Bachelor Informatica

University of Amsterdam

Audio Recording Function for Perspective Taking in 360Composer

Kimberley Roks

June 23, 2025

Supervisor(s): Dr. Rob Belleman

Signed: Signees

Abstract

Virtual reality allows for training several skills in a simulated environment. One of these skills is perspective taking: the ability to imagine and understand how others might react to ones own actions and words. By allowing an user to record their actions and have it played back at them, they get first-hand experience with how others might perceive them. By implementing an audio recording feature in 360Composer, a web-based virtual reality application that uses 360-degree videos, its users can experience this perspective switch in an accessible medium. This research details the implementation of this audio recording feature, which was validated using the *SUS* questionnaire. Fifteen participants were given a scenario where they recorded their voice, which was played back at them, after which they were asked to fill in the questionnaire. The results show that the implemented recording feature achieves a high average SUS score of 81.07. However, the participants noted that the current controls of the application are reducing the level of immersion.

Contents

CHAPTER 1

Introduction

Virtual reality (VR) is an applied simulation of 3-Dimensional environments. It allows for the simulation of a variety of different scenarios with varying levels of interactivity. The ability to experience all of these, potentially dangerous, scenarios without any physical risk makes it an interesting tool for research in the psychological field.

Several studies have been conducted regarding perspective-taking and its effect on participants' behaviour. Since it allows for high levels of immersion in a digital environment, the users are capable of fully immersing themselves in the given scenarios. When these scenarios combine the changing of perspective, it allows the users to literally see how their actions come across to others.

However, most virtual reality applications require the usage of expensive headsets, controllers, and software to be used. This makes it rather non-accessible to people with lower budgets. The controllers and software is necessary for the high level of interactivity. However, some implementations of virtual reality do not require these.

360-degree videos are videos that allow the usage of a device's orientation to change the camera's orientation. These videos are pre-recorded experiences, which come with the inherent disadvantage of limited interactivity, which lowers the immersion.

360Composer is capable of circumventing this disadvantage however. It is a web-based application that allows for non-linear virtual reality experiences by connecting several 360-degree videos together. By letting the user use their device's orientation to make selections within the application, they are capable of interacting with and influencing the experience. Used in *Parent-Child Interactive Therapy* (PCIT) virtual reality applications [4], it is able to provide an accessible tool for therapy contexts, as well as other fields.

Currently, 360Composer allows for interaction by three means: one is the ability to make a decision in an option menu. The other is to have the user shake or nod their head. Finally, 360Composer is able to detect when the user is blowing air into the microphone [4]. However, none of these facilitate the ability for users to perform perspective-taking exercises. This would be achievable with an audio recording function, which would allow the user to record their voice and have it played back at them at a different point in time.

This research implements this audio recording function in 360Composer and researches its usability by means of the *System Usability Scale* (SUS) questionnaire. The recording function must be simple to use, and not break the users' immersion. This research's experiment is performed by letting approximately 15 participants experience a scenario which requires them to record themselves, which will then be played back at them in a changed perspective. After the experience, they will be asked to fill in the SUS questionnaire. The results will then be illustrated and analysed for how the functionality can be further improved.

1.1 Research Question and Sub-questions

The main research question of this project is:

- How can we implement an audio recording function in 360Composer?

Due to the context of the research, which is focussed on the ability to switch the user's perspective, the audio must be able to be stored and played back to the user. Therefore, the following sub-questions will be answered:

- How can we record and play back audio in 360Composer?
- How can we store and retrieve the audio recordings?
- How do users experience the audio recording function in 360Composer?

In order to answer these questions, we will first give a short explanation of how virtual reality can affect its users through immersion, and how perspective-taking is capable of changing behaviour in chapter 2. This if followed by an in-depth explanation of the current workings of 360Composer. Finally, the requirements of the audio recording function are laid out.

The theoretical background is followed by an in-depth description is given on the implementation of the audio recording function in Chapter 3. This description will go over all the changes made to 360Composer, from updates made to existing scripts to new functionalities that had to be added.

The results, located in Chapter 4, will display several graphs detailing the results of the SUS questionnaire. These results will contain graphs about each question in the questionnaire, as well as a section dedicated to any extra comments the participants might have added. The total mean, median, and mode of the questionnaire's score will be displayed in a final graph.

The results are followed by the conclusion of this project in Chapter 5.

1.2 Ethics

Due to the nature of the recording function, there are potential ethical concerns that arise. This is due to the unavoidable storage of recordings of participants. In order to ensure the participants' privacy, extra measures are taken so the audio recordings created during this research are immediately destroyed upon completion of the experiments. This way, no unnecessary private data of participants is stored.

Furthermore, the experiment makes use of an ethical compliance form for low-risk student projects, of which a template was made available by the University of Amsterdam. This compliance form, as seen in 6, will fully inform the participants of the expectations of them, what is and is not stored for the experiments, and of the ability for them to withdraw from the experiments at any time. When they decide to withdraw, any and all data of said participant is immediately destroyed, and no further action is necessary of said participant.

CHAPTER 2

Theoretical background

2.1 Virtual Reality

Virtual Reality (VR) is a simulation of a 3-dimensional environment using the users' heads orientation to rotate the camera. This allows the user to see the simulated world as though they themselves are actually in it. Virtual reality can, but is not always, interactive, often using handheld controllers that are used to interact with the environment. An example of a non-interactive VR environment is a 360-degree video. Here the user can still look around, but they have no further influence over the experience itself.

A well-known method of creating an interactive VR experience is through the usage of video games, where the player can directly influence the environment around them. One of the earliest examples of this is the *Power Glove*. Built in the 1980s, this controller could be used for two games: *Super Glove Ball* and *Bad Street Brawler*.

However, the Power Glove and its implementation proved to be too expensive and not functional enough to be sold to a broader audience. Even today, virtual reality headsets, controllers, and software prove to be rather expensive for the broader audience.

2.1.1 Applications of virtual reality

Whilst virtual reality is often used in video games, it has also seen applications in other areas. In therapy, virtual reality is used in exposure therapy related contexts. In [12], several applications in the context of therapy are explored. An example is the usage of virtual reality to treat *blood-injection-injury phobia*. Patients with this phobia can experience high amounts of anxiety when they need to get their blood drawn, even to the point of losing consciousness or running away. By using *virtual reality exposure therapy* (VRET), the patients can be taught self-calming techniques or be given more positive experiences in this specific scenario. Knowing that they are not actually getting their blood drawn, patients are able to feel more detached from the scenario and become more accustomed to the other aspects of the procedure [12].

Other applications can be seen in the law enforcement. In 2024 the *European Union Agency for Law Enforcement Training* held a course on how to handle illegal drug production. In this course, the participants were given several virtual reality scenarios where the participants were able to practice and use simulations of the current technology used the visualize drug production sites [2].

2.2 Why do we need perspective-taking?

One of the key reasons behind the need to implement an audio playback function is the ability to allow the users to shift their perspectives and see themselves from another point of view. The ability to shift perspective to that of another is key to nurture empathy and understanding. As

such, several studies have been conducted on perspective taking and their benefits, including studies focused on perspective taking in a virtual environment.

Perspective taking is described as the action of looking at a situation from a different viewpoint. Associated with pro-social behaviour [8], perspective taking is key for healthy relationships [9].

2.2.1 Real World Perspective-taking

Due to the effect interpersonal relationships have in a variety of field, several researches have analysed the benefits of perspective taking exercises. In [7], A large-scale perspective-taking study was performed on both the teachers and students of a school network in the United States. The teachers were given a course in perspective-taking exercises, which they were to apply to students they could not understand the behaviour of. The research showed an increase in understanding, as well as an increase in the students' grades, highlighting a non-social advantage to perspective-taking and empathy.

2.2.2 Perspective-taking in Virtual Reality

[10] In previous research it has been shown that perspective-taking exercises in virtual reality are capable of influencing the participants' behaviour in social settings. Due to the immersive nature of virtual reality, several different scenarios can be created and practiced. In [5], two separate studies were performed: one analysed the influence of perspective taking using an online environment, whereas the other used a virtual reality environment. The virtual reality environment allowed the participants to experience how the other gender experienced the situation. It allowed the participants to further understand the other genders, showing an increase in empathy towards them.

2.2.3 The Effect of Immersion

Immersion is a key component in the virtual reality experience. The more immersion the user experiences, the more authentic the experience seems. In [13], they researched the effect on immersion on the perception of authenticity the participants experienced. In order to test the effect of immersion on customers' virtual experiences, they conducted several experiments, each focussing on different aspects of a virtual reality experience. They found out that by improving the levels of immersion within their virtual reality experience, the participants perception of authenticity improved. This lead to them both enjoying the experience more.

In [14], a study was conducted which analysed the effect of interactive virtual reality on children with autism spectrum disorder. Finding that previous research was unable to sufficiently immerse and engage these children, [11] sought to design a more immersive virtual reality system. Due to the effect autism spectrum disorder has on the social skills of these children, the goal was to create an experience where they would practice these.

2.3 360Composer

360Composer is a web-based application created by the Visualisation Lab at the University of Amsterdam. It is intended to be a low-requirement alternative to the more expensive virtual reality applications. It allows its users to view 360-degree videos in a non-linear fashion on any screen, such as a phone, a laptop, or a VR headset. The non-linearity is achieved through a choice-based video player, where the user can make decisions based on *annotations* within the application.

