FH Salzburg
MultiMediaTechnology

Navigation and Manipulation of Multiple Live Viewpoints for Virtual Reality Head-mounted Displays in a Museum Setting

Master thesis in partial fulfillment of the requirements for the degree of

Master of Science

Author: David Märzendorfer

Submitted to the Master degree program MultiMediaTechnology, Salzburg University of Applied Sciences

Advisor: FH-Prof. DI Dr. Markus Tatzgern, Bakk.

Salzburg, Austria, 01.12.2024

Affidavit

I herewith declare on oath that I wrote the present thesis without the help of third persons and without using any other sources and means listed herein; I further declare that I observed the guidelines for scientific work in the quotation of all unprinted sources, printed literature and phrases and concepts taken either word for word or according to meaning from the Internet and that I referenced all sources accordingly.

This thesis has not been submitted as an exam paper of identical or similar form, either in Austria or abroad and corresponds to the paper graded by the assessors.

SIGNATURINFORMATIONEN	Unterzeichner	David Märzendorfer
	Datum/Zeit-UTC	2024-12-01T22:47:41+0100
	Prüfinformation	Informationen zur Prüfung der elektronischen Signatur finden Sie unter: https://www.signaturpruefung.gv.at
Hinweis	Dieses mit einer qualifizierten elektronischen Signatur versehene Dokument hat gemäß Art. 25 Abs. 2 der Verordnung (EU) Nr. 910/2014 vom 23. Juli 2014 ("eIDAS-VO") die gleiche Rechtswirkung wie ein handschriftlich unterschriebenes Dokument.	

01.12.2024
Date

Signature

David Märzendorfer
First Name *Last Name*

Abstract

Multiple viewpoints in virtual reality (VR) enhance users' ability to explore and analyze environments by enabling the placement and manipulation of viewpoints in 3D space. Museums provide an ideal context for this technology, allowing visitors to examine fragile or distant artifacts in detail without risking damage. Multiple viewpoints can open new possibilities for exhibit designs, such as large-scale dioramas. This study evaluates two distinct approaches for viewpoint manipulation in a museum setting, comparing them in terms of usability, cybersickness, and task performance. The first is a restricted, automated, and assisted object-centric exploration (OCE) approach, represented by a customized HoverCam for VR. The second is a free-movement approach using a flying camera akin to a first-person-view drone. Findings indicate that the OCE approach is more effective for detailed object inspection, while the free approach excels in exploring larger scenes. Based on these results, a hybrid camera system is proposed to leverage the strengths of both methods. The sense of embodiment (SoE) in the free approach is analyzed concerning the impact of control schemes on the sense of body ownership (SoBO). The unique nature of multiple viewpoints in VR allows users to observe themselves from different angles while controlling a viewpoint, often requiring frequent context switches. While no significant correlation between control schemes and SoBO was observed during the study, this requires further research for greater insight.

Keywords: virtual reality, viewpoints, museum, HoverCam, embodiment, drone

Table of Contents

List of Figures

Listings

List of Tables

Abbreviations

AR Augmented Reality

OCE Object-Centric Exploration

XR Extended Reality

VR Virtual Reality

MR Mixed Reality

POI Point of Interest

POIs Point of Interests

FOV Field of View

POV Point of View

WIM	World in Miniature
HUD	Head-up display
UI	User Interface
HCI	Human-Computer Interaction
1stPP	First Person Perspective
3rdPP	Third Person Perspective
HMD	Head-mounted Display
SoE	Sense of Embodiment
SoA	Sense of Agency
SoBO	Sense of Body Ownership
SoSL	Sense of Self-Location
VRSQ	Virtual Reality Sickness Questionnaire
SUS	System Usability Scale
NASA TLX	NASA Task Load Index
IK	Inverse Kinematics
AI	Artifical Inteligence
IPQ	IGroup’s Presence Questionnaire
GEQ	Games Experience Questionnaire

1 Introduction

Multiple viewpoints in Extended Reality (XR) describe the process of not having just a single perspective but multiple perspectives of an environment instead. This promises to allow for a greater understanding and overview of an environment. For example, extra viewpoints can reveal POIs of an object that might not be in view otherwise and help with occlusion as described by Elmqvist and Tsigas (2008). One application that makes use of multiple viewpoints is remote collaboration. Additional viewpoints are used to show the perspective of remote collaborators (Zaman, Anslow, and Rhee 2023). Most research on multiple viewpoints and perspectives is done in VR. For example, Hoppe et al. (2022) examines perspectives in VR and the effect of viewpoints on the users' embodiment. They suggest a Perspective-Continuum in VR which describes that first-person perspectives lead to a greater amount of embodiment than third-person perspectives. Users with a third-person perspective feel more like spectators than they would in a first-person perspective. Another research on viewpoints is by Unruh, Lugrin, and Latoschik (2024) that researches the effect of different viewpoints on the perception of time in VR.

VR lends itself to perspective and viewpoint changes since it is easy to interpolate viewpoints and all environmental information is available. This thesis utilizes this for VR prototyping of multiple viewpoints in a museum setting. The idea is to give visitors of a museum the option to view an exhibit from any perspective of their choosing. A real implementation of such a use-case in Mixed Reality (MR) could be realized with the usage of multiple depth-cameras creating a live 3D model of an environment that can be used for viewpoint interpolation. Lindlbauer and Wilson (2018) have already experimented with such a setup for live 3D models using static kinect cameras and the RoomAliveToolkit[1]. Another approach comes from Zhang et al. (2021) using drones. To narrow down the scope however, this thesis only focuses on a VR prototype with the goal of gaining knowledge on how to best setup and manipulate viewpoints from a distance. This thesis does not focus on transitions between viewpoints, selection and deletion of viewpoints. There already is much research being done on viewpoints transitions such as Cmentowski et al. (2023) in regard to games and story telling and even transitions from VR to Augmented Reality (AR) by Pointecker et al. (2022).

Besides research on transitions there are already approaches for the manipulation of distant objects in VR. Some of these approaches are Voodoo dolls by Pierce, Stearns, and Pausch (1999) which suggests the usage of virtual copies of objects that are projected closer to the user. Or the efficient placement of distant objects in AR by Chae, Hwang, and Seo (2018) allows the user to dynamically squeeze their surrounding space in order to reach further. Another approach are go-go hands by (Poupyrev et al. 1996) that allow the user to shoot out their hands and reach further in order to interact with objects. Furthermore, there exists a proxy based approach for interaction and manipulation of distant objects by Pohl et al. (2021).

However, not every method is applicable for viewpoint manipulation since viewpoints are not the same as normal objects. For example the scale of a viewpoint does not have an effect on a viewpoint. Position and rotation do however. Note here, that one could link a viewpoints scale to its Field of View (FOV), albeit this thesis does not do so.

1. https://github.com/microsoft/RoomAliveToolkit

In order to gain further insight on how to best manipulate viewpoints in a museum setting two approaches of manipulation are reviewed in more detail.

1.1 Object-Centric Exploration

The first approach is one of OCE which focuses on an object in its center. This approach is based on the work of Tatzgern et al. (2015). The object in question here is an exhibit in a museum. With this approach the user selects an exhibit and is able to create a HoverCam (Khan et al. 2005) camera facing the object. Whilst not going into detail on the selection of different viewpoints, part of this approach is however the selection of the object the HoverCam centers on. This simply done via a ray cast, in addition the HoverCam can quickly be focused on another object with an additional ray cast, similar to the 'look-and-fly' feature described by McCrae et al. (2009). Overall this system focuses on a more automated and restricted approach.

1.2 Free Exploration

This approach tries to give the user more freedom in choosing and placing the viewpoint camera. For this a mental model of drones as viewpoints is suggested. This is not meant to suggest the usage of drones in a museum. The mental model of drones is merely meant as a way to seed the correct conceptual model to the user. As explained by Norman (2002) mental models help the user understand a system. People form mental models through experience, training and instruction. Good designed solutions make sure that users have the correct mental model since misalignment on how users think a system works and how the system works in reality can lead to terrible usability. For this reason the drone analogy was chosen, since the viewpoints behave like remote controlled drones with a camera on them. The user can select a drone and navigate it to a different position and rotation. To maximize the potential of this approach it features two Control-Schemes which are examined in more detail in regard to their effect on SoE and Sense of Body Ownership (SoBO) as well. Kilteni, Groten, and Slater (2012) describe embodiment to partly consist out of SoBO. SoBO is described as one's self-attribution of a body (Gallagher 2000). Kilteni, Groten, and Slater (2012) furthermore state that SoBO can be enhanced by: "increasing the sensory correlations between the physical stimulation of the biological body and the seen stimulation on the avatar's body." Therefore even just a change in Control-Schemes could lead to greater SoBO since it might change this correlation between physical stimulation and observed stimulation of the avatar.

These two methods are compared in a user study because they represent two distinct approaches. One method is *restricted, automated and assisted*, making movement easier but potentially limiting the user's freedom to view objects from certain angles. The other method offers *complete freedom*, giving users full control to look at whatever they want, but it may be slower and more demanding since the user has to make all adjustments manually. In addition, the drone approach, with its complete freedom, is examined in regard to the influence of Control-Schemes on the SoE and SoBO.

1.3 Museum Setting

The museum setting was intentionally chosen for this research, as it provides an environment characterized by differently sized objects arranged within a confined space. Such a setting inherently imposes certain physical limitations on user movement, making it an ideal scenario for exploring the potential of multiple viewpoints. Moreover, museums naturally cater to user curiosity, as visitors often wish to view exhibits from a variety of angles and perspectives. This makes them an excellent testing ground for viewpoint manipulation. Beyond this primary use, viewpoint manipulation in museums could also be applied to enhance guided tours. For example, a tour guide might use this functionality to showcase a specific Point of View (POV) of an exhibit to visitors, akin to how viewpoints are leveraged in remote collaboration, as discussed by Zaman, Anslow, and Rhee (2023).

Museum objects, also known as musealia, come in a wide range of shapes, sizes and dimensions, each presenting unique challenges for effective display. Museums must carefully consider how to position and present these objects to maximize the educational or aesthetic impact of their exhibits. Allowing guests the freedom to choose their own viewpoints could revolutionize the way objects are displayed, offering a more interactive and personalized experience. For additional insights into museum objects and exhibit design see Desvallées and Mairesse (2010).

Delgado et al. (2024) investigate the potential of VR in museums to create new opportunities for interactive and immersive experiences. They developed an interactive VR game for the Florida Natural History Museum, leveraging the museum's extensive 3D digital collection. In this game, visitors can swim alongside endangered underwater species in a simulated ocean environment, with the musealia presented in appropriate ecological contexts. Visitors also receive scientific insights about these creatures through a conversational Artifical Inteligence (AI) agent. Their research emphasizes the importance of making such experiences shared among visitors. For instance, they suggest displaying the VR experience on a TV screen to allow others to observe and promote social interaction.

It is crucial to distinguish that this paper focuses on the application of VR in museum settings to create interactive experiences. In contrast, this thesis explores the manipulation of viewpoints in AR for museums by prototyping in VR. However, the insight from Delgado et al. (2024) regarding the importance of shared experiences is applicable to both contexts.

Other research on VR in museums includes the work by Cao et al. (2023), who explore virtual museums and the potential of interactive exhibitions in VRChat guided by expert insight. Their approach takes place entirely in VR, independent of any physical museum premises.

While this thesis focuses on a museum context, the findings are by no means restricted to this domain. Consider, for instance, the VR game *Another Fisherman's Tale*[2], which incorporates innovative interactions with distant objects and out-of-body viewpoints. The gaming industry stands to benefit significantly from research into viewpoint manipulation, as these techniques can enrich user interaction and immersion in virtual environments. This overlap underscores the broad applicability of the insights gained through this work.

2. https://store.steampowered.com/app/2096570

1.4 Research Questions

The following main research question and two sub-research questions are addressed in this paper:

RQ1 What method is more effective in placing and manipulating viewpoints: an OCE based automatic and restricted approach or a free drone-inspired approach?

sRQ1 Which approach leads to greater cybersickness?

sRQ2 Which approach leads to better exploration of a virtual scene?

Besides these research questions comparing the two approaches, a third and fourth sub-research question regarding the Control-Schemes of the drone method are examined:

sRQ3 What drone Control-Scheme is preferred by users for an explorative task?

sRQ4 Can a correlation between Control-Schemes and SoE be observed?

2 Related Work

This paper delves into several key topics, including viewpoints, OCE, drones, distant interaction, embodiment and avatars. Each of these elements plays a significant role in the overall study and will be introduced through a review of related work in the following sections.

2.1 Viewpoints

Kim, Lee, and Lee (2022) examined optimal layouts for multiple viewpoint videos within a HUD in VR. However, their study focuses solely on passive viewing of VR content, without active user movement or interaction within the VR space. In contrast, this thesis allows users to actively control the position and orientation of their viewpoints. Importantly, this study does not aim to optimize the layout of the viewpoint video streams. Instead, it strikes a balance between layout restrictions and user preference, enabling users to freely align and position viewpoint videos. These can be placed either within a HUD or anchored in world-space.
Kusunoki et al. (2023) examined the benefit of multiple viewpoints whilst 3D drawing on objects in VR. They provided a bird's eye viewpoint and a viewpoint from the opposite side of the object to the user. These viewpoints where fixed and the user could change between them but could not create new viewpoints. The extra viewpoints led to mixed results, some users felt stressed by the viewpoints and others reported greater overview. In comparison, this thesis allows users to define their own viewpoints and manage them in order to allow each individual to specify the amount of views and not overwhelm them.
Prouzeau et al. (2019) examined the benefit of using visual links to show connections between data points in visualization and their context in the world in a VR setting. This thesis utilizes

visual links by connecting viewpoint videos to their respective viewpoint camera in the hopes of giving the user a greater overview and lessening confusion.
Shen, Mcgrenere, and Yoon (2024) explored the concept of viewpoints in relation to *perspective-taking* and the enhancement of empathy. Their research focused specifically on how perspective-taking and avatar embodiment can foster greater empathy in younger adults toward older adults. To investigate this, they conducted a user study involving pairs of young and old adults. Both participants were immersed in a VR environment, where the younger adult was assigned an avatar representing a younger version of the older adult. This avatar swap leverages the Proteus Effect (Praetorius and Görlich 2020), which describes how individuals' behavior can be influenced by the characteristics of their avatars. Within the study, the older adult shared a story from their youth, with the aim of fostering empathy and understanding through this interaction. The overarching goal of the research was to enhance intergenerational communication, with the expectation that such efforts would lead to increased mutual understanding and empathy between the participants. While Shen, Mcgrenere, and Yoon (2024) do not directly address the specific topic of multiple viewpoints as explored in this thesis, it is included here as a noteworthy example of how viewpoints can be applied within VR contexts. Beyond their utility in providing better spatial or situational awareness, viewpoints can also serve as a tool for *perspective-taking*. By allowing users to adopt the perspective of another character, such approaches can significantly enhance empathy toward that character and deepen the emotional connection between participants.

