hub overview. Missing this step will cause errors upon opening the project.

Git for Plugin

One of the plugins used to implement the AI logic (OpenAI Unity), was imported using Git. This is why everyone who is working on the project needs to download Git on their PC. If this is not done, Unity will throw errors, together with a reminder pop-up.

AI

To prevent errors upon start of the simulation, folders and files need to be created for the AI scripts.

The current version of the project uses Groq and Inworld for the AI generation parts. Both services require an API key to function and need to be placed in a specific directory for Unity to acquire it during runtime: "C:\Users\YourUserName".

Follow these steps:

1. Create a folder called ".openai" and another called ".inworld"
2. In both folders, create a .json file called "auth.json"

Format for the auth.json inside .openai:

```
{
   "api_key": "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"
}
```

Format for the auth.json inside .inworld:

{

"base64Key":
"YThQcGlUY1QzUmJGeVluR2tnMnI5UWd1Z0tPek1DelM6R1ZqRlpiZGdJN3RGYkVOZ2dUVUpaVkF2c0
s4MlQzNVZISllpSnZ0NFB6UERxQVJBZVZtRFpPNGd1eVN5OTg3bg=="

}

(The API keys used here will be invalid, so you will need to create your own account and key)

To make it work within VR standalone, you will need to get the .json files into the headset. Specifically, you will need to go to **"\Internal shared storage\Android\data\com.DefaultCompany.VRTemplate\files"**, though the com name might change, based on your project settings. Name the inworld .json "**inworld_auth.json**" and the groq one "**openai_auth.json**". The folder itself will only appear as soon as the application has been installed in the headset.

Useful to know when working within Unity

VR Character

All VR features within the project were implemented using the plugin Meta XR All-in-One SDK. This plugin holds building blocks, with which it was possible to create a versatile base for a VR character, as well as interactions such as grabbing and UI navigation using hands.

The base hierarchy of this VR character is already complex, as all components are intertwined and necessary in a way, while purpose of them is not always clear. The one used for this project, was carefully tailored and developed, to enable the following behaviours:

- Have physics hands (make them collide with other colliders properly)
- Make it possible to have physics grabbables
- Have different locomotion (slide & teleport move | smooth & snap turn)

A lot of additional GameObjects and scripts were created, to enable such interactions, and make them adjustable during runtime. To quickly explain the principle of the implementation:

Physics Hands

- LPhysicsHand & RPhysicsHand are the physics hands, created additionally to the base hands of the VR character
- The base hands still exist but were made invisible, and control the movement of the physics hands
- As such, the physics hands follow the real hands (which are not physics interactable), which in the project is called **Proxy** Approach. All objects called like this are used for this exact purpose (only the original hands were not changed in name)

Locomotion

Locomotion settings are (de)activated, based on what GameObject with the fitting component is active within the hierarchy. See the pictures to see where (Figure 23). Currently, these components are toggled by scripts, based on what configuration is chosen in the menu.

The character already changes between controllers and free hands, based on if the controllers are being held or not. To change what hands are shown when using the controllers (only controller, controller & hands, only hands), navigate into the character, look for the “OVRCameraRig”, and change the value “Controller Driven Hand Pose” within the hierarchy.

Camera

See the picture to find out, where the VR Camera is located at (Figure 24).

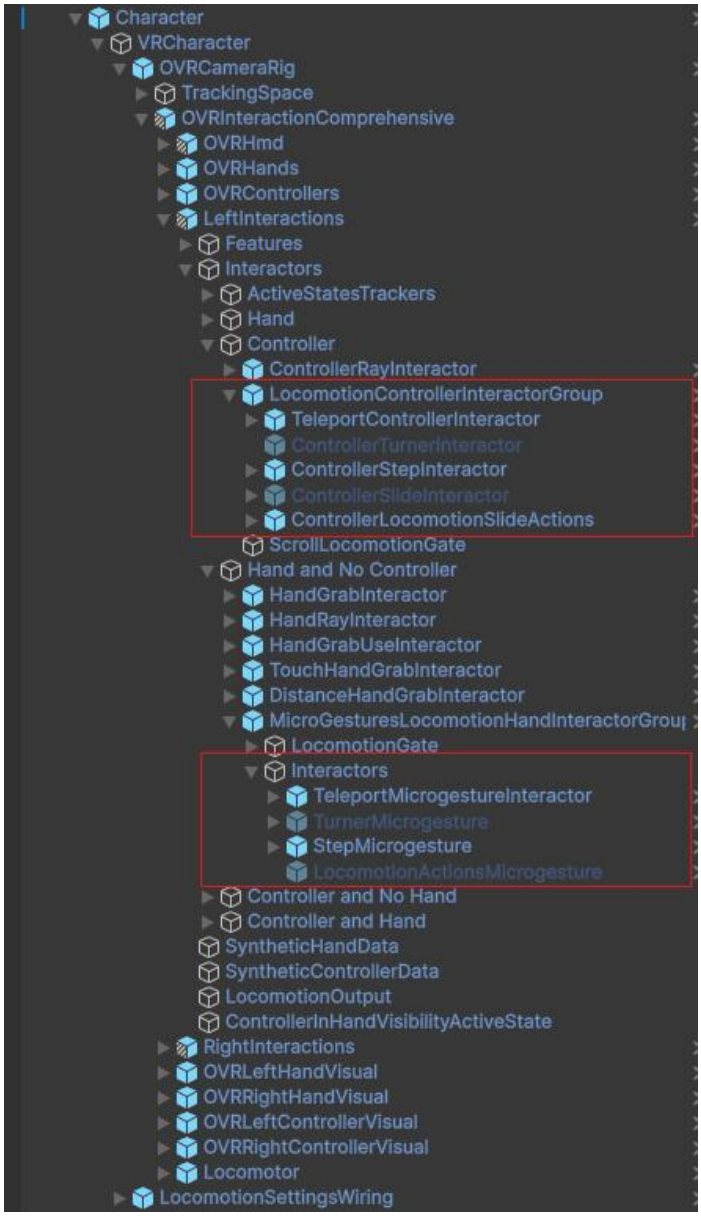


Figure 23 Upper rectangle is used for Controller locomotion | Lower for free hands