2.3.1 Users and Customers

The 360Composer environment has 2 types of roles: *users* and *customers*. The users are the people who can create and manage their projects. They are able to create several scenarios and timelines, which they can make accessible to predefined customers. The customers are the people

who can only access timelines that they have been given a link and an access code to. They are defined with a name, a tag, and an access code. They are unable to do anything else with the project. The 360composer environment collects statistics per project about the customers, which can be exported to a *CSV* file by the user.

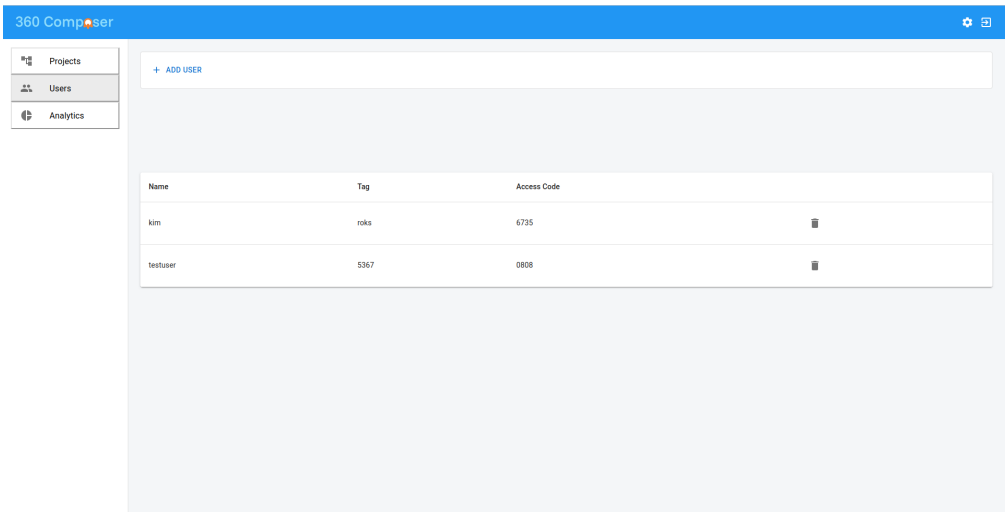

Figure 2.1: The user menu in 360Composer. Here the user can create new customers and their access codes, which can later be used to select who is allowed access to which timelines.

Whilst the 360Composer environment refers to the customers as users in its interface, this paper will continue using the previously stated definitions of users and customers.

2.3.2 Creating a project

In order to create a virtual reality experience in 360Composer, the user must create a project. Each projects data is stored separately on the server, meaning data, such as assets or scenes, cannot be shared between projects and have to be recreated.

When an user logs in for the first time, they will be taken to the projects screen. This will be empty, with only the option to create a new project being available. Once they do this, this first and all other projects will be displayed here. They can then select a project to either edit or delete.

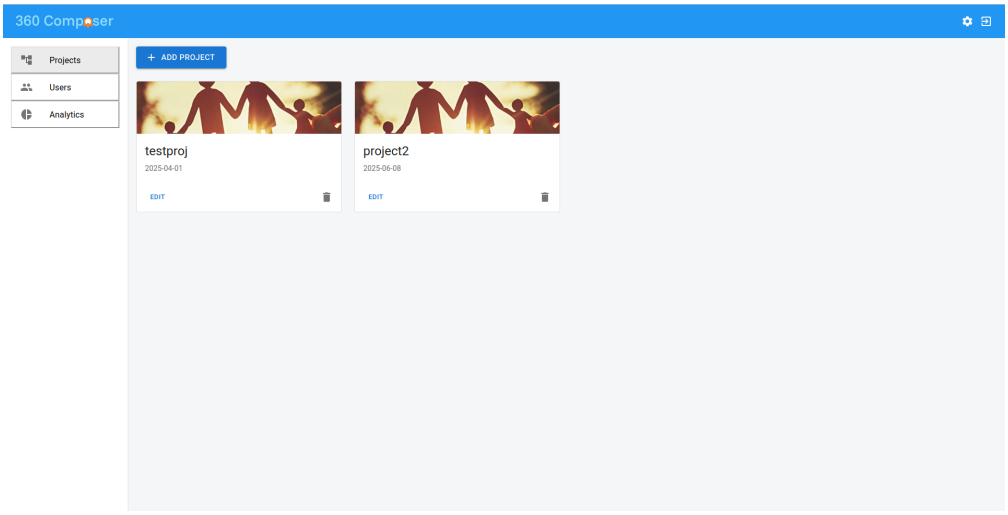

Figure 2.2: The project screen in 360Composer. Here the user can create, edit, or delete projects.

Each project contains its own components. These components consist of *assets*, *scenes*, *scenarios*, and *timelines*, which each previous component being a subcomponent for the next.

Assets

The assets are the 360-degree videos the user can use to create their scenes in 360Composer. These assets are uploaded by the user to the server, which stores these based on the *Assets* table in the database. This table contains, amongst other characteristics, the identifier of the asset, combined with the *user-id* and the *project-id*. The identifier is random, and the file-name and path is stored to ensure security.

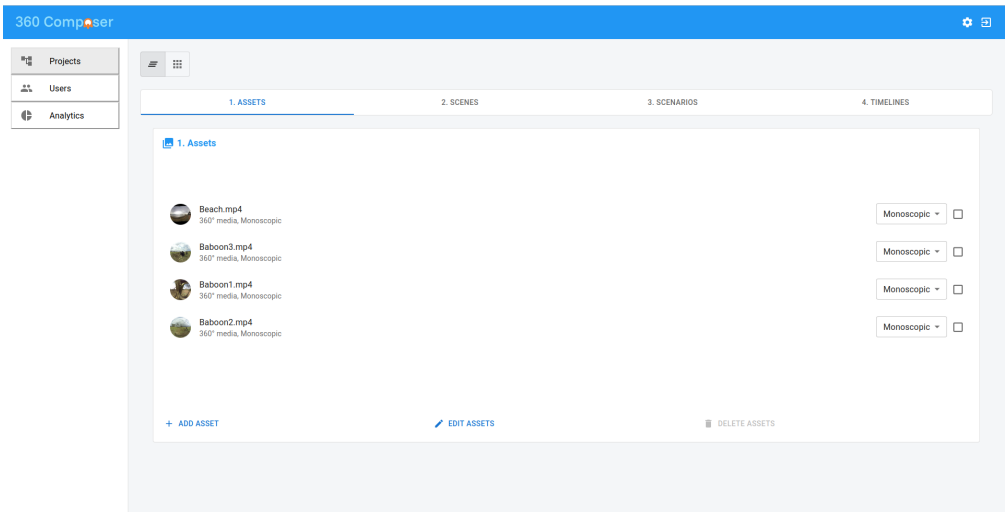

Figure 2.3: The asset screen in 360Composer. Here the user is able to upload and edit their video assets, which are later used in scenes.

The user is able to edit these assets in a simple video editor. Here they can cut an asset video into smaller parts and either duplicate, delete, or export these as is. Furthermore, the viewing type can be changed to a monoscopic or stereoscopic format. A monoscopic format means that in a virtual reality headset the eyes both receive the same image. Meanwhile, a stereoscopic video uses two slightly different perspectives to give the illusion of depth. This can either be a side-by-side view, or a top-bottom one. This is used when a single screen is used for the virtual reality headset, such as with a mobile phone.

Finally the projection format can be changed to either *unknown*, *cube mapping*, or *equirectangular*. When set to *unknown*, the video's own projection format is used. However, this can create issues when concatenating snippets of videos with different projection formats. *Cube mapping* uses a cube as the map to display the surroundings. Each face of the map corrseponds to a direction.

Equirectangular projection projects the view onto a rectangle. The most common example is a world map. An equirectangular projection is prone to distortion around the top and bottom edges, as these will appear stretched out.

Scenes

In the scenes section the user can create scenes, which are later used to create scenarios. Each scene can be given a name and a description. Upon editing the scene, the user can select an asset that will be used as the video. The brightness can be further adjusted, and *annotations* can be placed at any point in the video. These annotations are points in the video where the customer can interact with the scenario. The different types of annotations are: *text*, *nodding*, *blowing*, and *other*. *Text* will allow the customer to choose one of several options displayed in a menu with several buttons. Upon choosing an option, the corresponding scene loads. *Nodding* allows the customer to nod or shake their head to respond with respectively 'yes' and 'no'. The *blowing*

annotation type detects whether the user is giving any input, and loads the scene depending on whether they do or not.

Whilst it is possible to create several annotations per scene, the current implementation of 360Composer can only make use of one per scene. This is because the next scene is loaded depending on the decision made in the first annotation.