2.2 Object-Centric Exploration

Tatzgern et al. (2015) employed an OCE approach in AR to view real-world objects and locations in a city using an orbital camera on mobile devices. This study applies a similar automated and restricted OCE approach but compares it to a more freeform drone-based method in VR. Instead of an orbital camera, a HoverCam is utilized, which is based on UniCams (Zeleznik and Forsberg 1999), a system allowing camera control in 3D environments through 2D gestures. Although there are more advanced versions like ShellCam (Boubekeur 2014) and SHOCam (Ortega, Stuerzlinger, and Scheurich 2015), this study uses the basic HoverCam for its simplicity and sufficiency in representing an automated OCE approach. To enhance scene exploration, the 'look-and-fly' technique introduced by McCrae et al. (2009) is incorporated, enabling quick selection of objects to look at via ray casting. Another difference to Tatzgern et al. (2015) is that they only show a copy of the object that is being inspected. This thesis' viewpoints however show the real VR objects. Furthermore, this thesis utilizes OCE in VR with a Head-mounted Display (HMD) and not in AR with a mobile device.

Khan et al. (2005) define a HoverCam as a camera that maintains a fixed distance from an object while targeting a specific point on the object's surface. This target point can be moved with a two-dimensional input. A mouse was used in their study, while this thesis employs joystick input. As the target point shifts, the HoverCam follows, preserving its distance from the object and enabling detailed inspection of any surface point.

The original paper also introduced a field of influence around the object, where a traditional camera, upon entering this field, is drawn towards the object and behaves like a HoverCam. However, this influence field was not implemented in this thesis. Instead, the 'look-and-fly' feature (McCrae et al. 2009) enables users to select objects and have the camera immediately move to them. Additionally, Khan et al. (2005) proposed four distinct modes for defining the camera's Up-Vector, crucial for camera orientation and control. This thesis incorporates two of these modes and introduces three custom Up-Vector modes, detailed further in Section 4.6.4.

The selection of the 'look-and-fly' feature is done via ray casts. Jankowski and Hachet (2013) list ray casts as the most common way for object selection. Since it is the most well known technique it is also used in this thesis. An extension of this selection feature could also include gaze-based selection. Narkar et al. (2024) have tested gaze-based selection in combination with machine learning models to increase its effectiveness. This thesis does not focus on object selection in virtual environment and therefore only utilizes simple ray casting. However, the inclusion of additional gaze-based input could lead to better usability in future iterations.

2.3 Distant Interaction

Museums are often comprised of a diverse collection of exhibits and displays of differing shapes and sizes. These exhibit pieces, referred to as musealia, often require careful preservation and, as a result, cannot be directly handled or closely approached by visitors. This thesis aims to enable new methods for displaying these musealia by employing multiple viewpoints. For instance, consider a large-scale diorama display, which would ordinarily pose significant challenges for viewing due to its physical size and the potential occlusion caused by the objects in the diorama itself. By utilizing multiple viewpoints, it becomes possible to offer an in-depth exploration of such large dioramas, providing users with a more immersive experience than might be achieved with smaller, more limited dioramas.

With larger dioramas, the issue arises of enabling user interaction with objects that are physically located at a distance or otherwise inaccessible due to protective barriers. Therefore, an effective distant interaction method is essential to allow users to meaningfully engage with these viewpoints without compromising the integrity of the objects themselves. It is important to clarify, however, that this thesis primarily addresses the interaction with viewpoints and their dynamic manipulation rather than direct interaction with the musealia themselves. Essentially, users interact with virtual representations or 'virtual cameras' that offer remote perspectives of these objects, granting them the sensation of proximity and interaction without requiring physical access to the exhibits.
Pohl et al. (2021) introduces a system for creating spherical proxies, which serve a dual purpose by acting as both portals and viewpoints, allowing users to manipulate objects from a distance. These proxies enable spatial remapping and permit users to view and interact with distant objects within a virtual environment. While the poro system itself is not directly implemented in this thesis, it represents an innovative approach to remote interaction and is worth examining as a point of comparison. Specifically, the poro system exemplifies a unique integration of viewing and interaction within a single framework, an approach which is distinct from the one explored in this study.

In a museum setting, however, where direct interaction with objects is generally restricted, it becomes essential to limit the user's control to viewpoint adjustments alone. Although interaction features could potentially be disabled in the poro system to suit this type of environment, this thesis instead aims to explore and contrast a restricted method of viewpoint manipulation with a more unrestricted, flexible approach. The poro system allows a freely placeable spherical proxy, granting users freedom within a contained area to interact with the object it surrounds. As such, it remains challenging to strictly classify the poro system as either restrictive or freeform, given its combination of constrained spatial boundaries with substantial user flexibility.

2.4 Embodiment

Users can control a viewpoint where they can possibly see themselves controlling said viewpoint. This raises some interesting questions regarding embodiment. This is examined in more detail regarding Control-Schemes and embodiment. See **sRQ3** and **sRQ4**. In order to do so, a definition for *embodiment* is needed. multiple meanings for the term *embodiment* exist. Carrying on in this thesis, the term SoE is used. Kilteni, Groten, and Slater (2012) define SoE as: "SoE toward a body B is the sense that emerges when B's properties are processed as if they were the properties of one's own biological body." Furthermore Kilteni, Groten, and Slater (2012) describe SoE to consist of three sub-terms:

1. **Sense of Self-Location (SoSL)**: The sense of self-location refers to one's spatial experience of being inside a body and it does not refer to the spatial experience of being inside a world.

2. **Sense of Agency (SoA)**: The sense of agency refers to the sense of having global motor control, including the subjective experience of action, control intention, motor selection and the conscious experience of will.

3. **Sense of Body Ownership (SoBO)**: The sense of body ownership refers to one's self-attribution to a body.

Hoppe et al. (2022) investigated the effects of First Person Perspective (1stPP) and Third Person Perspective (3rdPP) in VR on user embodiment. Their study found evidence for a Perspective Continuum, where users in 3rdPP felt more like spectators, while those in 1stPP identified more as the main character. Participants experienced different perspectives and completed embodiment (Gorisse et al. 2017) and presence (Schubert, Friedmann, and Regenbrecht 2001) questionnaires after defending against attacking skeletons for three minutes. However, it is important to note that participants were not allowed to actively change their perspectives during the study. In this thesis, users are in full control of the viewpoint and can change their perspectives dynamically.
In comparison, Otono et al. (2023) explored the SoE in AR, where physical activity is directly linked to visual representation. Participants could flex their muscles to transition their avatars into more muscular versions in 3rdPP. While this study focused on avatar transformation rather

than perspective change, it provides insight into how physical interaction can influence embodiment. Similarly this thesis provides some insight on how Control-Schemes might influence embodiment.

Avatars in VR

Avatars are embodied virtual representations of users in VR (Dufresne et al. 2024). They can vary in degrees of realism, ranging from full-body representations to partial depictions, such as head, torso and arms. The realism of an avatar can negatively impact interaction when it does not align with user expectations (Herrera, Oh, and Bailenson 2018).
Additionally, avatars significantly influence SoBO, which in turn affects SoE (Kilteni, Groten, and Slater 2012). For this reason, an avatar with head and arm tracking, as well as inverse-kinematic walking animations, was implemented for this thesis. Users are able to view themselves from a third-person perspective, where a realistic avatar enhances the SoSL.
Dufresne et al. (2024) also argue that avatars suggest affordances, including possible ”false affordances.” Norman (2002) introduced the concept of affordances to the Human-Computer Interaction (HCI) community, defining them as: ”... the term affordance refers to the perceived and actual properties of the thing, primarily those fundamental properties that determine just how the thing could possibly be used.” Thus, an avatar with realistic hands suggests to the user that they can interact with objects using their hands. A ”false affordance” occurs when this is not the case—for example, if the hands are incapable of grabbing objects. This discrepancy would result in poor usability, creating an incongruence with user expectations (Latoschik and Wienrich 2022). Although the inverse-kinematic legs of the avatar in this thesis may suggest ”false affordances” - since users cannot, for instance, stomp with their feet - these features were nevertheless implemented to enhance the SoBO.

Fribourg et al. (2020) look into the influence of appearance, *control* and POV on the SoE of avatars in VR. They observed that *control* and POV are more important to users than the appearance of an avatar. With *control* they mean what the avatar is able to perform, which in turn contributes to the SoA, which has influence on the SoE (Kilteni, Groten, and Slater 2012). This however focuses on the *control* (SoA) of an avatar and not the *Control-Scheme* (SoBO). As an illustrative example of the potential impact of *Control-Schemes* on the SoBO, consider a VR game where players control a crab-like creature capable of grasping objects with its pincers. Typically, the Meta 2 controller employs the grip button located on the side of the device for object interaction (see Figure 1). However, an alternative *Control-Scheme* requiring the use of both the trigger and face buttons could more closely mimic a crab's natural pinching motion (see Figure 2). This control method, by more accurately reflecting the mechanics of a crab's pincers, may enhance the SoBO by aligning the user's physical actions with the virtual creature's movements.

Inoue and Kitazaki (2021) investigate the role of virtual mirrors in enhancing the SoE towards a VR avatar. Their paper elaborates on the psychological basis of virtual mirrors. When users view their avatar reflected in a virtual mirror, it leads to a stronger SoBO. The virtual mirror allows users to observe their avatar's movements in real time, reinforcing the synchronization between their own physical actions and the avatar's corresponding motions.
Although virtual mirrors are not employed in this thesis, the additional viewpoints have a similar

effect. By utilizing these viewpoints, users can observe their avatar from an external perspective, witnessing the avatar's movements align seamlessly with their own. This capability not only strengthens the connection between the user and their avatar but also enhances the overall immersive experience by providing visual confirmation of the avatar's responsiveness.

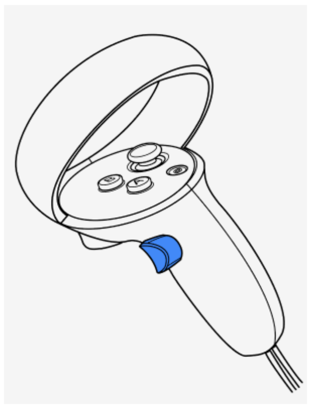

Figure 1: The Meta Quest 2 controller's grip button might be well suited to simulate the grab motion usual for human hand interaction but is a less fitting Control-Scheme for simulating a pinch motion for a crab-like avatar.

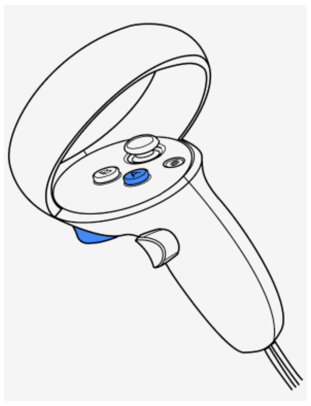

Figure 2: Using two buttons to represent the grab interaction of a pincer-like grab can lead to a greater SoE.

2.5 Drones

Erat et al. (2018) used drones in AR with an exocentric viewpoint to explore hidden areas, allowing users to select a destination for the drone to automatically fly to. However, this thesis opts for a more manual approach. Users are not able to just select a target point for the drone but need to steer it there themselves. Inoue et al. (2023) employed an automatic follower drone to provide a 3rdPP view of a manually controlled main drone, enhancing the user's understanding of the drone's height, heading and surroundings. Although their study only utilized

drone-centric controls—where the forward direction on the controller aligns with the drone's forward direction—this thesis considers the potential advantages of user-centric controls, where the controller's forward direction matches the user's orientation. Given the exploratory nature of this thesis and the close proximity of the user to the drone, user-centric coordinates may be more appropriate. Users are therefore allowed to choose their preferred Control-Scheme during the study.
Despite the prevalence of drone-centric controls in practical applications - likely due to the lack of information about the drone's position relative to the user, making user-centric controls impractical - this thesis allows participants to choose their preferred control mode. The findings will provide insights into whether user-centric or drone-centric controls are more effective in a VR setting. In addition, it is hypothesized that the preferred controls depend on the focus of the participants. If participants focus on the View-Panel whilst flying the drone, they might prefer drone-centric controls and user-centric controls when they are not. In alignment with Kilteni, Groten, and Slater (2012), when users want to embody the drone more they will also prefer the drone-centric Control-Scheme since it leads to better SoBO. Additionally, this might hint at, that the user's desired body ownership can quickly change by just focusing on the View-Panel and controlling a drone.
Ryu, Park, and Kim (2023) examine first-person view drone flights and investigate methods to mitigate the cybersickness often experienced by operators. Their approach involves using a deep neural network to integrate reverse optical flow on static landmarks into the drone-captured footage. This technique has demonstrated promising results in reducing cybersickness for first-person view drone operators. In contrast, this thesis adopts a different strategy and does not utilize a true first-person perspective for the drone camera. Instead, the drone's first-person view footage is displayed on panels situated within the virtual environment, or world-space, rather than occupying the user's entire FOV. This design choice serves two primary purposes: it simplifies the process of viewing multiple viewpoints simultaneously and helps reduce the cybersickness that can arise from displaying drone footage across the full FOV. By enabling the user to control the drone from a ground-level perspective, augmented with a smaller panel displaying the drone's first-person view, the visual motion mismatch between the user's stationary body and the drone's rapid movement is minimized. Reducing this sensory conflict directly addresses one of the primary causes of visually induced motion sickness, as outlined by Hettinger and Riccio (1992) and contributes to a more comfortable and manageable user experience.