Figure 24 GameObject marked is the VR Camera

Setting up a physics Grabbable Object

Follow these steps to create a properly working physics grabbable object:

1. Create an empty GameObject and call it appropriately (e.g. Sugar_PhysicsGrab)
2. Drag the model/object you want to turn into a grabbable into the new GameObject
3. Copy the Model. Name them “ExampleObjProxy” and “ExampleObjPhysics”
4. On “ExampleObjProxy”, press right click -> Interaction SDK -> Add Grab Interaction
5. Make sure both Proxy & Phyiscs GameObjects have a RigidBody (RB) and the same collider

“Proxy” Configurations for RB & Collider

- Activate IsTrigger on the BoxCollider
- Deactivate UseGravity and activate IsKinematic on the RB

“Physics” Configurations for RB & Collider

- Keep IsTrigger deactivated
- Activate UseGravity and keep IsKinematic deactivated

6. Add a GrabCollisionHandler script to “…Proxy”. Attach the “…Physics” GameObject as target and leave IgnoreHandLayer as is (unless you plan on renaming the layer)
7. Add a GrabFreeTransform Script to the proxy. Open the Optionals drop-down in the Grabbable script and attach the GrabFreeTransform as OneGrabTransform and TwoGrab. Attach its own GameObject as Target Transform
8. Add a Physics Proxy Follower script to “…Physics”. Drag the Proxy GameObject into Proxy Target and GrabCollisionHandler, and attach PlayerController to Player Locomotor (Figure 25)

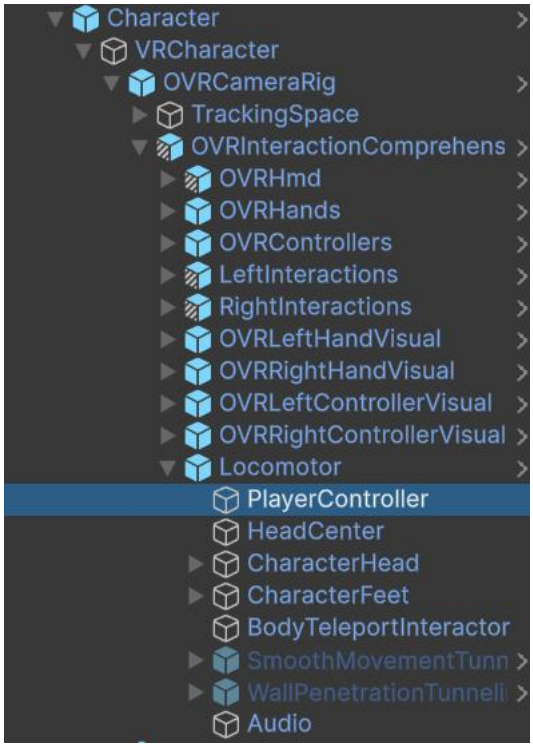


Figure 25 Location of the PlayerController

9. Open the Hierarchy with "…Proxy" and scroll down to the two Interactable Unity Event Wrapper scripts (add them if they are not there). On both, add an event on WhenSelect and WhenUnselect. Attach "…Proxy" to both events, and call HandleGrab() from the GrabCollisionHandler script for WhenSelect, and HandRelease on the other. Repeat for the same component beneath as well. One is for free hand grab, the other is for controllers.

With this, the object should be fully grabbable and interactable with physiscs.

Implementing Story Sequences

The Managers GameObject holds the TaskManager script. This script holds a collection of GameObjects, all of which have a script attached, which inherits from SimulationTask. To add new tasks, create a new script, which inherits from the previously mentioned class, and implement the logic needed for that story part. By attaching it to the TaskManager, it will become part of the sequence. Look at the class itself and its previous usage, to figure out the details.

Post Processing Manager

The visual part of the mood was implemented using Post Processing. To manage those, together with the FMOD sound events, scriptable objects were created and used within the manager.

Volume Configurations are used to set and mix moods, and Volume Collection is used to define which Post Processing prefabs & FMOD sounds belong to which mood.

The Post Processing Manager then takes care of using the references saved within the Prefabs, to make it work within the simulation. To use the PlayMode Mood Controls, and see its change in runtime, press play on the project, switch to the scene view, and change the sliders.

VR UI

To make any future UI work as intended, either copy the instances which are already available and work, or create your new base, by taking an example of the Prefab given by the Meta XR All-In-One plugin called “FlatUnityCanvas”.

If you increase the size of it, make sure that the Surface GameObject (Bounds Clipper script) covers the whole UI, as only this will enable the VR index finger to be detected. The “Unity Canvas” rectTransform should also be appropriate of size. If you end up with a UI that is not working in VR, one of these settings being wrong is the most likely culprit.

The UI itself can be designed, by replacing the children located within “Unity Canvas”.

7.7 Appendix G – Contributions Showcase Video

https://media.saxion.nl/media/Graduation+Video+Showcase+-+540076+Andreas+Degtjarow.mp4/0_00fdjo35