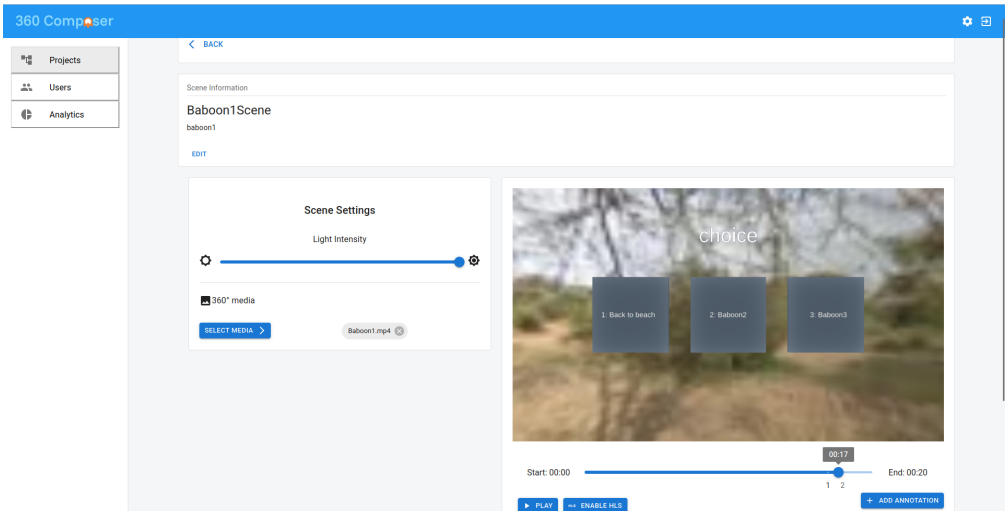

Figure 2.4: The scene editor screen in 360Composer. When a scene is created, it can be edited here to select which video asset is to be played. Furthermore, annotations can be added, which can be used in the scenario editor to determine which option goes to which scene next.

Scenario

Upon creating a *scenario*, the user can load in their scenes. All scenes loaded in are displayed as a scene object containing the scene name, asset name, and input/ output points. Any options in the annotations made in the scene editor will show up as an output point on the scene object. The input and output points of the scenes can be linked together, allowing for the options in the scenes to point to the next scene.

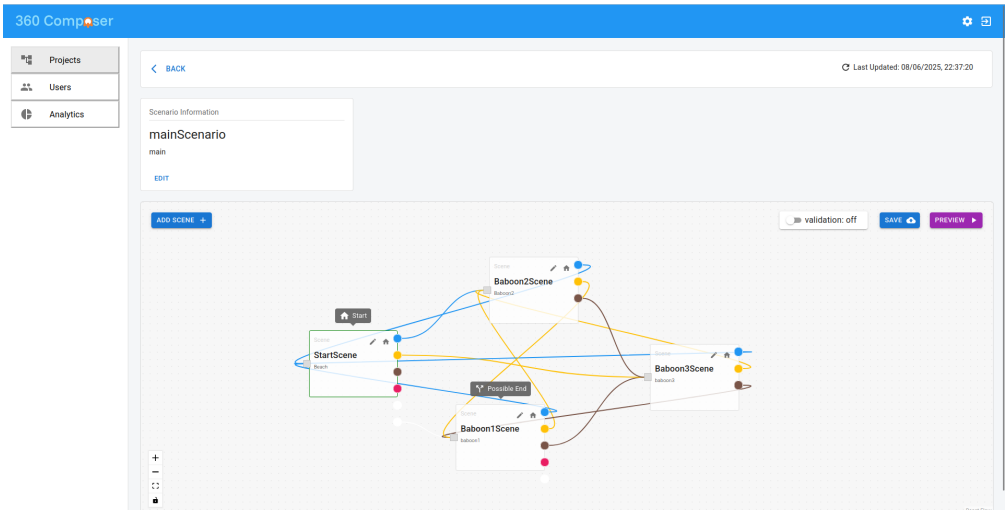

Figure 2.5: The scenario editor in 360Composer. Here the user can determine which scenes to add to the scenario, and where the annotations' options point towards, as displayed by the color-coded lines.

Timeline

Timelines are the final component of a project. Here the user can select which scenarios they want to use, combined with which customers will be given access to it. The timeline editor *user interface* (UI) is similar to the UI of the scenario editor, where each scenario is displayed as an object in the preview box.

Each timeline can either be ordered or randomized, which makes the order of scenarios either pre-set or random. When set to ordered, the scenario objects can be linked together in a manner similar to the scenes in the scenario editor.

Finally, similar to the scenario editor, the user can preview their current timeline.

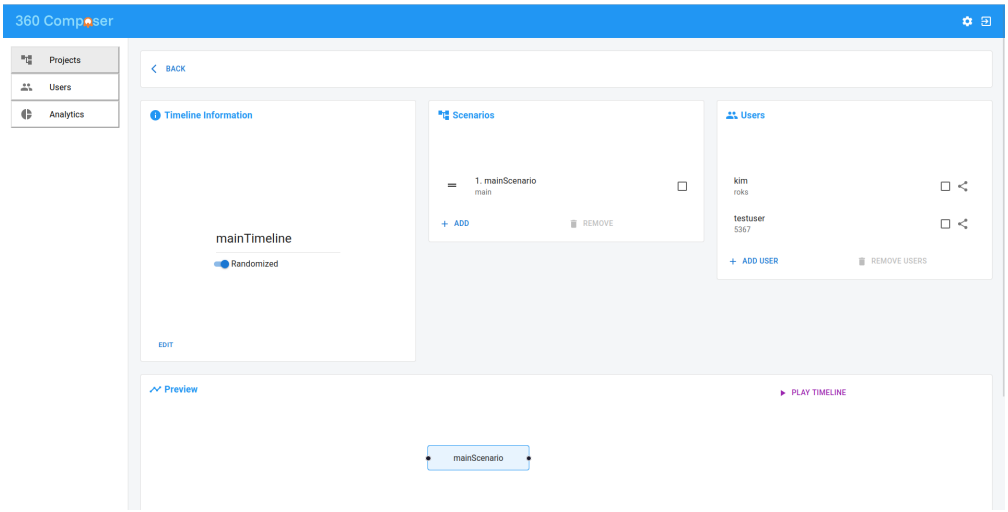

Figure 2.6: The timeline editor in 360Composer. Here the user can determine which scenarios to add to the experience, as well as which customers are allowed to access it.

2.3.3 Using a Timeline

The user can choose which customers will be given access to a timeline. Each customer is given their own unique link to the project, which will send them to a separate login screen. When they enter their access code, they will immediately be taken to the first scenario, and the corresponding first scene.

Upon first loading into a scenario, the customer will be asked to look at a button on the screen. This button is to ensure the user has interacted with the page, allowing 360Composer to play audio. When the first button is pressed, the video will start playing. Once the first annotation of the scene is reached, the customer will be asked to make a decision based on what they saw earlier in the video. This decision is displayed as an option menu with several buttons, which will take them to the corresponding scene in the scenario. If the option is given a description in the scene-editor, this will first be displayed. The description has to be dismissed before the customer is taken to the next scene. Once an end-point has been reached, the user will be given the new scenario. If there are not any, they will reach the end screen.

2.3.4 360Composer Frontend

When the user or customer is interacting with 360Composer, they interact directly with several front-end scripts. Furthermore, several front-end scripts are used to dynamically change and update the website's html code, which in turn allows the webpage to change without needing to reload entirely. These are located in the *Viewing App Components* directory.

Viewing App Components

This directory contains several objects used for the *User Interface* (UI) of the 360Composer webpage. This directory contains all the *typescript* scripts used in the displaying of menus. One of these is the *StartMenu.tsx* file. This is the menu the customer first sees upon entering a scenario. The script ensures that the button is always located at a set position, and uses event listeners so the buttons activate the corresponding function calls.

Other scripts are for displaying other types of menus in the VR environment, such as the annotation option menu. This research required adding four more menus, which were necessary for starting and stopping the audio recording and playback.

Viewing App Controller

The *viewing app* controller is responsible for the correct loading and ordering of the scenarios, scenes, and corresponding annotation data. Through the usage of *states*, the controller keeps track of which scenario and scene it is currently displaying. The controller and the AFrame communicate with each other in order to fetch and display the correct scenes by using function calls and callbacks.

By communicating with the API in the back-end, the controller is capable of fetching the correct data. It achieves this through using API post- and fetch-requests with the current scenario and/ or scene identifier as arguments. The received data is then sent back to the AFrame, which is responsible for displaying them.

Due to the controller being responsible for communication with the back-end, it was a critical file for this research.

Viewing App AFrame

The *AFrame* is the front-end script responsible for updating the virtual reality environment and responding to events. It is used to display the correct video and respond to annotations once they are reached. Several callback functions are implemented to ensure proper communication with the controller, and to set app states to the correct values.

One of the core functions is the *onTimeUpdate* function. This function is responsible for pausing the video once an annotation has been reached, allowing for the annotation options menu to be displayed. The app state *videoPlaying* is set to false, whilst the *menuEnabled* state is set to true. only when these two states are set to these values will the AFrame display the annotation options menu. These annotation options will trigger the *chosenMenuOption* function, which first sets the *menuEnabled* state back to false, and then communicates the option values to the controller. The controller will use the *menuOptionCallback* function of the AFrame to communicate whether the application needs to be exited or ended. If the option would require a new scene to be loaded, the callback is not used, and instead the AFrame is reset with the new video and scene annotations.