3 System Design

For the user study scenario, participants are required to find and view three hidden objects within a life-size diorama. To accomplish this task, they must utilize the implemented system, which comprises two distinct methods for placing and manipulating viewpoints: an automated and restricted approach using the OCE HoverCam (Khan et al. 2005) and a free drone mode that provides complete control over the viewpoint's position and orientation.

Using these methods, a virtual camera can be positioned, with its viewpoint projected onto a Render-Texture displayed in a View-Panel. A *View-Camera* and a *View-Panel* together form a pair, referred to as a *View-Pair*.

As previously mentioned, the system facilitates the placement and manipulation of viewpoints to locate hidden objects. The following are the key requirements for the system, with the stipulation that users need to view up to three hidden objects simultaneously:

1. **Spawning a View-Pair**: To view objects from specific perspectives, the system must enable users to spawn a View-Pair. To prevent confusion when multiple viewpoints are active, a hard limit of five View-Pairs at a time is imposed. Whenever a View-Pair is spawned a View-Camera and its corresponding View-Panel are instantiated.

2. **Moving a View-Camera**: To locate the hidden objects, users must have the ability to move the View-Camera to various points in space.

3. **Deleting a View-Pair**: To encourage free exploration, users need the capability to delete viewpoints when necessary, in addition to creating and placing them. When a View-Pair is deleted its View-Camera and View-Panel are destroyed.

4. **Viewing the View-Camera**: The system must provide a means for users to view what a View-Camera sees. Both approaches share a common solution for this: a View-Panel. The View-Panel displays the Render-Texture of its associated View-Camera. Since this thesis does not focus on optimizing the layout for multiple viewpoint streams, the arrangement of these streams is left to the user, as discussed in Section 2.1. Users can move View-Panels freely, positioning them within the HUD or placing them in world space.

5. **Selecting and Unselecting a View-Camera**: Given that up to five View-Cameras are supported and only one should be manipulated at any given time, a feature for selecting and unselecting View-Cameras is essential. This thesis emphasizes viewpoint manipulation rather than selection, which is relatively straightforward and consistent across both methods. A View-Camera can be selected and unselected via a button in its corresponding View-Panel.

3.1 OCE Approach: HoverCam

The HoverCam (Khan et al. 2005) is defined as a camera that always looks at a specific object while maintaining a fixed distance from it. The camera can be moved along the surface of the object using a 2D input, but its distance from the object remains constant. This method was selected as an automated and restricted way to control viewpoints in an OCE setting. It is restricted because it centers on a single object (OCE) and is automated because, while the camera's position can be adjusted, its orientation is always directed toward the object.
A HoverCam can be spawned by selecting an object with a ray cast from the right-hand controller and pressing the 'A' button. This action spawns a new HoverCam, provided no other HoverCam is currently selected. If a HoverCam is already selected, pressing 'A' changes the object being inspected. A selected HoverCam is indicated by a green border around its model. The current object that is being inspected, is also highlighted with a blue border. The selected

HoverCam can be moved around its object using the right joystick. HoverCams can be deselected either by pressing the 'B' button or by using a button on its corresponding View-Panel. The HoverCam can be moved closer or further away with the 'X' and 'Y' buttons. See Figure 3 for a visual guide to the controls.

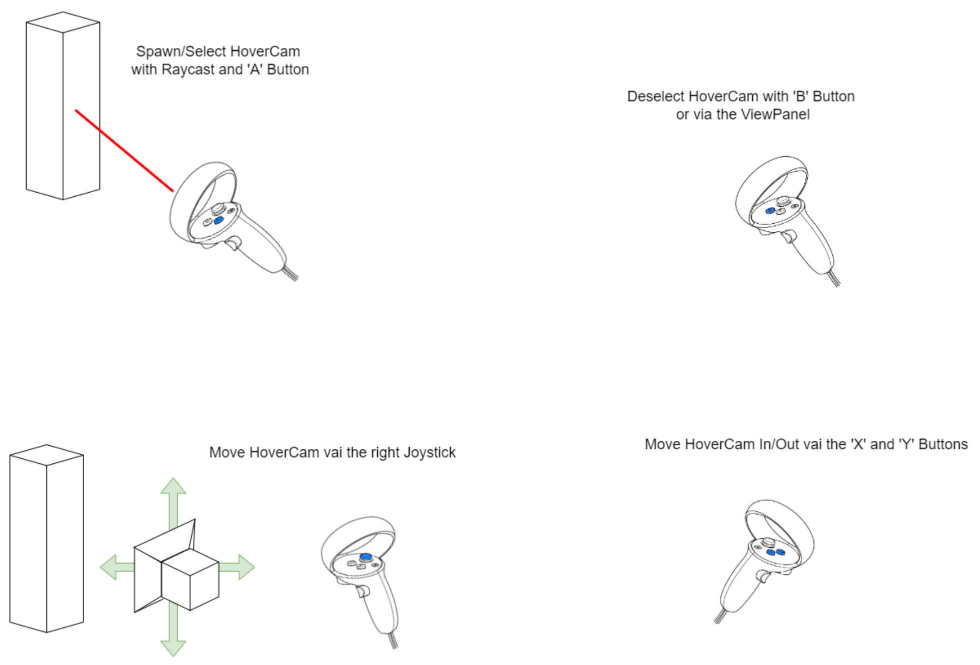
Spawn/Select HoverCam
with Raycast and 'A' Button
Deselect HoverCam with 'B' Button
or via the ViewPanel
Move HoverCam vai the right Joystick
Move HoverCam In/Out vai the 'X' and 'Y' Buttons

Figure 3: The OCE approach consists of a HoverCam that can be controlled with the joystick and buttons on both controllers. The HoverCam can be spawned and selected, selected and moved to a new object with a ray cast. The HoverCams position in relation to an object can be moved around the object and can also be put closer or further away from the object.

To summarize how the system fulfills the requirements: the HoverCam can be spawned via a ray cast and button press. It can be moved via a joystick whilst focused on an object and can be moved from object to object via a ray cast and a button press. The deletion, selection and viewing can be done in the View-Panel.

3.2 Free Drone Approach

In contrast, the free drones are not restricted in their orientation or position and can be placed wherever the user desires. The drones support two methods of movement: drone-centric and user-centric. The difference lies in how the drone's forward direction is determined. In drone-centric mode, forward is defined by the direction the drone faces, whereas in user-centric mode, forward is based on the direction the user faces.
The controls for the drone are the following: a drone can be spawned via the 'A' button. A selected drone can be maneuvered with the 'X' and 'Y' buttons to move up and down and the

right joystick to move forward/backward/left/right. A drone can be selected by grabbing the camera's model or via its View-Panel. Drones can be deselected with the 'B' button or the View-Panel. The selection status of a drone is also displayed via a green border around the cameras model. For a visual explanation of the controls see Figure 4.

To summarize how the system fulfills the requirements: The drone can be spawned with a

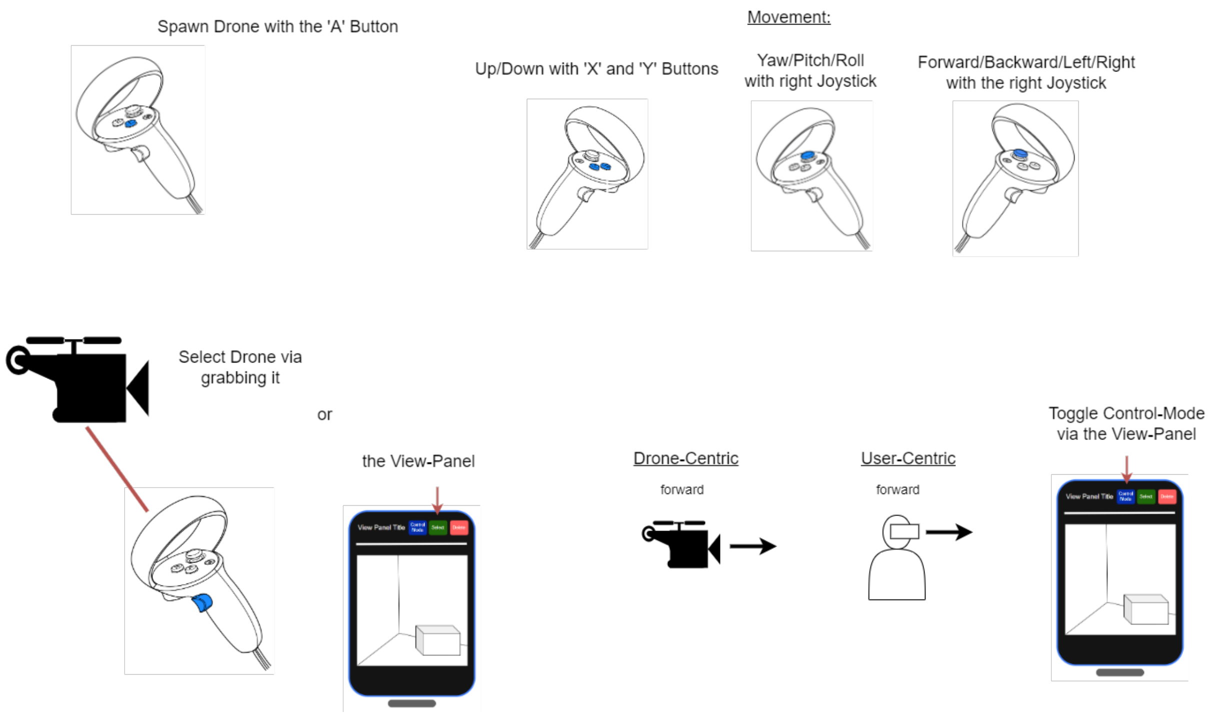
Spawn Drone with the 'A' Button
Movement:
Up/Down with 'X' and 'Y' Buttons
Yaw/Pitch/Roll with right Joystick
Forward/Backward/Left/Right with the right Joystick
Select Drone via grabbing it
or
the View-Panel
Drone-Centric
forward
User-Centric
forward
Toggle Control-Mode via the View-Panel

Figure 4: Drones can be controlled with both controllers. The drone can be spawned via a button press and can be selected via a ray cast. Movement of the drone is possible via the buttons and joysticks of the controllers. The drone supports two Control-Schemes that can be toggled between in the corresponding View-Panel of the drone.

button press and it can be moved with both joysticks and the face buttons of the left controller. Similar to the HoverCam the deletion, selection and viewing are handled by the View-Panel. In addition, the drone can also be selected by grabbing the drone itself.

3.3 View-Panels

Each camera is connected to a View-Panel that displays the camera's view. See Figure 5 for reference. View-Panels can be grabbed with either the controller by its title or the handle on the bottom and can be placed anywhere in world space. As already mentioned in the related work, this is done to allow users to customize the location of the viewpoint stream themselves. While the View-Panel does not move, it always rotates to face the user. When grabbing the View-Panel, the trigger button on the controller can be used to dock it into the HUD. Docked View-Panels have a gray background instead of a black one. Similarly, they can be undocked again. Every

View-Panel also includes a blue line curve that is linked to its corresponding camera for easier orientation. Prouzeau et al. (2019) has demonstrated the benefits of such visual links.

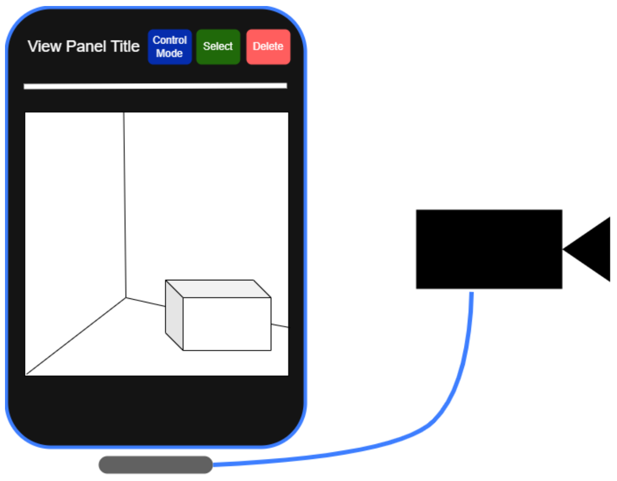

Figure 5: Every View-Camera is connected to a View-Panel. The connection is visualized with a blue line for easier orientation. The View-Panel displays the video feed of the View-Camera and has options to select and delete the View-Pair. For Drone-Cameras they also have the option to change the Control-Scheme. View-Panels can either be docked to the HUD or be in World-Space. They can be grabbed and moved freely.

Each View-Panel also features buttons for the deletion and selection of the view-pair. The blue 'Control Mode' button is only available for the free drone camera. With this button, the drone's control scheme can be toggled between a user-centric and a drone-centric version.

4 Implementation

The whole project was built in *Unity* (version 2022.3.9f1) and can be found at https://gitlab.mediacube.at/fhs44512/mmt-masterarbeit-david-maerzendorfer. For the implementation of VR, the *XR Interaction Toolkit* (version 2.5.0) was used. The XR Interaction Toolkit is provided by Unity and allows the creation of VR and AR experiences. It is hardware-independent and already provides useful systems for interactions and locomotion. The XR Interaction Toolkit can be easily extended if the provided components do not suffice for one's needs.

The study was developed for the *Meta Quest 2*. For the creation of the diorama synty [3] asset packages have been used.

In the following, the created systems for the study will be described and explained in more detail.

3. https://syntystore.com/collections/polygon-series

4.1 View-Pair

A *View-Pair* is a foundational component in this system, combining a *View-Camera* and a *View-Panel* to facilitate a linked display setup. Conceptually, the View-Pair acts as an intermediary that synchronizes camera output with the display panel, ensuring consistent and accurate visual rendering. This is implemented in code as an abstract base class called `BaseViewPair`, which establishes essential methods and properties to manage the lifecycle and interactions between the view components. For reference, see Listing 1.

```
public abstract class BaseViewPair: MonoBehaviour
{
    public ViewCamera viewCam;
    public BaseViewPanel basePanel;

    public UnityEvent onViewPairDeleted = new UnityEvent();

    public virtual void Awake()
    {
        // Setup render texture for the camera and set in panel
        viewCam.CreateRenderTexture();
        basePanel.SetRenderTexture(viewCam.renderTexture);
        basePanel.myViewPair = this;
    }

    public abstract void ReceiveSelect();

    public virtual void DeleteViewPair()
    {
        onViewPairDeleted.Invoke();
        Destroy(basePanel.gameObject);
        Destroy(viewCam.gameObject);
        Destroy(this.gameObject);
    }
}
```

Listing 1: A base View-Pair class, providing core structure for further specializations like the drone and HoverCam implementations.