Due to the AFrame being reset upon selecting an option, it was not possible to resume the video after an annotation was reached. Since all annotation types thus far always connect to either a new scene or the end of the scenario, any annotations in the scene after the first would never be reached. This would prove problematic for the audio recording and playback functionalities, as they would use their own annotations. These would cause errors, as they have no values that indicate whether a new scene or the end screen would have to be loaded.

Creating annotations

In the *Editor Components* directory the scripts for creating and updating annotations are present. These are used to communicate to the user when an annotation was successfully created or updated. These scripts further handle the API calls necessary to store the annotation data in the database, and contain the UI html code for the scene editor.

The *NewAnnotationDialog* and *UpdateAnnotationDialog* scripts both fulfill a similar purpose, and therefore have very similar functions. However, they still needed to be kept separate in order to prevent unnecessary API calls to the database. The *NewAnnotationDialog* script only uses

a *POST* request once the user saves their new annotation. Any changes before will be saved locally instead.

2.3.5 360Composer Back-end

Routes

The *Routes* directory stores the scripts responsible for handling API calls. For example, when a new asset is uploaded a *POST* request is sent to the *project.py* script with a formdata containing further video properties. The request path is then used to route the request to the *ProjectAssets* class, which in turn stores the video and creates the database entry.

Database

The back-end is also responsible for setting up the database upon newly creating the environment, as well as any and all API calls and routing. In the *migrations* directory are the version updates stored under *versions*. These use *alembic* and *sqlalchemy* to create and update the database tables for the first time. These tables are then used for storing asset file paths, creating annotations and their types, keeping analytics, and more. For this project, the annotations and annotation types tables were the most important, as these would be directly edited to allow for the addition of the audio recording and playback annotations.

2.4 The implementation of an audio recording feature

In order to implement an audio recording feature in 360Composer, it must remain easy to use and accessible to a wide audience. This means that the customers should not require any special equipment, and the feature should be created in a manner similar to other functionalities in 360Composer. This requires the ability to start and stop both the recording and the playback by looking at a button. Furthermore, the annotation function needs to be adjusted to allow for the usage of a recording- and playback-type annotation. This requires the addition of more annotation types to the database. Finally, the function calls from the front-end to the back-end need to be able to facilitate the storage and retrieval of audio files. This can best be achieved in a manner similar to how video-assets are stored, as to not over-complicate the design. This requires a new table in the database, which stores the audio file path and identification.

2.4.1 Audio file format

There exist audio file types, each with varying levels of compression and support from different sources. The quality of these audio files can be determined by the level of compression, bit rate, and sample rate rate. For example [1], the *MP3* format is used to store audio with a Layer 3 compression, allowing for smaller file sizes.

Not all browsers support the same file types however. For example, the *WAV* format, commonly used for storing uncompressed audio, was not supported on earlier versions of the Chrome browser, and currently is still not supported by the *Opera Mini* browser. The *Opus* format is currently only partially supported on the Safari browser.

Due to high levels of accessibility being a requirement for the audio recording function in 360Composer, the decision had been made to use the *WebM* format. This format was designed with browser support in mind [15]. The *WebM* project based their file format on the *Matroska* multimedia container format. The Matroska container is designed to allow users to store large amounts of data in a single file whilst still being quickly accessible[16].

WebM prizes itself on being computationally cheap to use and accessible on any device. Due to the WebM container using the *Opus* This makes it ideal for usage in 360Composer, as it would allow customers to use it on lower-end devices as well. The WebM format is currently supported by all browsers except *Internet Explorer* and *Opera Mini*.

2.5 System Usability Scale Questionnaire

The *System Usability Scale* (SUS) questionnaire is a questionnaire developed to quickly gauge the complexity and ease of use of systems [6]. It consists of ten questions which are to be rated on a scale of one to five, with a score of one corresponding to 'completely disagree', and a score of five to 'completely agree'.

The questions used in the SUS questionnaire are as follows:

1. I think that I would like to use this system frequently.
2. I found the system unnecessarily complex.
3. I thought the system was easy to use.
4. I think that I would need the support of a technical person to be able to use this system.
5. I found the various functions in this system were well integrated.
6. I thought there was too much inconsistency in this system.
7. I would imagine that most people would learn to use this system very quickly.
8. I found the system very cumbersome to use.
9. I felt very confident using the system.
10. I needed to learn a lot of things before I could get going with this system.

The SUS questionnaire was designed to be quick and simple to use, allowing for participants to easily complete it, removing some frustration and effort that more complicated questionnaires might create. The SUS score of a system is calculated using the following formula:

$$SUS = 2.5 * (\sum_{i=\{1,3,5,7,9\}} (Qi - 1) + \sum_{i=\{2,4,6,8,10\}} (5 - Qi))$$

The SUS scores vary between the values zero to one hundred. The top fifty percent of systems have a score of 68 or higher, with the top ten percent having a score of 80.3 or higher.

There are downsides to the SUS questionnaire, however. [6] found that participants gave scores that would not always objectively reflect their experiences with the program. The participants' scores would vary based on how successful they were in completing their tasks, as well as on how tired they were, indicating several factors other than their actual opinions affected their given score.

This is one of several reasons why the resulting SUS score might not be indicative of the usability of a system. The lack of a truly objective score makes it unsuitable to truly test the usability. However, this does not mean that the SUS score has no value, as it can still represent how the users experience a system at that point in time.

Therefore, [6] found that the SUS score might be more useful as a benchmark for future iterations of the tested system. Instead of comparing the results with that of different systems, the results can instead indicate whether improvements were made since the last version.

CHAPTER 3

Implementation of an Audio Recording Function in 360Composer

360Composer had to be adjusted to facilitate the ability to record and store audio. This required adjustments to the database, the API controller and AFrame, the video-player, and it required the creation and usage of a recording script that would run client-side.

3.1 Client-side recording script

The client-side script was written in JavaScript. This allowed for a simple implementation that used the *MediaStream Recording API* to manually start and stop an audio recording. This script consists of several functionalities. First and foremost, the MediaRecorder selects the *MIME type*. This dictates what the recording format is, and how it is encoded. For the recording to be supported across as many browsers as possible, the *audio/webm* MIME type was used. This ensured that only the audio is recorded. Further options such as the bit-rate are set as well. All the data is stored in the *chunks* array, which is later converted into the correct format.

Once the *startRecord* function is called the correct settings for the recording are defined. Once the *mediaRecorder* is ready, it is started with a set amount of time to record. This ensures that the recording cannot continue indefinitely, which would cause large audio files to be created. Once the recording gets stopped manually, the surplus time left will remain unused.

The *mediaRecorder* makes use of two listeners: one is the *onDataAvailable* listener, and the other is the *onStop* listener. The first listener is used to continuously push any available audio data to the *chunks* array. The second listener gets called once the *mediaRecorder* is stopped, either manually or after the allotted time has expired. When the recording is stopped by the customer, the *chunks* array gets converted into a *Blob* object. This is an object which is able to be of a file-type not supported by JavaScript. The Blob object is put in a *FormData* object, which is used in API calls. The *FormData* object contains the method, the file, and can contain other arguments in a key-value pair format. This *FormData* object is sent to the *AFrame* controller, which sends it to the back-end.

A separate function has been made that gets called upon login of an user or customer. This function retrieves the local *mediastream*, causing a pop-up to appear which asks the user or customer if they want to give the website access to their devices' microphone. Calling this function upon logging in ensures that once the first scenario starts they will not be interrupted later when they need to record their audio.

3.2 Server-side adjustments

3.2.1 Front-end

Currently, the two most important scripts on the frontend are the *Viewing App* controller and AFrame. The controller is responsible for setting up the correct scenarios and scenes, and the *viewing app AFrame* is used to check when certain events should occur. In 360Composer, it is used to let the user start the video once it is loaded in. Once an annotation is reached, the AFrame pauses the video so the user can make their decision. After a decision has been made, the AFrame calls to the controller, which can then load in the corresponding scene. If no decision has been made, or the decision points to the end, the AFrame takes the user to the end screen.

However, because the next scene is immediately loaded in upon choosing an option in the annotation, the core functionality of the AFrame had to be changed.

The front-end required several adjustments in order to facilitate multiple annotations. Since the recording and playback annotations should not end the current scene entirely, the *AFRAME_Update* script needed a way to know whether or not it should continue the current video after the annotation. It needed to be able to distinguish whether the annotation ends the scene or not.

Viewing App Controller

The controller needed new functions to make the correct API calls to the back-end. These functions would be called by the AFrame, after which the front-end would call the correct *JavaScript* scripts. These new functions are called: *onStartRecording*, *onStartPlayback*, and *onFinishRecording*. *onStartRecording* would call to the *recording.js* script, which would allow the customer to start their recording. *onFinishRecording* would also call to this script, which in turn would return the *FormData* object, allowing the controller to upload the file to the back-end.

onStartPlayback would first make an API call to the back-end in order to retrieve the correct file path, which is then sent to the *playback.js* script.

Finally, one new state had been introduced to the controller: *allAnnotations*. This state stores all the annotations in a scene, which is then used by the AFrame to retrieve the second annotation in the event the first is a recording or playback type annotation. This state, which is an array of annotations, is set immediately after the annotation data of the scene has been requested.

Viewing App AFrame

In order to distinguish whether the next scene must be loaded after an annotation or not, the AFrame needed a check in the function which pauses the video upon reaching an annotation. This check would ensure the correct states were set to the correct values. These states are necessary for the AFrame to display the correct menus to the customer.