The `ViewCamera` component consistently outputs its view to a `RenderTexture`, which is subsequently displayed by the View-Panel. The View-Pair class automates the initialization of this `RenderTexture` and assigns it to the View-Panel, establishing a self-contained view module where the camera's rendering is directly presented on the panel interface.

In this design, `BaseViewPair` serves as an abstract superclass, setting up key functionalities that are extended by specific implementations, such as the drone and HoverCam View-Pairs (detailed in Chapters 4.5 and 4.6). Through inheritance, these implementations can add unique functionalities while relying on the core framework provided by `BaseViewPair`.

Overall, the View-Pair structure facilitates the linkage between the View-Camera and View-Panel components, providing each component with a straightforward means to locate and interact with its paired counterpart. This approach not only enhances modularity but also ensures that changes in one component are reflected in the other, streamlining updates and interactions within the view system.

4.2 View-Camera

The *ViewCamera* script represents a modular viewpoint component within the system, serving as the foundational camera class. While both the *HoverCam* and *drone* implementations ultimately derive from this `ViewCamera`, they are distinguished primarily by their respective control mechanisms, which vary to suit their individual functional requirements.

```
public class ViewCamera : MonoBehaviour
{
    public Camera cam;

    public int textureWidth = 1024;
    public int textureHeight = 1024;

    [HideInInspector]
    public RenderTexture renderTexture;


    public void CreateRenderTexture()
    {
        renderTexture = new RenderTexture(textureWidth, textureHeight,
           24);
        renderTexture.enableRandomWrite = true;
        renderTexture.Create();

        cam.targetTexture = renderTexture;
    }

    private void OnDestroy()
    {
        if (renderTexture != null)
        {
            renderTexture.Release();
            renderTexture = null;
        }
    }
}
```

Listing 2: The ViewCamera class shared by both HoverCam and drone implementations, differing only in control mechanisms.

As shown in Listing 2, the View-Camera class extends the standard Unity `Camera` functionality by rendering its output directly into a `RenderTexture`. This `RenderTexture` allows the View-Camera's viewpoint to be displayed on a texture, which can be linked to various display elements. The resolution for this study has been set to 1024x1024 pixels, balancing visual clarity with performance efficiency.

The `CreateRenderTexture` method within `ViewCamera` configures a new `RenderTexture` with the specified dimensions, ensuring that the camera's output is consistently captured. This `RenderTexture` is initialized by the View-Pair, which manages the setup process for both the camera and the panel display. When the `ViewCamera` instance is destroyed, it ensures resource efficiency by releasing its `RenderTexture`, thus freeing memory and preventing potential memory leaks.

In this architecture, the `ViewCamera` acts as a flexible, reusable camera component adaptable for different control schemas while maintaining a standardized texture output. The encapsulation of rendering logic within `ViewCamera` promotes modularity, allowing distinct implementations to leverage this shared camera infrastructure with minimal redundancy.

4.3 View-Panel

The *View-Panel* framework is comprised of two main classes: `BaseViewPanel` and `DroneViewPanel`. The `DroneViewPanel` class extends `BaseViewPanel` by introducing an additional control mode button specifically designed for the drone, enhancing interactivity and allowing users to toggle the drone's control mode directly from the panel interface.

The `BaseViewPanel` contains the core interaction logic and is designed to be highly modular. A critical requirement is that `BaseViewPanel` must be attached to a GameObject that also includes the `XRGrabInteractable` component, part of the XR Interaction Toolkit provided by Unity. This toolkit facilitates interaction within XR environments, enabling users to grab and manipulate objects in a 3D space. For detailed guidance on the XR Interaction Toolkit, please refer to its official documentation.

The `XRGrabInteractable` component plays a central role in managing the docking functionality of the View-Panel within the HUD. By listening to events provided by `XRGrabInteractable`, `BaseViewPanel` is able to detect when it is docked to or removed from the HUD. Although `BaseViewPanel` registers its state in relation to the HUD, the actual docking mechanics, such as positioning and alignment, are managed externally by the `ViewManager`. This separation of responsibilities allows for a clean, modular approach where the `ViewManager` can centralize positioning logic across multiple View-Panels.

Interaction within the View-Panel is further facilitated by standard User Interface (UI) buttons. Each button is connected to methods within `BaseViewPanel`, which act as intermediaries to execute specific actions, such as selecting options or modes. These methods delegate functional tasks to the relevant View-Pair, allowing the View-Panel to communicate seamlessly with its linked components.

The design of `BaseViewPanel` as a customizable interface allows for diverse interaction capabilities tailored to specific components, like the drone, while retaining a consistent base functionality. This modularity enables the system to expand flexibly, allowing future integrations of additional control elements or interactive features.

4.4 View-Manager

The `ViewManager` serves as the core class responsible for orchestrating the instantiation of `ViewCamera` instances and managing the docking of `BaseViewPanel` objects within the HUD. This central component ensures seamless interaction between the view elements, allowing for a streamlined experience in setting up and interacting with camera panels.

```
public class SingletonMonoBehaviour<T> : MonoBehaviour where T :
    MonoBehaviour
{
    public static T Instance { get; private set; }

    public virtual void Awake()
    {
        if (Instance != null && Instance != this)
        {
            Destroy(this.gameObject);
            return;
        }
        else
        {
            Instance = this.GetComponent<T>();
        }
    }
}
```

Listing 3: Implementation of a simple Singleton pattern for MonoBehaviours, allowing scripts to be easily accessible from any part of the application.

The `ViewManager` class is implemented as a `SingletonMonoBehaviour`, as shown in Listing 3. By applying the Singleton pattern, `ViewManager` becomes globally accessible, allowing other classes to interact with it without needing to pass references. Although the Singleton pattern is sometimes labeled as an ”anti-pattern” due to its potential to create hidden dependencies and reduce flexibility, in this case, it was chosen to simplify access to such a central script, where only one instance should manage the View-Camera and View-Panel interactions.

The Singleton pattern is implemented through the `SingletonMonoBehaviour` generic class, which enforces a single instance of the View-Manager while preventing duplication. During initialization, `SingletonMonoBehaviour` checks for existing instances and destroys any duplicates, ensuring only one active instance persists. This pattern facilitates the View-Manager's ability to perform critical tasks across multiple scenes or components without the need for redundant instantiation or complex referencing structures.

Thus, the `ViewManager` efficiently centralizes control over view creation and docking processes, enabling a cohesive and manageable structure for view handling across the application.

4.4.1 HUD Management

Each `BaseViewPanel` notifies the `ViewManager` when it requests to be transferred to the HUD. The `ViewManager` then carries out the necessary actions to achieve this transition. In this implementation, the HUD functions as a simulated or ”fake” HUD, where View-Panels are not moved to screen-space but instead remain in world-space, attached to the user’s head movement. This approach provides an immersive, semi-static interface without the need for complex screen-space transformations.

One limitation of this approach, however, is that keeping the View-Panels in world-space introduces minor positional discrepancies when following head movement. Nonetheless, as HUD interaction is not a central focus of this study, a simplified solution was adopted for efficiency.

To incorporate a `BaseViewPanel` into the HUD, the `ViewManager` reassigns the panel’s parent to the `HMD` GameObject, which tracks the user’s head movements. This parenting method allows the `BaseViewPanel` to maintain a relative position within the user’s field of view, simulating an attached HUD experience. Additionally, the `ViewManager` adjusts the background color of the View-Panel to provide a visual indicator of its docked status, making it more distinguishable from non-docked panels.

This approach to HUD management is a practical compromise, providing a functional solution to HUD emulation in world-space while maintaining focus on the study’s primary objectives.

4.4.2 View Spawning via BaseViewModeHandlers

The `ViewManager` employs a `ViewMode` enum to indicate the active mode, either *HoverCam* or *drone* mode. For each mode, a designated `ViewModeHandler` is maintained to handle the instantiation of View-Pairs. By delegating View-Pair spawning to specific handlers, the code within `ViewManager` is modularized, enhancing readability and simplifying future expansions with additional View-Modes.

At the core of this architecture is the `BaseViewModeHandler` abstract class, which establishes a blueprint for all `ViewModeHandlers`. This design facilitates standardized implementations of new view modes, allowing for flexible extension without extensive refactoring of the View-Manager. Listing 4 illustrates the structure of `BaseViewModeHandler`.

The `BaseViewModeHandler` provides core functionalities for managing View-Pairs. It includes methods for deleting all active View-Pairs and tracking the number of currently active views, as well as abstract methods `SpawnViewPair` and `Activate` which must be implemented by derived classes. The `Deactivate` method is partially implemented in `BaseViewModeHandler` to allow deletion of all View-Pairs; however, it can be overridden in specific handlers if additional deactivation logic is required.

Each `ViewModeHandler` maintains a list of `ViewConfigs` that dictate the View-Pairs available for spawning. This configuration also sets the maximum number of View-Pairs per mode, which, for this study, is capped at five per mode. This limitation ensures controlled spawning within the scope of the user study and prevents resource overload.

Additionally, `ViewManager` provides various events, such as `onAnyCamSpawned` and `onAnyCamDestroyed`, which support tracking and logging during the user study. These events help monitor user interactions with View-Pairs and facilitate data collection for analysis.

This modular, event-driven approach enhances the scalability of the system, allowing for efficient management of View-Modes while maintaining a clear, maintainable code structure in the `ViewManager`.

4.5 Drone Implementation

The drone implementation includes a dedicated version of the `BaseViewPair` called `DroneViewPair`, shown in Listing 5. This class extends the functionality of the `BaseViewPair` by incorporating the `DroneCamController`, which encapsulates the logic required to control the movement of the drone.

In `DroneViewPair`, the `Awake` method is overridden to initialize the `DroneCamController` and associate it with the View-Pair. This setup enables seamless integration of the camera controller within the View-Pair's lifecycle, allowing it to inherit the base functionality of `BaseViewPair` while adding specific control logic for the drone.

The `ReceiveSelect` method is also overridden to manage the selection state of the drone. When called, this method toggles the `IsSelected` property of `DroneCamController`, activating or deactivating the drone's control based on user input. This selection mechanism is critical for interaction management, allowing the drone to be easily targeted and controlled by users as part of the application's interface.

This design effectively modularizes the drone control logic, encapsulating it within the `DroneViewPair` and allowing the `ViewManager` to handle drone-specific functionality independently from other view types. This separation improves the scalability of the system, as additional view types can be implemented without requiring modifications to the core view-management framework.

4.5.1 Drone Controller

The `DroneCamController` script encapsulates the functionality required to control the drone camera's movement and orientation. This controller provides configurable movement speeds and utilizes Unity's Input System package for handling user inputs. Two key components within the controller are the movement speed properties (Listing 6) and the `DroneActions` class, which configures the `InputActions` necessary for controlling the drone (Listing 7).

The `DroneCamController` has a property, `IsSelected`, which manages the drone's selection state. When a drone is selected, a visual outline is applied to its model to indicate activation. This effect is achieved using an outline package that combines all meshes and applies a shader-based outline. In addition, while the user is controlling the drone, they are prevented from moving, as input is redirected to drone control. To handle this, the XR Interaction Toolkit's `LocomotionSystem` component is utilized. By toggling the active state of the `LocomotionSystem` GameObject, user movement is effectively disabled or re-enabled as needed. The code for locating and enabling/disabling locomotion is shown in Listing 8.

The drone's movement is controlled by hooking `InputActions` to methods that record input values and store them in variables. These values are then applied in the `Update` method to adjust the drone's position and rotation. The `Update` method also differentiates between movement modes, applying input values accordingly based on the selected mode. This design allows the drone to seamlessly switch between movement styles, ensuring responsive and adaptable control that can be fine-tuned for user experience requirements in this study.

4.5.2 Drone View Mode-Handler

The `DroneViewModeHandler` is responsible for managing the spawning and lifecycle of View-Cameras in drone mode. This handler is used by the `ViewManager` and inherits from `BaseViewModeHandler` (Listing 4). It specifically implements the methods required for spawning, activating and deactivating drone camera views, thereby providing a tailored behavior for drone operations.

In addition to view management, `DroneViewModeHandler` handles essential Input-Actions related to drone control. These include actions for spawning a new drone camera and for unselecting any currently active cameras. This input-based setup allows for responsive and dynamic interactions within the drone view mode, facilitating real-time control adjustments based on user inputs.

The `DroneViewModeHandler` is configured to spawn each new drone camera at a predefined distance from the user's left controller, ensuring that the drone camera appears in a spatially intuitive location relative to the user's position. This spatial setup aids in user orientation and simplifies the process of locating the drone camera in the virtual environment.

This modular approach, where specific input actions and spawning parameters are encapsulated within the `DroneViewModeHandler`, enhances code readability and extensibility, allowing for seamless integration of additional features or configurations tailored to drone mode.

4.6 HoverCam Implementation

The HoverCam implementation is inspired by the foundational principles outlined by Khan et al. (2005). However, the algorithm was adapted and fine-tuned to suit the specific requirements of this study.

Before delving into the detailed workings of the HoverCam, it is useful to explore its precursor, the *Orbit-Camera*, as it provides context for the design evolution of the HoverCam.

4.6.1 Excursion: Orbit-Camera

The Orbit-Camera was an early prototype for implementing an automatic, restricted OCE approach. This system allowed users to select certain objects within the scene to act as points of focus, referred to as objects-in-the-center. Upon selecting an object, a View-Camera would be spawned to enable the user to orbit around it. The View-Camera would always orient itself toward a predefined point on the object, thus maintaining the object's central position within the user's field of view. A conceptual visualization of the Orbit-Camera's behavior is presented in Figure 6.