Once the first annotation is determined to be a recording annotation, the state *StartRecMenu* is set to *true*, which activates the menu that lets the user start their recording. Once the user starts the recording, the state *StopRecMenu* is set to true, displaying the button that triggers the *StopRecording* function, allowing the user to finish their recording at any time.

3.2.2 Back-end

Several scripts in the back-end needed to be edited or newly created to allow for the creation and usage of audio files. A new database model had to be created so the communication to the database could be executed. This model was subsequently called the *audioAsset*, which would allow for the API to retrieve the correct audio file entry in the database via it's compound key.

API calls

As a significant amount of communication was necessary from the front-end to the back-end, adjustments had to be made to the scripts handling API calls.

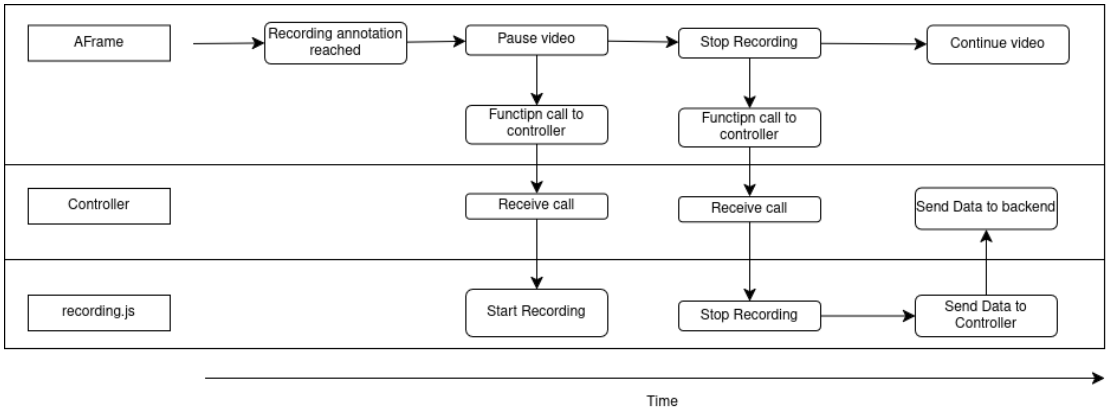

Figure 3.1: This illustrates how the front-end handles its function calls when a recording annotation is reached.

In the script *scenario.py* in the *routes* directory a new class has been added. This class, named *ScenraioCreateAudio*, allows the back-end to handle API calls from the front-end regarding the creation and retrieval of the audio files and the corresponding database entries. The *post* function handles the uploading of the created audio file by storing the file in the *audio_assets* folder in the *data* directory. It works practically identical to the function in the *ProjectAsset* class in the *project* routing script. This class is responsible for the uploading of 360-degree videos, a process that did not need any major adjustments to be made usable for audio files instead.

3.3 Database

In order to facilitate the implementation of the new API calls, the database had to be updated. In a new version script in the *migrations* directory the table *audio_recording* has been created. This new table contains a *path* column that stores the file path, and a compound key consisting of a main identifier, the customer's identifier, and the scenario's identifier. The reason behind this compound key is to ensure that there can be only one audio file per customer per scenario. Otherwise, it would complicate the retrieval of the audio files, as there would be ambiguity over which file to use.

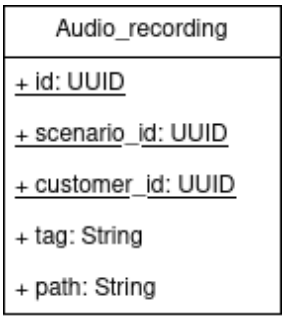

Figure 3.2: The new *audio_recording* table added to the database. Here the *id*, *scenario_id* and the *customer_id* for the compound key.

Furthermore, in the *annotation* table a new column named *tag* has been added. This column's value is set to an empty string by default, and allows for the API to store and retrieve the correct audio recording. To further facilitate this, two new annotation types have been created: *audioRecording* and *audioPlayback*. These new annotations can be used in the scene editor to create opportunities for the audio recording and playback. These annotation types were

necessary for the *AFrame* to detect which menu to display, and whether the video needed to continue playing after the annotation or not.

3.4 Client-side Playback Script

As mentioned before, the recording annotation makes use of a *tag*, which allows a playback annotation with the same tag to use the correct file. Once the AFrame reaches an annotation with the *playback* type, it will call upon a function from the controller. The controller will send a GET request to the api, which returns the file path to the correct audio file. The controller then gives this path as an argument to the *startPlayback* function in the *playback.js* file. This function simply starts the audio recording and nothing more. Once the customer has finished listening to the audio file, they can press the *stop recording* button to continue the scene.

CHAPTER 4

Experiments

4.1 Objective and Approach

The main objective of these experiments is to research the level of user friendliness of the audio recording function in 360Composer. In order to test this, the *System Usability Scale* questionnaire was used in order to gauge where there is still room for improvement.

First the participants are contacted and shown the ethical compliance form as seen in the appendix 6. When they agree to participate in the experiment, they are given access to the test version of 360Composer, which is running on a separate server of the University of Amsterdam. Once they have finished the created virtual reality experienced, they are asked to fill in the SUS questionnaire, which is accompanied by additional questions. These additional questions are as follows:

- Question 11: This system made me think about my own behaviour.
- Question 12: This system gave me a feeling of discomfort during use.
- Question 13: This system gave me a clearer view on how others see me.
- Question 14: This system has influenced my behaviour in the future.

Whilst these questions are not suitable for any definitive statements about the immersion of the perspective taking aspect of the recording function, they are instead chosen to generally gauge how the recording function might be used for more behavioural research in future work.

Finally, the last section of the questionnaire is a space where participants can share their thoughts, both positive and negative, on the system that they felt has not been represented by the questionnaire. This has intentionally been made optional, as to not force the participants to give an opinion they do not feel strongly about. This allowed them to make suggestions on how the recording function might be improved, as well as share any difficulties they encountered during the experiment.

4.2 Participants

Aside from being 18 years and older, there were no general requirements made from the participants. They needed to be able to read English, and understand and speak Dutch for the experiments. This is due to the fact the ethical compliance form was written in English, whilst the videos recorded for the experiment and the questionnaire were in Dutch. The participants consisted of mostly students from varying studies, along with three middle-aged adults.

4.3 Procedure and Testing Environment

The application will be tested by letting a group of fifteen participants try the recording functionality out and filling in a *System Usability Scale* (SUS) questionnaire. They will all be given the same scenario, after which they can fill in the questionnaire either online or on a physical sheet. The scenario will feature a social situation where the participant will be asked to respond vocally, after which a perspective switch will take place. During this, the video will pause at the same point, playing back the audio file.

The participants will be given the following scenario: They are at a job interview for an undisclosed position. They meet the interviewer, who introduces himself and asks three general questions. These questions are: "tell me a little about yourself", "imagine you are working in a group project where one colleague does not fulfill his tasks making you unable to continue. how would you handle it?", and "if you were in a project group with only clones of yourself, what would go well and what would not?" These questions were chosen because they elicit longer answers, with the last two being somewhat unconventional.

The results of the questionnaire will remain anonymous, as their personal information has no effect on the test itself. After the questionnaire, the participants will be given an opportunity to give any other remarks, outside of the questionnaire. This will give a more personal input on how the recording function is experienced, and they will be able to give suggestions on how they would have preferred certain aspects.

The official version of 360Composer that is currently running on the server does not contain this recording function. Considering the version of 360Composer with the audio recording function is only available on a personal laptop, the tests will have to be done in person. This way, the audio recordings of the participants can also immediately be deleted upon finishing the test. This will further ensure the privacy of the participants.

4.4 General Results

The version of 360Composer containing the created audio recording function is publicly accessible and can be found at [3]. This section will go into the results from the questionnaire, starting with the questions belonging to the SUS quetsionnaire, followed by the added questions.

4.4.1 SUS Results

A bar chart was made for the total SUS score received from each participant, which is visible at 4.1. A table of all the results per participant can be seen in 6.

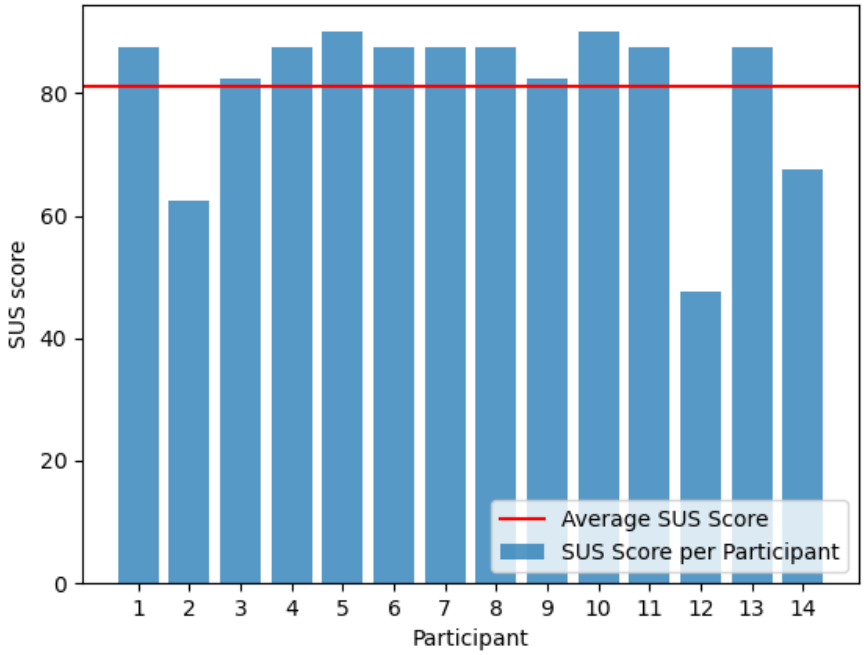

Figure 4.1: The resulting SUS scores for each participant. One participant has been left out, as they had forgotten to answer one question.