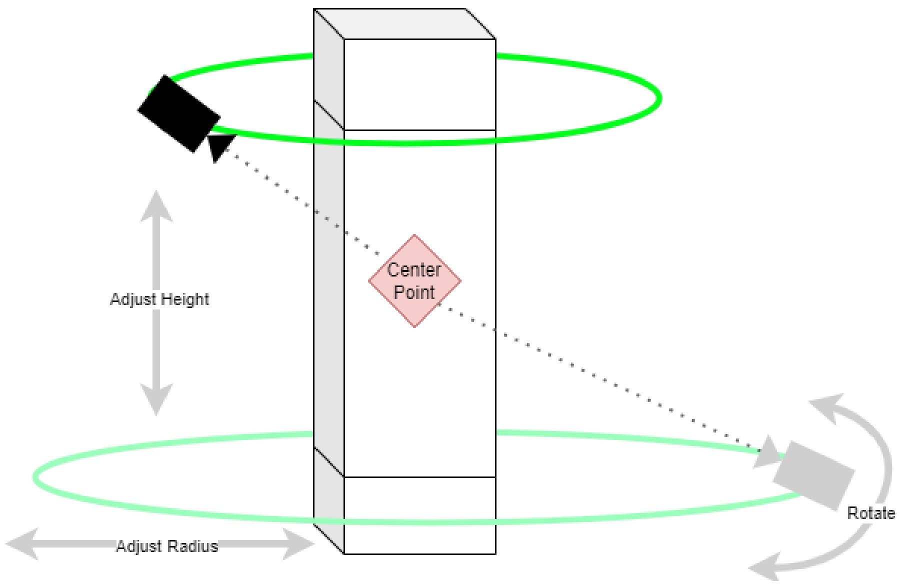

Figure 6: The Orbit-Camera enabled the selection of predefined objects. These objects had a fixed look-at point and the camera would orbit around them, always directed towards this fixed point. The user could adjust the orbit of the camera, but the look-at point remained static.

The Orbit-Camera implementation leveraged Unity's Cinemachine package, which provides Virtual Cameras capable of orbiting around an object. This package facilitated the core functionality of the Orbit-Camera. However, a limitation arose from the necessity of assigning each Virtual Camera to a unique Unity Layer, which ensured that their views did not overlap. This, in turn, imposed a constraint on the number of possible cameras in the system.

Despite the technical merits, the Orbit-Camera was ultimately deemed unsuitable for the user study comparison with the drone camera system. The key limitation of the Orbit-Camera was its rigid functionality: it could only focus on a select group of pre-configured objects and always maintained a fixed look-at point on those objects. This rigidity would likely have introduced bias into the comparison, as it did not allow for flexible exploration of the environment. Consequently, the Orbit-Camera was replaced with the more versatile HoverCam, which offers greater flexibility and control for the user.

4.6.2 HoverCam View Mode-Handler

The `HoverCamViewModeHandler` functions in a manner analogous to the `DroneView-ModeHandler`, extending the `BaseViewModeHandler` to manage the spawning, activation and deactivation of HoverCam instances. It is also responsible for handling the InputAction associated with unselecting all active cameras.

However, unlike the `DroneViewModeHandler`, the `HoverCamViewModeHandler` does not manage the Input-Action for spawning a HoverCam. This decision was made due to the distinct nature of the HoverCam's control system, which incorporates a specialized 'look-and-fly' functionality (McCrae et al. 2009). To ensure modularity and maintain a clean separation of concerns, the management of this feature is delegated to a dedicated component. This modular design approach enhances maintainability and scalability, allowing for easier adjustments and additions to the HoverCam functionality in future iterations of the system.

In summary, while the `HoverCamViewModeHandler` retains responsibilities similar to its drone counterpart, the separation of concerns allows each component to focus on its core functionality, improving both performance and flexibility within the overall system.

4.6.3 HoverCam Spawner

The 'look-and-fly' mechanic facilitates user interaction by allowing them to point a ray cast at a specific object, press a button and either spawn a new HoverCam or reposition an existing one to that location.

This functionality is implemented in the `HoverCamSpawner` script, which leverages the XR Interaction Toolkit's `XRRayInteractor`, typically assigned to the right controller by default. The script queries the `XRRayInteractor`'s current hit point and checks whether the hit object belongs to a specific layer mask. For example, one such layer mask is the *Default* layer, which ensures that users cannot interact with objects like the floor or other HoverCams, as these are assigned to the *Floor* and *CamModel* layers, respectively.

Upon detecting a valid layer, the `HoverViewModeHandler` is invoked, which either returns a newly spawned HoverCam instance or the currently selected HoverCam. Once a valid target is identified, the `HoverCamController` is responsible for ensuring that the camera smoothly transitions and orients itself toward the newly selected location. This approach provides a seamless user experience by combining precise target selection with smooth camera movements.

By utilizing the XR Interaction Toolkit's built-in ray casting and layer masking features, the system ensures that user interactions are both efficient and intuitive while maintaining a clear separation between different types of objects in the scene.

4.6.4 HoverCam Controller

The `HoverCamController` manages the movement of the HoverCam according to the algorithm described by Khan et al. (2005). The `HoverCamController` exposes fields for defining the used Input-Actions for controlling the HoverCam as well as the movement speed and zoom speed of the HoverCam.

The HoverCam is always focused on a look-at-target that can be set in this controller and it can only be moved if it is currently selected. For the algorithm and how the movement logic works see the original paper by Khan et al. (2005).

During the pilot test for the study, a usability problem was found. The HoverCam allows the movement of a camera around an object with just a simple 2D vector as input. The original HoverCam was designed for the usage with a mouse however, this means the camera was pushed or pulled into a direction. In this study, the camera is moved with a joystick. The rest of the algorithm does not change, however. The camera always stays a certain distance away from the object. The original paper suggested four different modes for the ”Up” direction of the HoverCam. The Up-Direction is important since it is needed to orient the camera. The four suggested modes are:

1. **Global**: the up-direction is the up-direction of the world
2. **Local**: the up-direction is the local up-direction of the camera
3. **Driving**: ”For some objects, the user may wish to have the feeling of moving the input device left or right should turn the object so that moving (the device) up is always ’forward’”(Khan et al. 2005).
4. **Custom**: some objects might require custom up-directions. Khan et al. (2005) give the example of a car: when viewing the car from the side, up should be towards the roof but when viewing the car from the top, up should be towards the hood of the car.

Prior to the pilot test, only the Global Up-Vector Model was implemented. This was later improved since this mode brings with it a problem when viewing an object from the top. When a user tries to view an object straight from the top, they can get stuck in this position. See Figure 7 with an example of the original paper.

In comparison, the Local Up-Vector Model does not have a problem with the ’north-pole’ as well as the ’south-pole’ of an object. It can however lead to a camera that is upside-down. See Figure 8 with an example of the original paper.

To circumvent these problems five different modes have been implemented for this study. The modes are the following:

1. **Global**: For the up direction the world’s up is used. This leads to problems when the camera approaches the ’north pole’ as well as the ’south pole’ of an object.
2. **Restricted Global**: This mode restricts the top and bottom-most area of an object as not accessible. Therefore the camera cannot get stuck. This restricted area is defined by the minimum angle between the up-vector of the object and the forward-vector of the camera. See Figure 9 for further explanation.
3. **Local**: The up direction is the camera’s local up-vector. With this mode, there is no problem with the ’north/south pole’. However, this can lead to an upside-down camera.

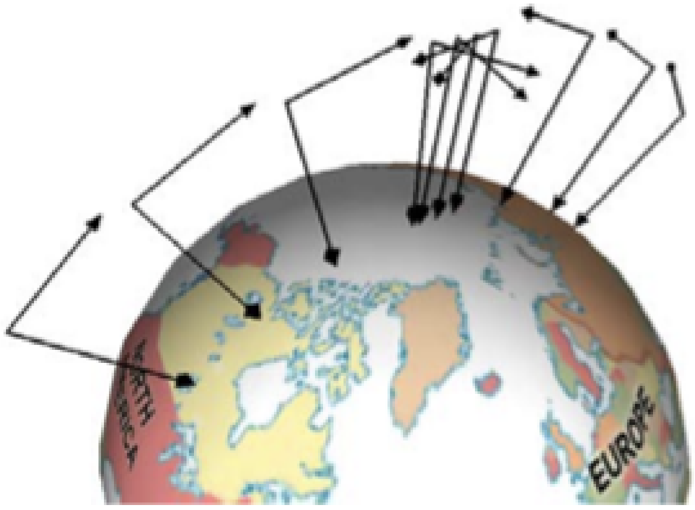

Figure 7: The Global Up-Vector Model of the Hover-Camera causes issues if the camera approaches the 'north-pole' of the object. The Camera tends to get stuck.

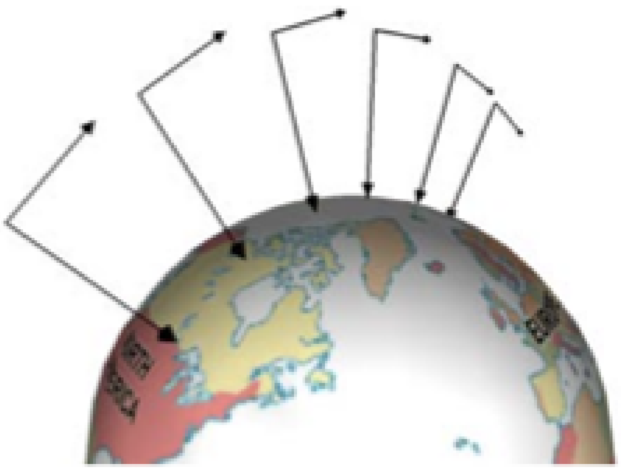

Figure 8: The Local Up-Vector Model of the Hover-Camera does not have a problem with the 'north-pole' as well as the 'south-pole' of an object, it can however lead to an upside-down camera.

4. **Rectifying Local**: This mode acts similar to the Local mode, however whenever there is no input the camera rectifies itself to align with the global up-direction. This means any upside-down cameras turn right-side-up again.

5. **Selective Local**: This mode is similar to the Restricted Global mode, however instead of not allowing movement in the restricted area whenever the area is entered, the camera behaves as if in Rectifying Local mode. Users can get over the 'north-pole' and once they stop their input they get rectified to Global mode again.

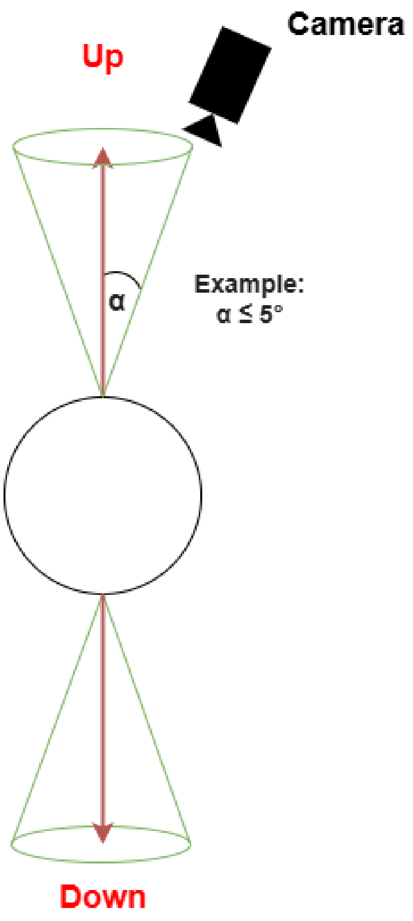

Figure 9: The restriction area is defined by the minimum angle between the up/down vector of the object and the camera. The camera cannot enter the restricted area.

HoverCam Mode Pilot Test

A small pilot test was conducted to decide on the best mode for the study. Before the study, the *Selective Local Mode* was hypothesized the best mode since it combines the benefits of all modes. The subject group (n=5, average age = 25.4) was asked to test each mode. They had a few minutes with each mode and were asked to try them out in the same environment that was later used for the study. Participants ranked the modes from 1 (best) to 5 (worst). The result can be seen in the following Table 1.

Contrary to the expected result the **Restricted Global Mode** was deemed the best mode for the study. The disorientation of the rectifying effect was too disrupting, a more clean and restricted mode leads to better usability. **Restricted Global Mode** was therefore used for the study. During the test, the participants have given some feedback. Many participants disliked the instant jump of the rectifying modes, this jump led to unwanted disorientation and had a negative effect on the flow of camera control. Based on this feedback, implementing a smoother transition between orientations could help alleviate disorientation. Whilst testing, it was also discovered that in the local mode the camera could end up tilted even if just moving left/right

HoverCam Mode Rankings					
Participant	Global	Restricted Global	Local	Rectifying Local	Selective Local
1	3	5	1	4	2
2	5	1	4	3	2
3	4	1	5	2	3
4	4	1	5	3	2
5	5	1	4	3	2
Avg Rank	**4.2**	**1.8**	**3.8**	**3**	**2.2**

Table 1: The results of the HoverCam Mode Pilot Test indicate that the Restricted Global Mode was ranked highest overall.

and slightly up/down. Additionally, some participants were unhappy with the speed of the *movement vignette*, it was showing up too fast.

Movement Vignette

The *XR Interaction Toolkit* provides a *Tunneling Vignette Controller*[4]. Whenever a user moves with the left joystick a vignette effect appears to restrict the users' FOV. See figure 10 for an example of the effect. The aim of this restriction is to lessen cybersickness during movement (Wu and Suma Rosenberg 2022).