This chart indicates that most participants gave the system a score higher than 80. This would mean that the recording function could be considered user friendly. There is a notable outlier in the SUS scores, which is likely given by one participant who reported not understanding how the system worked and not being able to hear their voice played back at them.

Only the SUS scores of fourteen participants are displayed, as one participant forgot to answer one question in the form. This led to the decision to not count that participant in the compilation of the average SUS score, as it was otherwise uncertain how to replace the missing answer.

Individual questions

Here the results for the SUS questions themselves is displayed as a graph containing the average scores and standard deviations 4.2.

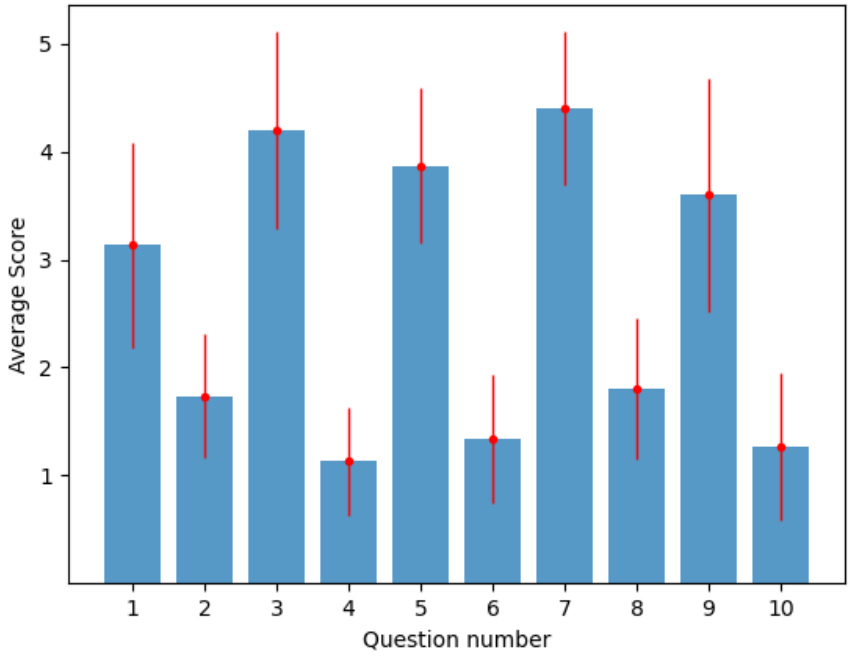

Figure 4.2: The mean scores given by the participants for each question, alongside the standard deviation depicted in red.

The following will refer to the total results as displayed in 6.

It would seem that the participants' willingness to use the system frequently varies. Whilst a large portion neither agreed nor disagreed with the first question, the average score seems to be between three and four, meaning that the participants would generally only slightly like to use the recording function frequently. Furthermore, it would seem that most users did not find the recording function too complex, with the majority disagreeing. None of the participants seemed to agree with the statement.

The majority of the participants seemed to agree that the system was easy to use. Furthermore, a grand majority of the participants completely disagreed with the statement that they would need the help of a technical person to be able to use the recording function. These two scores indicate that the recording function was easy to understand and use.

Most users found the functions to be well integrated. With a vast majority giving the statement a score of four, meaning they agree somewhat. Secondly, most participants found that the recording functionality was very consistent.

Most participants completely agreed that people would be ably to quickly learn how to use the recording function. Furthermore, almost no participants found it cumbersome to use.

Whilst the results are varied, the vast majority of participants indicated that they felt confident using the recording function. With nine participants giving a score of four, it would seem that there might still be room for improvement. Furthermore, with the exception of two participants, most found that they did not need to learn a lot before they could use the recording function.

One participant forgot to answer one of the SUS questions, so in order to calculate the average SUS score that participant was left out. This was because it was uncertain what might have been the correct course of handling otherwise. To use the average score for that particular question might have shifted that participant's SUS score to either a higher or lower value.

The mean SUS score from these experiments is an 81.07, which is in the range of what would be considered user friendly. The lowest SUS score from a participant is a 40, and the highest SUS score is a 90. Most SUS scores stay in the 80 to 90 range, with the mode being a score of

87.5.

4.4.2 Extra Questions Results

The extra questions made to further test the effect of the recording function were not taken into account for the calculation of the SUS score. Instead, they are each analysed separately, and are only used to test the immersion and effect of the recording function to the participants. The full results of these questions can be found in the appendix at 6. These graphs indicate the frequency each participant gave a certain score to the question.

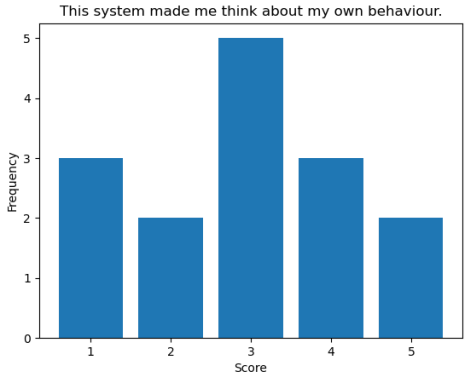

Figure 4.3: The results of question 11.

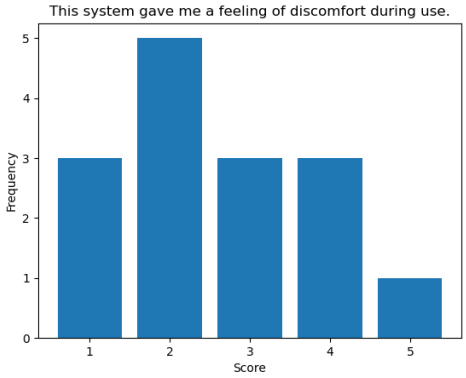

Figure 4.4: The results of question 12.

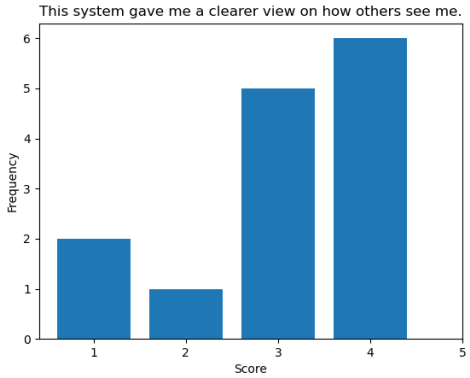

Figure 4.5: The results of question 13.

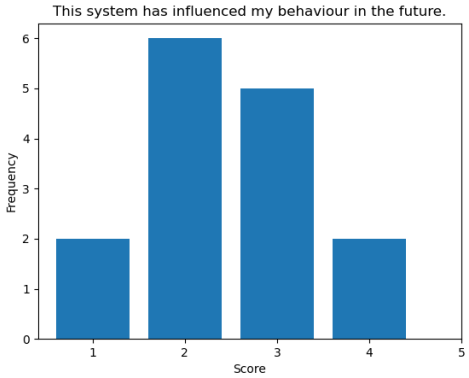

Figure 4.6: The results of question 14.

The results of these questions vary more than those of the SUS questionnaire. Whilst some participants indicated that recording their voice and hearing it played back was uncomfortable, others were unbothered. Whilst not shown in the graphs, some participants indicated that they were feeling nervous being put in a job interview scenario, but less so due to it being a virtual reality scenario. The variation in responses could be caused by the different backgrounds of the participants. Approximately half of the participants were either students or graduates from an IT study, whilst the other half were from other fields. It is possible that the participants with an IT background were more used to testing similar types of applications, making them less emotionally influenced by them.

4.5 Specific Remarks

The specific remarks question had been purposefully left optional, as to not force the participants to come op with a remark. In total, 8 out of 15 participants left a specific remark.

The results have been translated from Dutch as accurately as possible:

- "I found the 'next' button nice, because it allows you to take the time to calm down, even more so than in a real conversation."
- "It is easy to learn how to use."
- "Two things: First of all, it was a bit odd that there was always a small bit of video playing after the 'stop recording', but that seems more of a 360Composer issue in general. Secondly, the audio kept playing if I went to the next phase early. Especially after the restart my recording played simultaneously with Rob."
- "The 'buttons' that you had to press to continue took away from the realness of the situation, especially upon playback."
- "The system has many application possibilities."
- "At a certain point in the conversation i had accidentally stopped one of the questions too early, but I kept hearing my voice while Rob continued with the next question. Aside from that it is a very intuitive system."
- "Sometimes I accidentally aimed at 'stop recording' whilst talking. Luckily I was paying attention and it didn't stop. It seemed like sometimes part of my word dropped out, but that may be due to my phone? I found it a very interesting experiment and I am very curious about your findings! I hope we could have something like this in the future. It almost felt like a real conversation!"
- "The usage of looking at the button to press it can cause issues. It might make you skip something by accident, and people with a handicap might have issues positioning the device."