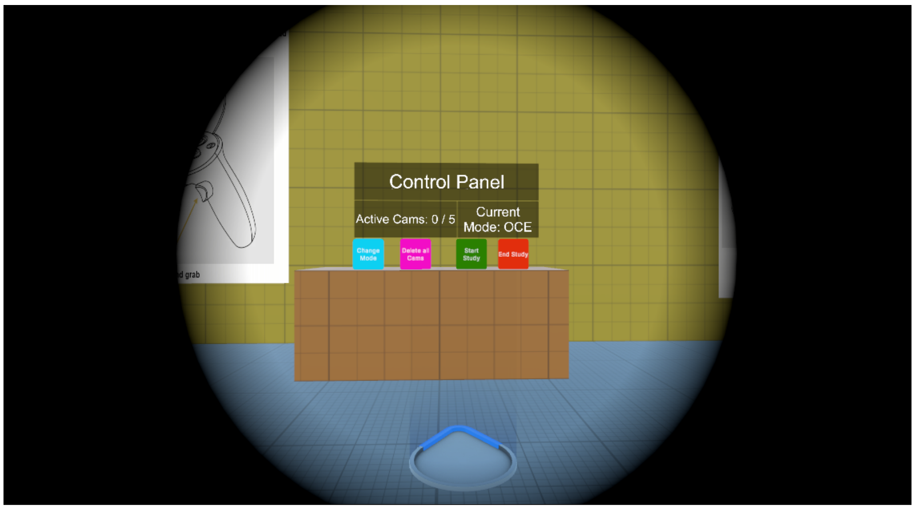

Figure 10: A vignette is displayed during the movement with the left joystick. This is done to mitigate cybersickness.

Some participants during the pilot test for the *HoverCam Mode* mentioned that the speed at which the vignette appears and disappears is too fast and abrupt. The speed was not adjusted however due to the reason that the user study was performed soon after the *HoverCam Mode*

4. https://docs.unity3d.com/Packages/com.unity.xr.interaction.toolkit@2.1/manual/tunneling-vignette-controller.html

test. Due to time constraints, these settings were not adjusted before the study. In addition, the user study is not about the movement of the users themselves but the movement of the viewpoints. During the user study, participants did not need to move much. Furthermore, most participants preferred using the teleport instead of the joystick movement anyway.

4.7 Avatar

The avatar used in this study was developed through a series of incremental solutions and tutorials, but it does not represent the most advanced or optimized implementation in terms of functionality and system integration. Initially, the project relied on the XR Interaction Toolkit, which did not provide a built-in avatar system. As a result, a temporary avatar was implemented early on to allow progress in the core aspects of the study. This avatar served as a placeholder while other critical components were being developed. However, after addressing the more pressing requirements of the study, the avatar system was revisited and enhanced.

At this stage of development, the Meta Avatar SDK was explored as a potential solution. The Meta Avatar SDK provided a relatively simple integration process and performed well in terms of basic functionality. However, this implementation was limited in scope. The SDK avatar consisted only of a floating torso without any Inverse Kinematics (IK) for the legs, making it unsuitable for full-body interaction. Moreover, integrating the Meta Avatar SDK caused significant incompatibilities with other aspects of the project, particularly in disrupting the interaction logic provided by the XR Interaction Toolkit. Given the extent of these issues, it was determined that the Meta Avatar SDK would require a complete overhaul of the project's existing interaction systems, making it an impractical solution for the study.

Consequently, the initial temporary avatar solution was revisited and refined to create a more complete and functional avatar. The final avatar used in this study was based on a rigged human model provided by Mixamo[5]. This model was imported into Unity, where it was enhanced using the *Animation Rigging* package. The humanoid rig was augmented with IK target handles for key body parts, specifically the head, hands and feet. These additions allowed the avatar to be more dynamic and responsive to user input, making the interaction more immersive.

To achieve proper control of the avatar's movements, the head and hands were linked to the movements of the user's HMD and controllers. This allowed for the avatar's head and hands to follow the user's motions in real-time, creating a more believable and interactive experience. To implement this, a tutorial[6] was followed, which included detailed steps for controlling the IK targets of the head and hands, as well as animating hand gestures. These gestures included actions such as grabbing, which were essential for certain tasks within the study.

A separate script was written specifically to manage the IK targets for the feet. Foot placement was an important aspect of the avatar's interaction with the environment and a tutorial by Unity[7]

5. https://www.mixamo.com/#/?page=1&query=passive+marker&type=Character
6. https://www.youtube.com/watch?v=v47lmqfrQ9s
7. https://www.youtube.com/watch?v=acMK93A-FSY

was used to implement realistic foot movement, ensuring that the avatar's feet properly followed the terrain or floor during movement.

Despite these improvements, several limitations persisted in the avatar's functionality. One significant limitation was that the avatar could not be viewed from a first-person perspective (1stPP). The avatar was only visible from external viewpoints, meaning the user could not see their own avatar while interacting in the virtual environment. This limitation was a trade-off in the interest of development time and resource constraints. Nonetheless, the avatar system was functional for the scope of the study and future iterations could further refine the avatar to include full first-person visibility or more advanced IK solutions.

In summary, the avatar used in this study was a refined solution developed with the help of various online resources. While it is not a state-of-the-art avatar system, it provided sufficient functionality for the needs of the study. The final implementation offered basic full-body interaction with the user's HMD and controllers, but there remain areas for improvement, particularly with regard to first-person visibility and further IK integration.

4.8 Point of Interest

The `PointOfInterest` script plays a critical role in the user study, as it governs the detection of POIs within the camera's field of view and triggers events when they enter or exit the view. This functionality is integral to determining whether a user has successfully located a POI, which is a central task in the study. The `PointOfInterest` script is designed to be flexible, allowing for various detection mechanisms based on the camera's position and visibility criteria.

For the `PointOfInterest` script to function correctly, the object that contains the script must also have a `Renderer` component attached. The presence of this component is essential for determining whether the object is visible to the camera. The POI system is configurable, allowing for a minimum required distance to be set for detection. If the camera is too far from the object, it will not be considered within the camera's view and no detection event will be triggered. This distance threshold ensures that the POI is only detected when it is within a reasonable range for the user to interact with or observe.

To provide visual feedback during the detection process, an outline effect is applied to the POI when it enters the camera's view. This outline highlights the object, providing clear indicators to the user that the POI has been located. The outline effect is an essential feature for improving user awareness of their progress in the task, as it directly reflects the visibility status of the POI.

Initially, Unity's built-in `isVisible` flag, part of the renderer component, was considered the primary method for detecting visibility. The `isVisible` flag returns a Boolean value that indicates whether any part of the object is within the camera's view. This flag is used by Unity to call a MonoBehaviours `OnBecameVisible` and `OnBecameInvisible` functions. However, this approach was ultimately discarded due to its limitations. The most significant issue with `isVisible` is that it returns true even if only the object's shadow is visible to the camera, which could result in false positives where an object is considered visible when it is not actually observable by the user.

As a more accurate alternative, ray casting was implemented to determine visibility. The ray casting mechanism is triggered during each physics update cycle (*FixedUpdate*) to check whether the POI is within the frustum (the camera's viewable area) of any active camera in the scene. If the object is within the camera's frustum, it is considered visible, provided it is not occluded by other objects. To detect occlusion, a single ray is cast from the center of the POI toward the camera. If this ray hits an object before reaching the camera, the POI is considered occluded and not visible.

Although this ray casting method works effectively in many cases, it has some limitations. For example, since only one ray is cast from the center of the object, partial occlusion (where part of the object is blocked while other parts remain visible) could result in the entire object being marked as invisible, even if portions of it are still clearly in view. Despite this limitation, the approach was deemed sufficient for the purposes of this study, as the primary goal was to detect objects that are fully or largely visible in the camera's view.

Overall, the `PointOfInterest` script provides an effective and efficient solution for detecting visibility and triggering events based on the camera's perspective. The use of ray casting ensures more reliable detection than the `isVisible` flag and the inclusion of a configurable distance threshold and visual feedback (via outlines) enhances the user experience by making the interaction more intuitive and responsive.

4.9 Control-Manager

The `ControlManager` script governs the logic behind the *Control-Table*, which serves as an interface for managing various aspects of the user study. The *Control-Table*, as illustrated in Figure 11, provides users with an intuitive way to interact with the View-Camera system. It facilitates several key functionalities, such as switching the current View-Camera mode, removing all active cameras and starting or prematurely terminating the user study.

One of the primary roles of the `ControlManager` is to manage the dynamic interaction with the View-Camera system. The user can easily change the active View-Camera mode using buttons provided on the *Control-Table*. This feature is crucial as the user study involves testing different camera modes and the *Control-Table* allows for quick transitions between these modes during the study.

Another important functionality provided by the `ControlManager` is the ability to remove all active View-Cameras. This can be particularly useful in resetting the system between study trials or after the user study has been completed. The *Control-Manager* ensures that this process is smooth and does not leave any residual cameras that could interfere with further tests or analysis.

In addition to its management of the View-Camera system, the *Control-Table* also provides real-time feedback on the current status of the View-Camera modes. It displays which View-Camera mode is currently active, as well as the number of active View-Cameras. This real-time information is essential for the study, as it helps both the participants and the researchers track the system's state during the experiment.

The *Control-Table* is also responsible for managing the flow of the user study. It includes buttons for starting and stopping the study, which provides researchers with control over the timing of the experiment. If the study needs to be prematurely terminated, the `ControlManager` ensures that all active cameras are cleared and the system is returned to its initial state.

The design of the *Control-Table* provides an effective and user-friendly interface for controlling the View-Camera modes and managing the user study, contributing to the overall success of the experimental setup.

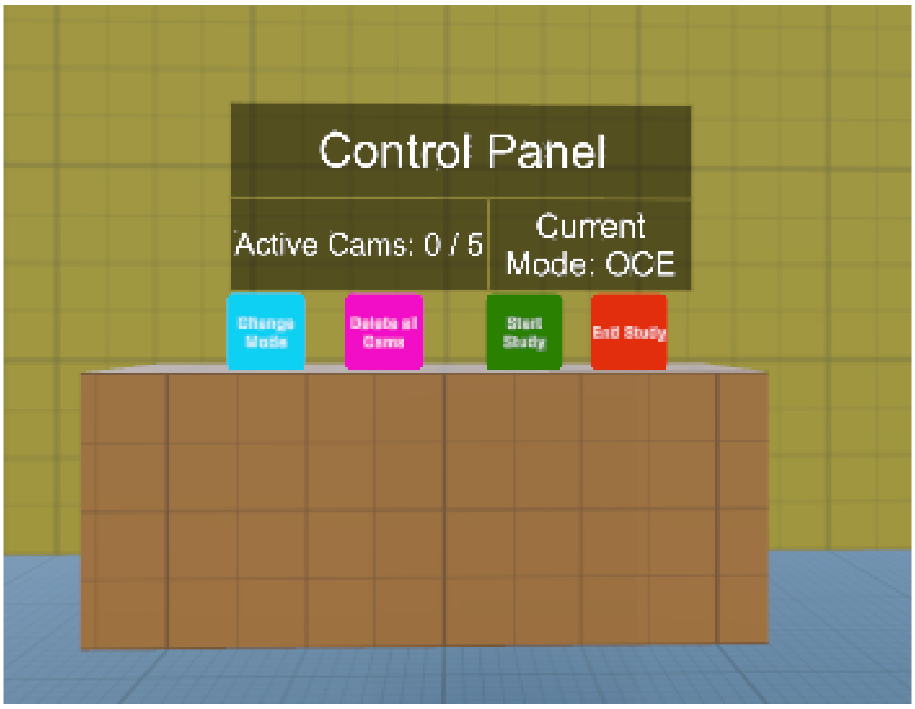

Figure 11: A Control-Table was created. This table displays information and holds buttons for changing the View-Camera mode, deleting all active cameras as well as starting and ending the study.

4.10 Study-Manager

The `StudyManager` script is responsible for orchestrating the entire user study, providing essential functionality for tracking, setting up and ending the study. It handles a wide range of tasks, including the selection of POIs, controlling the diorama's visibility, displaying information to the user and playing audio cues to indicate when a user finds a POI for the first time.

4.10.1 Tracking and Data Logging

To track and record study data, the *Study-Manager* utilizes a logging package from the Unity Asset Store [8]. This package allows the collection of data regarding the user's interactions and progress during the study. The logged data includes details such as which POIs the user has located, how long it took to find each POI and other relevant metrics.

4.10.2 POI Configuration

Initially, the study was designed to include six POIs for the user to locate. However, the number of POIs that can be tracked is adjustable, with the current configuration set to three. This decision to lower the amount of POIs to find was done after the pilot study. A study with six POIs was taking too long and was not providing any significant benefit. The *Study-Manager* is responsible for determining which POIs will be used in each instance of the study, ensuring that they are randomly selected and do not repeat.

4.10.3 Diorama Control

The *Study-Manager* also has control over the visibility of the diorama used in the study. This allows for the diorama to be enabled or disabled depending on the stage of the experiment, ensuring that the user is only interacting with the environment when appropriate.

4.10.4 Information Text and Audio Cues

Throughout the study, the *Study-Manager* provides real-time feedback to the user in the form of information texts and audio cues. For example, when the user locates a POI for the first time, an audio cue is played and information is displayed to confirm that a POI has been found. This feedback is vital for the user's understanding of their progress during the study and helps keep them engaged.

4.10.5 Study Lifecycle Management

The *Study-Manager* is also responsible for managing the lifecycle of the study. It handles the setup process, ensuring that all necessary components are in place before the study begins. Once the study is completed, the *Study-Manager* triggers the termination sequence, saving the logged data and preparing for the next user or study session. The system provides a clear transition between study phases, from preparation to execution to conclusion, ensuring that each participant's data is correctly recorded.

8. https://assetstore.unity.com/packages/tools/input-management/fast-log-to-file-73210?srsltid=AfmBOor6xPlU0zS9U3Qt9Re1n5w2lgWfGhqiGxjapO-WTlAKdQga5Fgt

In summary, the `StudyManager` script serves as the central control hub for the user study, managing the selection of POIs, the presentation of information and the logging of data while maintaining control over the study's flow and user experience.

5 User Study

The two systems are compared to each other in a user study in order to find out their differences in usability, task difficulty and VR sickness. In addition, the user study was used to determine the effect of control schemes on the embodiment of the drone system.

Design. The user study is a within-subject study with the independent variable of which system is used. This independent variable has two conditions, the study is either conducted in *OCE* or in *drone* mode. This means a participant has to find three hidden POIs in a diorama with one of the systems, then fill out a set of questionnaires and afterward do the same again with the other system. Note, however, that for the second task, three other POIs are used. During the task, the users' performance is tracked. The result of the questionnaires in combination with the performance gives information on what system is more suitable for the task.
During the task with the drone system, additional data is tracked on what Control-Scheme is used. In combination with embodiment questions during the questionnaire part this gives further insight into the influence of Control-Schemes on embodiment.
In order to simulate a museum environment, a virtual large-scale diorama of a medieval town was created using low poly synty asset packages [9]. See Figure 12 for an overview of the diorama.

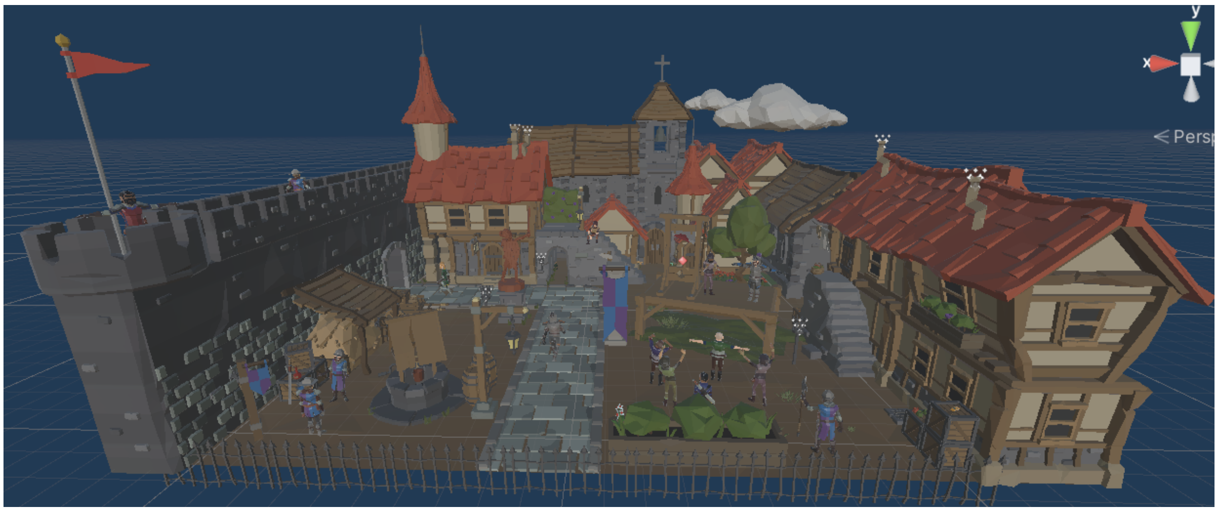

Figure 12: A virtual large-scale diorama of a medieval town was built to simulate a museum environment.

9. https://syntystore.com/collections/polygon-series

Historic objects are often fragile, requiring minimal physical interaction to preserve their integrity. To display such items, they are typically protected behind glass or kept at a distance from visitors. Large-scale dioramas present an opportunity to display original musealia within a suitable setting that reflects the object's historical surroundings. In contrast, a miniature diorama cannot feature the original piece and usually lacks detail. However, a large-scale diorama would still need to be off-limits to direct visitor interaction since it includes the original historic objects. The View-Camera system addresses this limitation, allowing users to closely examine the entire diorama from any viewpoint.
In this diorama, the three POIs were placed. The POIs are symbolized as objects that do not belong, for instance, a burger does not belong in a medieval setting. See Figure 13 for an example of one such POI. Originally five POIs were planned, but pilot tests have shown too many POIs just make the user study overly time-consuming without giving any benefit. Three POIs are enough to compare the two systems.
The diorama is also very suitable for a user study. It allows for plenty opportunity to place

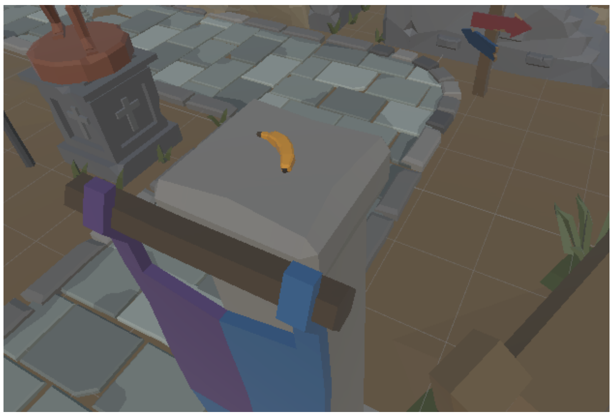
Figure 13: Three such POIs are hidden in the diorama for the user to find with the two presented methods of navigating and manipulating multiple viewpoints.

POIs at various locations for a great diversity of viewpoints that participants need to spot. The participants are tasked to find POIs with both systems in this diorama, the POIs change however. In order to consider all POIs they need to have all POIs in view at the same time, meaning they need to have multiple View-Camera at the same time that see the POIs.

Data Collection. During the study, the participant's performance is tracked. This includes the dependent variables of the **completion time**, **which POIs** were used, how many **View-Cameras** were **spawned and deleted** and how often **View-Panels** were **docked and undocked**. Every POI is also tracked individually on **when** it was **first found** and **how often** it **disappeared from view**. For the drone system specifically, it was also tracked **how much time** was spent in the **drone-centric and user-centric** Control-Scheme. In order to compare the two methods the VRSQ (Kim et al. 2018), the System Usability Scale (SUS) (Brooke 1995) and the NASA TLX (Hart and Staveland 1988) are used. The questionnaires were filled separately for each system right after the participant had completed finding the POIs. The survey was conducted on a tablet device and participants had to remove the HMD. Additionally to these questionnaires, a small survey regarding the participant's embodiment is compiled. For both the OCE and drone approach, a scale on which participants mark their SoE is provided. See Figures 14 and 15 for the scales in question. For the drone approach, another scale regarding the preferred Control-Scheme is provided. See Figure 16 for this scale. This allows to identify a possible correlation between Control-Scheme and SoE and can also be compared to the actual time in each Control-Scheme which is tracked during the study. The scale is inspired by the scale used by Hoppe et al. (2022). After the completion of both tasks, the participants were verbally interviewed for any remarks and they were also asked about their preferred approach.

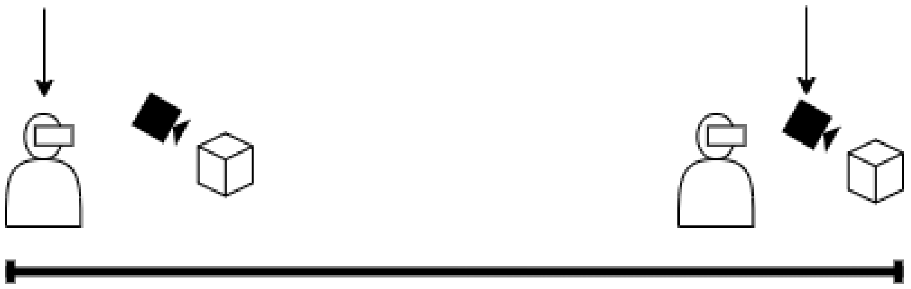
Figure 14: The participants are asked to mark their SoE on an embodiment scale for the OCE approach

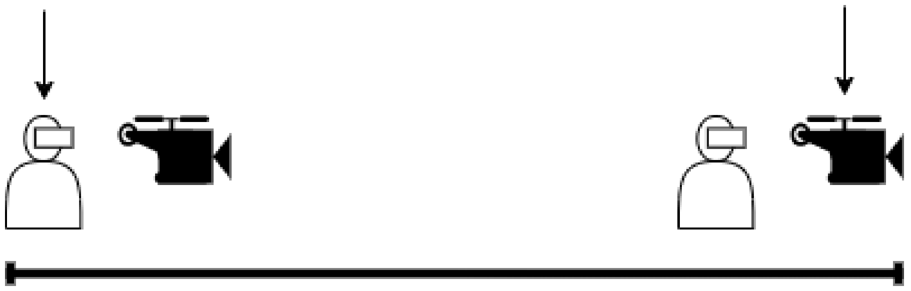
Figure 15: The participants are also asked to mark their SoE on an embodiment scale for the drone approach

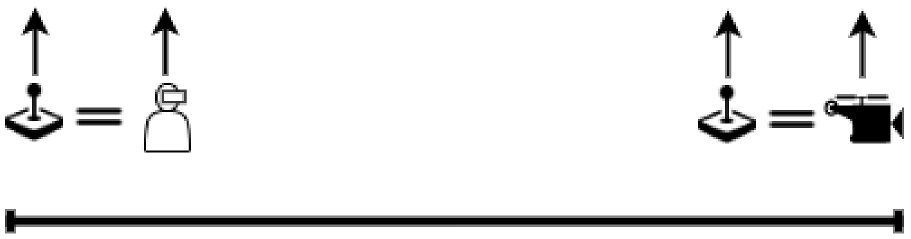
Figure 16: Specifically for the drone approach participants are also asked to mark their preferred Control-Scheme on this scale

Besides the questionnaires participants were verbally interviewed for any feedback and observations were noted during their task performance.

Procedure. The participants for the user study consist of the closer social environment of the author. A Meta Quest 2 is used for the study. Prior to the study, participants signed an informed consent and filled out a demographic questionnaire. Afterwards, the participants are introduced to the subject matter and are taught the system controls. Pilot tests have shown that explaining the study and teaching the controls of two systems is quite time-consuming, especially if the participant is not familiar with VR and basic locomotion and interaction in VR. In order to streamline and make the procedure faster a video introduction was prepared which shows the participants what they can expect in the study. This makes it easier to explain the participant the concept of multiple viewpoints and their task with said viewpoints. After they have been introduced to their task with the video, they are put into VR for them to learn and explore the systems themselves. Detailed button controls for the system are explained to them verbally here as well. Besides the verbal instructions, multiple diagrams of the controls are displayed in VR. Once the participant feels sufficiently confident with the controls they are sent into the user study. The study follows a within-subject design, meaning every participant is exposed to both suggested systems for navigation and manipulation of viewpoints. Participants start out with one of the two systems. The starting system alternates between the participants. The three POIs they need to find are randomly selected from 15 possible POIs. After they have completed finding the POIs participants are interviewed with the corresponding questionnaires of the system. Afterwards, a new set of three POIs is randomly selected from the POIs, note however that not the previous POIs can be selected, this makes sure the participants do not know the location of the POIs beforehand. In case participants take too much time and cannot find all three POIs they are able to end the study early, this is then noted in the tracked data. On average the full procedure takes 30 to 50 minutes depending on the speed at which the participant completes the task and on how long the introduction takes.

Participants. As already mentioned, all the participants for the study were drafted out of the closer social environment of the author. In total twelve (7 female, 5 male) participants took part in the study. The average age of the participants was 26. Most of the participants stated they had used VR once or twice already. The standard deviation is *0.79*, where zero is 'Never used' and four is 'Frequently use'. See Figure 17 for a detailed overview of the participants VR experience.

Hypotheses. The following hypotheses are suggested:

H1 Regarding **RQ1**, it is hypothesized that the HoverCam (Khan et al. 2005) OCE approach will lead to faster exploration due to its intuitive interface, which allows for quick adjustments of the look-at targets via a simple ray cast mechanism.

H2 Regarding **sRQ1**, it is hypothesized that the drone camera leads to less cybersickness since it moves smoother. The 'look-and-fly' feature of the HoverCam is an abrupt jump which might induce more cybersickness.

H3 Regarding **sRQ3**, it is hypothesized that no Control-Scheme will have a clear majority since it depends on the embodiment of an individual participant.

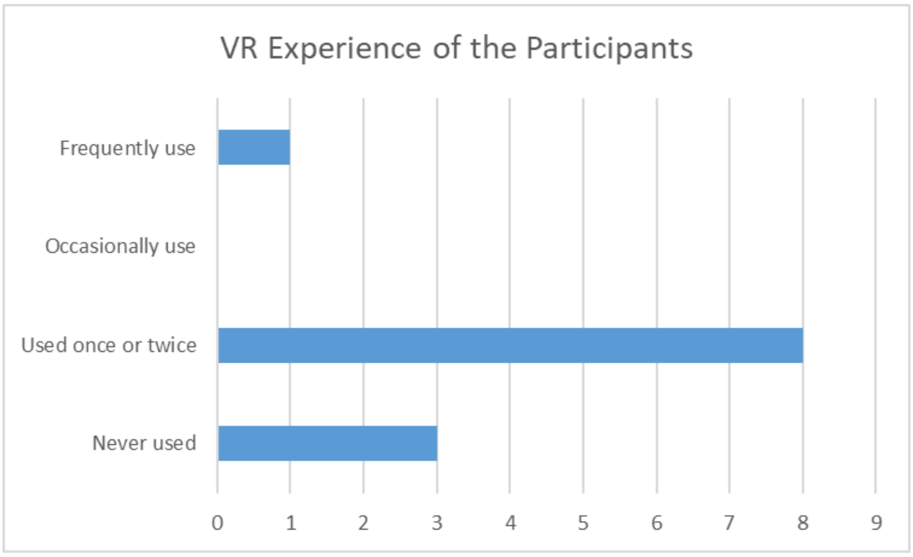

Figure 17: Most of the twelve participants of the user study stated they had used VR once or twice already.

H4 Regarding **sRQ4**, it is hypothesized that a correspondence between Control-Scheme and embodiment of the participants can be observed. Participants who prefer the drone-centric controls will more identify with the drone in the embodiment spectrum. This will be due to it better corresponding with controls for controlling a first-person drone. Therefore leading to better SoBO.

6 Results

The users' study concluded with the following results. On average users completed the task with the hover camera in *6.77min* and with the drone camera in *4.76min*. The hover camera displays a standard deviation of *2.85min* and the drone camera of *2.36min*. Figure 18 displays the task completion times in a boxplot and Table 2 shows the task times.

Task Completion Times		
	Hover Camera	Drone Camera
Average	6.77min	4.76min
Std Dev	2.85min	2.36min

Table 2: The task completion times show that the drone camera is faster.

Out of the twelve participants *ten preferred the drone camera*. Only two chose the hover camera as their favorite.

The questionnaires showed the following results.