The specific remarks have a large variety of content. Some indicated problems that arose during the experiment, whilst others expressed the positives of the system.

4.6 Further Notes

Whilst the first half of the participants experienced no issues during the experiment, the second half had problems. It would seem that 360Composer generated invalid URLs for them, causing their screen to go white upon clicking start after logging in. Once older participants' access were removed from the timeline editor and new URLs were generated, they were able to access the scenario. It is uncertain what caused this problem. Almost all participants were able to complete the experiment once they were given a valid URL.

However, one participant was unable to complete the experiment due to unknown reasons. With both their mobile device as well as their laptop they were unable to start the experience after logging in. They reported only seeing a white screen upon clicking start after logging in, with no improvements once given valid URLs.

One other participant reported that they heard nothing upon playback when they tried the experiment on their mobile device. They indicated that it was an older model, so it might be possible that 360Composer was unable to access their microphone. Once they retried the experiment on their laptop, they were successfully able to record their voice and hear it being played back to them later.

CHAPTER 5

Conclusions

The main research question of this project was:

- How can we implement an au dio recording function in 360Composer?

with the following sub-questions:

- How can we record and play back audio in 360Composer?
- How can we store and retrieve the audio recordings?
- How do users experience the audio recording function in 360Composer?

Ultimately, a functional audio recording and playback feature was successfully created for 360Composer. Users were able to easily understand how the functionality worked, and were able to successfully complete the experiment. several changes had to be made to the already existing code. One of which was to allow for multiple, usable annotations per scene, which in turn allowed for the *AFrame* updater to correctly handle the function calls and app states. By using the *MediaRecorder* API for JavaScript, this project was successfully able to obtain access to the customers' microphones and use them to record the audio.

By creating new tables in the database of 360Composer, we were able to successfully track which audio recordings belonged to which customers. By using the new *tag* system for the annotations, it was possible to link playback annotations to their corresponding recordings. This allowed for customers to create several audio recordings per session, which each recording being able to be played back at the indicated timestamps. Once a playback annotation was reached, the front-end controller was able to request the correct audio recording. This was made possible with an update to the *API* handler on the back-end, which would receive a *GET*-request from the controller with a given tag and scenario ID. Since the back-end could access the customers' ID, it is able to retrieve the correct filepath to the recording from the database by combining the tag, scenario ID, and customer ID. This filepath would then be sent back to the controller, which in turn activated the playback function with this filepath.

With regards to the third sub-question, the participants seemed rather positive about the audio recording function. With an average SUS score of 81, it would be deemed an user friendly functionality. There is room for improvement, however. The results indicate that the recording and playback functions could be more streamlined, with the participants' extra remarks highlighting that the frequent pausing of the experience to press a button reduces the immersion they experience. This is reflected in questions two, three, and eight. The results of question two, for example, indicated that whilst the recording functionality is not too complex, they did not fully disagree with the statement. Most users gave it a score of two, meaning there is some level of complexity that might be mitigatable.

Whilst [10] and [5] showed a notable change in the behaviour of participants in their perspective taking experiments, these results were not necessarily recreated in this project. Most participants indicated that the experiment would not affect them in future situations. Some

neither agreed nor disagreed, with a minority of 2 participants indicating that this experiment would affect their behaviour in the future. The participants did report that the experiment gave them a somewhat clearer view on how other people might see them.

5.1 Discussion

The current implementation of the audio recording function still has room for improvement however. As several participants have indicated, the decision to pause the video upon recording and playback comes at the cost of the authenticity of the scenario. Participants have reported a loss of immersion due to the pausing, as well as due to the fact that they have to press a button before they can start the recording. While these two problems are simple to correct, it comes at the cost of a positive side effect; one participant reported that they preferred the pausing, as it allowed them to calm down and get their bearings better, making the virtual reality training experience preferable over a real life training scenario. Whilst it is possible to add an option that allows the user to indicate whether the video pauses or not, it might further complicate the scene editor. This would directly contradict one of the main requirements of the audio recording function, namely that it must be simple to use.

Furthermore, the issues that arose during testing should be looked into. The current theory is that 360Composer stops generating valid URLs once too many customers have access to the timeline. This actively hindered the experiment, as participants were unable to begin the scenario. Finally, more research is necessary with regards to which versions of both browsers as well as operating systems are capable of running 360Composer.

As the specific remarks indicated, opinions between the participants vary on how immersive the experience was. For future study one might fully research the level of immersion experienced by participants.

Furthermore, since opinions varied between the users on whether pausing the video upon recording and playing back is preferable, researches could investigate how to further improve the current implementation in order to allow for a continuous experience. This would require some considerations on how much freedom they would want to give the participants with regards to time limits. The researchers would also have to consider the varying durations of the audio recordings, and how to anticipate those when recording the 360-degree videos in advance. Once a solution has been found that allows for recording and playback whilst the video keeps playing, they can research the advantages and disadvantages compared to pausing the video instead.

Bibliography

[1] URL: https://docs.fileformat.com/audio/mp3/.

[2] *30 law enforcement learn with VR tools how to combat drug production crimes*. 2024. URL: https://www.cepol.europa.eu/newsroom/partners-news/30-law-enforcement-learn-vr-tools-how-combat-drug-production-crimes.

[3] *360Composer with audio recording function*. URL: https://github.com/robert-belleman/360Composer/tree/kim-project.

[4] Jesse van den Berge. "Ontwerp van een behandelmethode voor selectief mutisme op basis van 360 graden video's". Bachelor's Thesis. Universiteit van Amsterdam, 2021.

[5] Cassandra L. Crone and Rachel W. Kallen. "Interview with an avatar: Comparing online and virtual reality perspective taking for gender bias in STEM hiring decisions". In: *PLOS ONE* 17.6 (June 2022), pp. 1–35. DOI: 10.1371/journal.pone.0269430. URL: https://doi.org/10.1371/journal.pone.0269430.

[6] Mandy R. Drew et al. "What Does the System Usability Scale (SUS) Measure?: Validation Using Think Aloud Verbalization and Behavioral Metrics". eng. In: *Design, User Experience, and Usability: Theory and Practice*. Lecture Notes in Computer Science. Cham: Springer International Publishing, 2018, pp. 356–366. ISBN: 331991796X.

[7] Hunter Gehlbach, Bryan Mascio, and Joseph McIntyre. "Social Perspective Taking: A Professional Development Induction to Improve Teacher-Student Relationships and Student Learning". eng. In: *Journal of educational psychology* 115.2 (2023), pp. 330–348. ISSN: 0022-0663.

[8] Jeffrey C. F. Ho and Ryan Ng. "Perspective-Taking of Non-Player Characters in Prosocial Virtual Reality Games: Effects on Closeness, Empathy, and Game Immersion". eng. In: *Behaviour & information technology* 41.6 (2022), pp. 1185–1198. ISSN: 0144-929X.

[9] Deirdre Kavanagh, Yvonne Barnes-Holmes, and Dermot Barnes-Holmes. "The Study of Perspective-Taking: Contributions from Mainstream Psychology and Behavior Analysis". eng. In: *The Psychological record* 70.4 (2020), pp. 581–604. ISSN: 0033-2933.

[10] Austin van Loon et al. "Virtual reality perspective-taking increases cognitive empathy for specific others". In: *PLOS ONE* 13.8 (Aug. 2018), pp. 1–19. DOI: 10.1371/journal.pone.0202442. URL: https://doi.org/10.1371/journal.pone.0202442.

[11] Gonzalo Lorenzo et al. "Design and application of an immersive virtual reality system to enhance emotional skills for children with autism spectrum disorders". In: *Computers Education* 98 (2016), pp. 192–205. ISSN: 0360-1315. DOI: https://doi.org/10.1016/j.compedu.2016.03.018. URL: https://www.sciencedirect.com/science/article/pii/S0360131516300811.

[12] Elizabeth. McMahon and Debra. Boeldt. *Virtual Reality Therapy for Anxiety : A Guide for Therapists*. eng. 1st ed. Oxford: Taylor & Francis Group, 2010. ISBN: 9781000473759.

[13] Yupeng Mou et al. "Impact of virtual reality immersion on customer experience: moderating effect of cross-sensory compensation and social interaction". eng. In: *Asia Pacific journal of marketing and logistics* 36.1 (2024), pp. 26–47. ISSN: 1355-5855.

[14] Pelagia Papathomas and Karen Goldschmidt. “Utilizing virtual reality and immersion video technology as a focused learning tool for children with autism spectrum disorder”. In: *Journal of Pediatric Nursing* 35 (2017), pp. 8–9. ISSN: 0882-5963. DOI: https://doi.org/10.1016/j.pedn.2017.01.013. URL: https://www.sciencedirect.com/science/article/pii/S0882596317300349.

[15] *The WebM Project.* URL: https://www.webmproject.org/about/.

[16] *What is Matroska.* URL: https://www.matroska.org/what_is_matroska.html.

CHAPTER 6

Appendix

Participant	Q1	Q2	Q3	Q4	Q5	Q6	Q7	Q8	Q9	Q10	SUS Score
P1	3	2	5	1	4	1	5	2	5	1	87.5
P2	2	2	3	2	4	2	4	3	2	1	62.5
P3	4	2	5	1	4	2	4	2	4	1	82.5
P4	4	2	3	1	5	1	5	1	4	1	87.5
P5	3	1	5	1	4	1	5	1	4	1	90
P6	5	2	5	2	4	1	4	1	4	1	87.5
P7	4	2	5	1	4	1	5	2	4	1	87.5
P8	3	1	5	1	4	1	5	2	4	1	87.5
P9	4	1	4	1	3	1	4	2	4	1	82.5
P10	3	1	5	1	4	1	5	1	4	1	90
P11	3	2	4	1	5	1	5	1	4	1	87.5
P12	2	3	3	2	2	3	3	3	3	3	47.5
P13	1	2	3	-	4	1	4	2	1	3	-
P14	3	1	5	1	3	1	5	2	5	1	87.5
P15	3	2	3	1	4	2	3	2	2	1	67.5
Average Score	3.133	1.733	4.2	1.214	3.867	1.333	4.4	1.8	3.6	1.267	81.071

Table 6.1: The results of the SUS questionnaire. One participant forgot to answer a question, hence that question's value is marked with '-', and their SUS score is neither calculated nor included in the average SUS score.

UNIVERSITEIT VAN AMSTERDAM

Ethical Compliance Procedure for Low Risk Student Projects in Informatics involving Human Participants

Introduction
Some informatics projects involve research with human participants. For example, researchers may conduct interviews or focus groups with potential users to gather requirements. They might also want to evaluate new technology designs, prototypes, or digital devices with users.

It is crucial to conduct such research in an ethical manner. Participants must be treated respectfully, fully informed about the study before they agree to take part, allowed to withdraw at any time, and protected from harm or coercion. Additionally, any identifiable information must be anonymized, and study data stored securely.

Research with human participants carries different levels of risk. The majority of Master's and Bachelor's projects that involve human participants will be low risk. This ethical compliance procedure has been developed for these types of projects. It comprises an ethical compliance form to be read and signed by you and your supervisor. If you, and your supervisor, can answer "yes" to all of the statements, then your project is indeed low risk.

However, certain studies—such as those involving children, sensitive topics, or potentially stressful situations—are considered to be high risk. In these cases, a full ethical review by the ECIS-SP (Ethics Committee Information Sciences – Student Projects)[1] must take place before the research can be conducted.

Who is this form for?
This form is for Bachelor's and Master's students whose projects involve research with human participants. If your project does not involve human participants, then you do not need to complete this form. If you are a PhD candidate or staff member, then you should also not use this form.

How to use this form?
For any project involving human participants, the student and their supervisor should go through the form together, point by point. Assuming the project complies with all of the points, then the project is low risk, and the form should be signed by both the student and supervisor and included as an appendix to the final report/thesis.

The student must also prepare an information sheet and consent form for participants using the template provided at the end of this document. The supervisor should check and approve the information sheet and consent form before starting the study.

NB: If the student and supervisor determine that the research does not comply with all points (or if the research changes in such a way that it no longer complies with the points), then the student cannot use the ethical compliance procedure and must instead apply for ethical advice.

[1] Use the following link to access the FNWI Research Management Portal, where you can apply for ethical review of your project https://rms.uva.nl/servicedesk/customer/portal/14. Please ensure you select "I am a student" so that the application is routed to the ECIS-SP.

UNIVERSITEIT VAN AMSTERDAM

Ethical Compliance Form for Low Risk Student Projects in Informatics involving Human Participants

In signing this form, I and my supervisor confirm that my project complies with all of the following points:

- Participants will not be exposed to risks greater than those encountered in daily life.
  Researchers must protect participants from physical, mental and emotional harm.
  Ethical advice must be sought for any projects involving sensitive topics (e.g. sexual activity, drug use, political behaviour, ethnicity, or discrimination) or which might induce discomfort, stress or anxiety (e.g. violent video games, bright or flashing lights, loud noises).

- Any study materials used by the participant should be either paper-based or involve software running on commonly available, commercially produced hardware (e.g., personal computers, laptops, tablets, or smartphones).
  Participants should not be exposed to risks associated with using specialised, experimental, or custom-built equipment. If biometric data (e.g., eye movement, skin temperature, or heart rate) is collected, ethical advice must be sought.

- No information about the evaluation or materials is intentionally withheld from the participants.
  Withholding information from participants or misleading them is unacceptable without justifiable reasons for doing so. Any projects requiring deception (for example, only telling participants the true purpose of the study afterwards so as not to influence their behaviour) are deemed high risk and require ethical advice.

- No participant is under the age of 18.
  Any studies involving children or young people under the age of 18 are deemed to be high risk and require ethical advice.

- No participant has a disability that limits their understanding of the project or capacity to consent.
  Any studies involving participants with disabilities that impact their capacity to consent are deemed to be high risk and require ethical advice.

- Neither I nor my supervisor are in a position of authority or influence over any of the participants, and participants were not pressured by anyone (e.g. employer) to take part in the study.
  A position of authority or influence over any participant must not be used to pressure participants to take part in, or remain in, any study.

- Participants receive an information sheet which contains the following information:
  - *A detailed description of the study*
  - *A statement informing participants that they can withdraw from the study at any time*
  - *A statement informing participants that they can withdraw their data after the study (typically there is a cut-off date once data analysis begins: this should be specified)*
  - *A confirmation that any data collected will be stored securely (i.e. locked filing cabinets for hard copy, password protected computer for electronic data), and in an anonymous form*
  - *The contact details of both the student and the supervisor.*

- Participants sign a consent form, explicitly stating that they agree to take part, and that their data can be used in the project.
  Participants must not take part in the study without their knowledge or consent (i.e. no covert observation: this is considered to be high risk).

UNIVERSITEIT VAN AMSTERDAM

Project title: Audio Recording Function for Perspective Taking in 360Composer
Student Name: Kimberley Roks

Student Registration Number: 14617811

Student Signature:

Date: June 12, 2025

Supervisor Name: dr. Robert G. Belleman

I have read and approved the participant information sheet and consent form.
Supervisor Initials: RGB

Supervisor Signature:

Date: June 12, 2025

UNIVERSITEIT VAN AMSTERDAM

Information Sheet and Consent Form Template (for Low Risk Informatics Student Projects with Human Participants)

Invitation to take part

You are being invited to take part in a research study to further our understanding of the usability of 360Composer. Thank you for carefully reading the following information. This study is being conducted by student researcher Kimberley Roks doing a degree in Bachelor Informatica at the UvA.

What is the purpose of this research?

To evaluate the usability of 360Composer.

Who can take part?

Everyone over the age of 18.

What is involved in taking part?

1. You will be given a short explanation about 360Composer and what the specific experiment entails.
2. You will be asked to click on a link using a mobile device such as a phone. This will open 360Composer.
3. You will be asked to log in with the given access code.
4. You will be presented with a scenario where at some points you will be requested to say something.
5. Finally, we ask you to answer a number of questions using a digital form.

In total, this will take 10 to 15 minutes of your time.

Is participation voluntary?

Your participation is voluntary and you can choose to take part in only certain parts of the study, or not at all. You may withdraw from the research at any time and without providing a reason, in which case, we will not use your data and will destroy it immediately. After having taken part in the study, you can still withdraw consent and request the removal of your data up until the start of data analysis (June 20, 2025). To do so, please email kim.roks@student.uva.nl and include your full name).

How will my data be treated?

During this experiment we will record your voice. The audio files acquired will be permanently deleted upon completion or termination of the experiment.

Your answers to the questions will be anonymized such that it is not possible to identify you. This data will be stored securely (e.g. for paper-based data, in a locked filing cabinet or for digital data, on an encrypted or password-protected server). Furthermore, no data (anonymised or otherwise) will be used as input to Generative AI tools (e.g. ChatGPT). In line with GDPR and relevant regulations, the data will be kept for one year after data collection, after which it will be destroyed.

Questions or Concerns:

If you have any questions about the study, please feel free to contact the student researcher: kim.roks@student.uva.nl. You can also contact the student's supervisor: dr. Robert G. Belleman, r.g.belleman@uva.nl.

Declaration of Consent

I confirm that I:

- am 18 years or over
- have read and understood the provided information
- agree to participate in this study
- consent to the use of the resulting data as described above
- consent to the research team carrying out further analysis of my **anonymised** data (tick here if you agree) ____

....

Name of participant Signature of participant

.....

Date

Participant	Q11	Q12	Q13	Q14
P1	1	3	3	2
P2	2	4	3	2
P3	3	2	3	3
P4	4	1	2	2
P5	1	3	3	2
P6	5	2	4	4
P7	4	4	4	3
P8	1	1	-	1
P9	5	3	4	4
P10	2	2	1	1
P11	3	2	4	3
P12	4	4	3	3
P13	3	5	1	2
P14	3	1	4	3
P15	3	2	4	2
Average Score	2.933	2.6	3.071	2.467

Table 6.2: The results of the extra four questions.