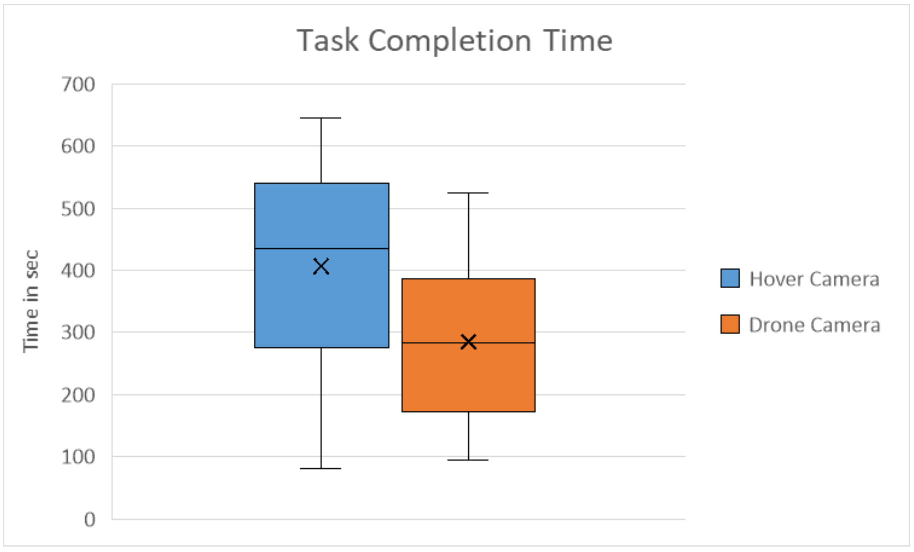

Figure 18: The task completion times of the hover and drone camera show that on average the drone camera is faster.

SUS. The average SUS-Score for the *drone camera* is *73.8* with a standard deviation of *15.39*. The average SUS-Score for the *hover camera* is *57.9* with a standard deviation of *18.61*. See Figure 19 for a boxplot of the SUS-Scores.

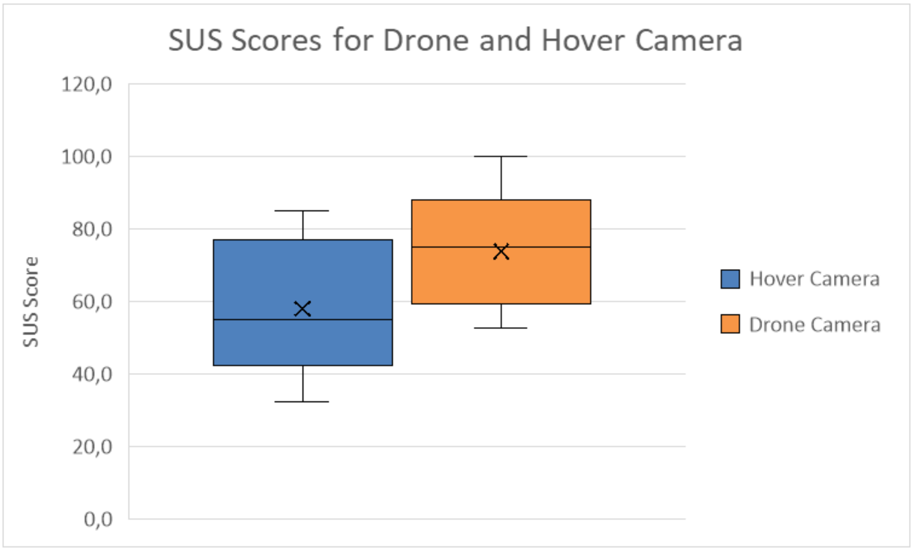

Figure 19: The drone camera displays a higher SUS-Score than the hover camera.

VRSQ. The Total Score for the VRSQ of the hover camera is *21.87* with a standard deviation of *16.4*. The drone camera scored a Total Score of *12.22* with a standard deviation of *14.69*. The Figure 20 show the boxplots for the VRSQ. It is also noteworthy that the drone camera has an average oculomotor score of *13.88* whilst the hover camera has an average score of *29.86*.

NASA TLX. Due to a slip-up, the raw NASA TLX was used instead of the weighted version.

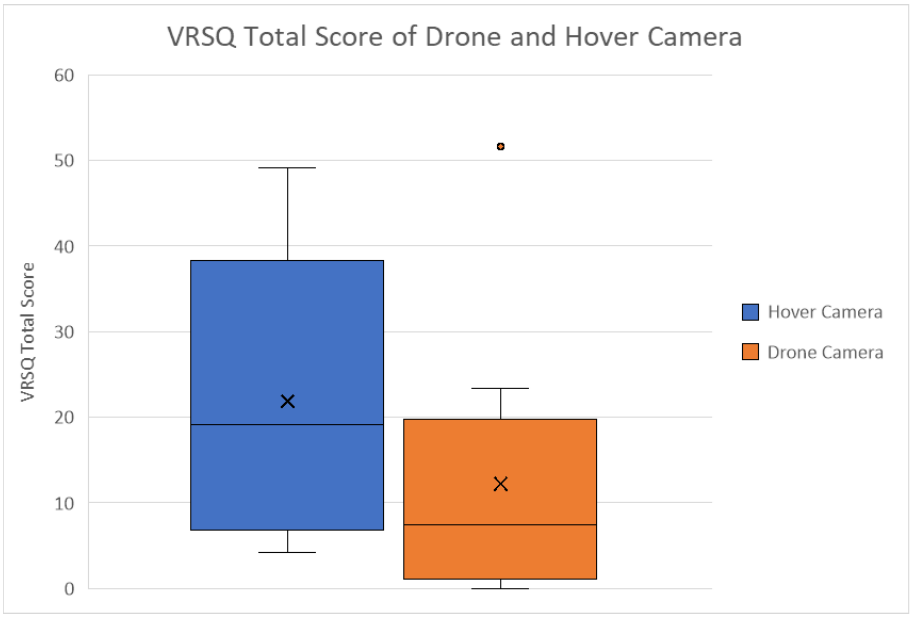

Figure 20: The drone camera displays a lower Total Score in the VRSQ.

However, Hart (2006) states that the raw NASA TLX, while simpler to apply, does not show a significant difference compared to the weighted version. The results of the raw NASA TLX are shown in the following Table 3.

NASA TLX Results				
	Hover Camera		Drone Camera	
	Avg	Sd	Avg	Sd
Mental	40.00	21.11	36.67	24.25
Physical	20.45	17.34	16.82	17.12
Temporal	36.25	24.61	30.00	27.56
Performance	42.92	22.31	25.91	18.96
Effort	43.75	23.27	25.00	16.38
Frustration	52.50	30.41	27.50	24.54
Overall	**37.92**	**10.70**	**27.01**	**6.51**

Table 3: The results of the NASA TLX show the drone camera has a better score than the hover camera.

Embodiment Scales. The embodiment part of the user study was not conclusive. The embodiment of the drone and hover camera were varying. See Figure 21 for the results. The embodiment for the hover camera is on average *0.608* with a standard deviation of *0.34*. Where zero is a greater embodiment towards the user and one is a greater embodiment towards the camera. In comparison, the drone camera displays an average embodiment of *0.72* with a standard deviation of *0.29*. See Figure 22 for the perceived embodiment boxplots of the two modes.

The preferred Control-Scheme for the drone camera is leaning towards the drone-centric ap-

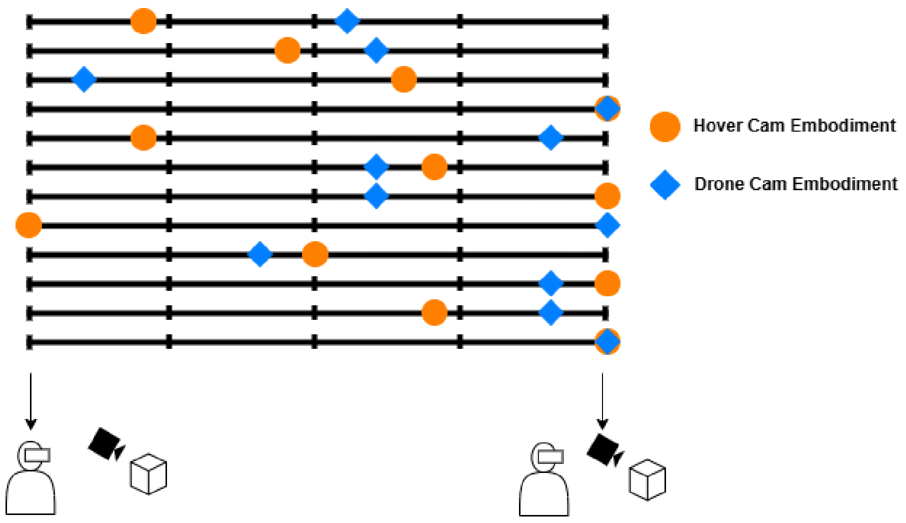

Figure 21: The embodiment values for both modes are not conclusive.

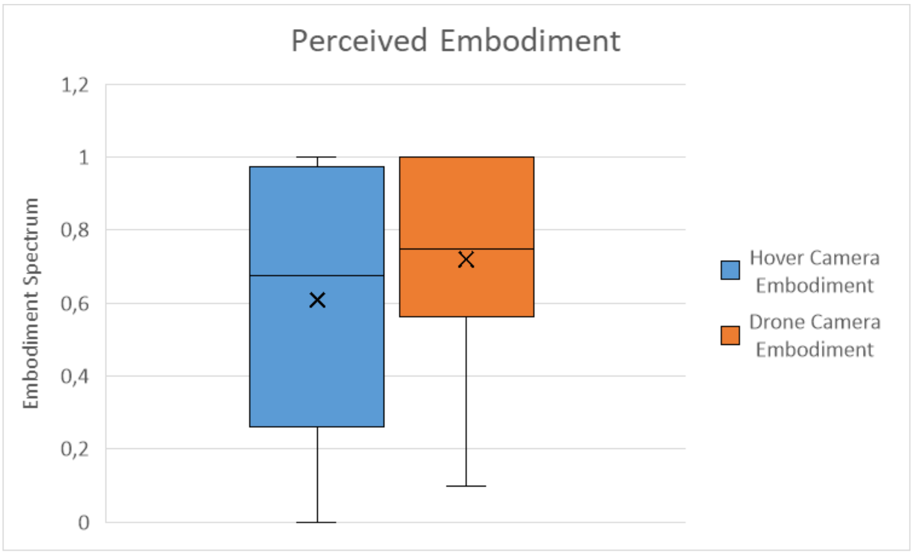

Figure 22: The hover camera displays a greater deviation in perceived embodiment in comparison to the drone camera. Both modes do not display a conclusive embodiment.

proach. The average preferred Control-Scheme is *0.69* with a standard deviation of *0.41*, one being drone-centric and zero being user-centric. See Figure 23 for a boxplot of the preferred Control-Scheme and see Figure 24 for the participant's individual results.

When converted to an absolute scale (where >0.5 indicates a preference for the drone-centric control scheme and <0.5 indicates a preference for the user-centric control scheme), **eight** participants preferred the drone-centric scheme, while **four** preferred the user-centric scheme.

The tracking during the user study shows that on average participants spent *123.56s* in the user-centric scheme with a standard deviation of *115.32s* and *109.59s* in the drone-centric scheme with a standard deviation of *103.95s*. The time spent in the different Control-Schemes is visu-

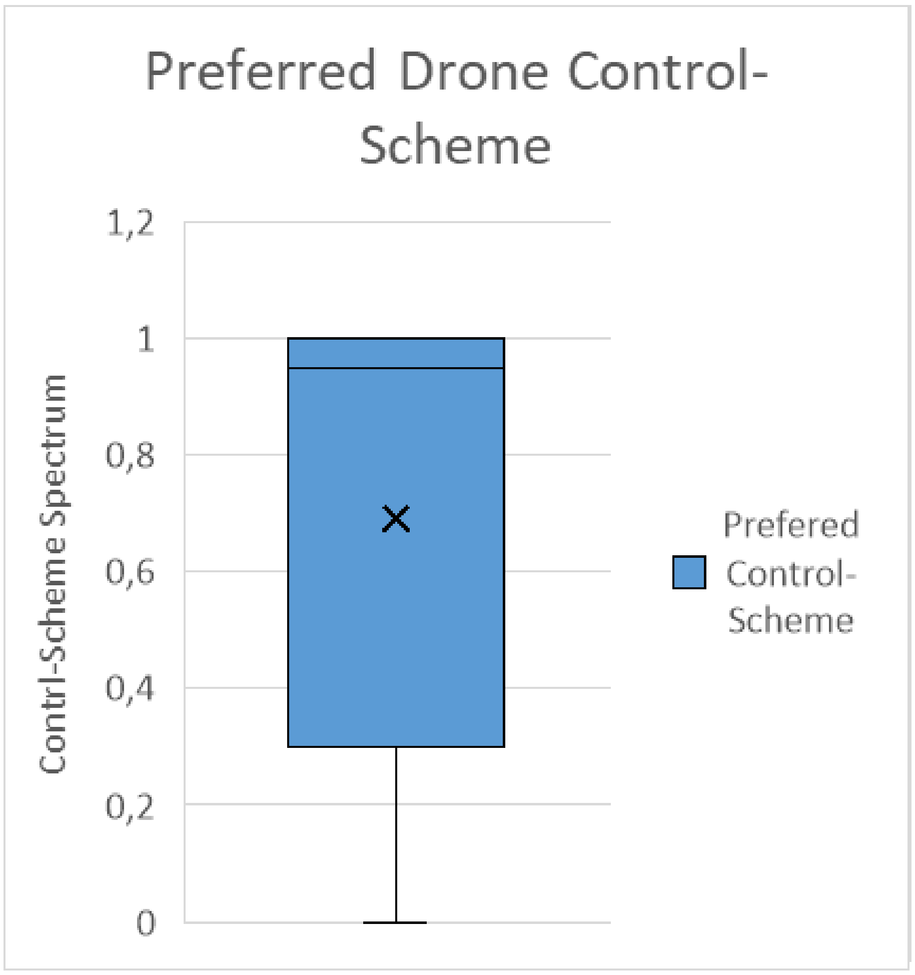

Figure 23: The drone-centric approach is one on the Control-Scheme Spectrum and the user-centric approach is zero. The participants of the study preferred the drone-centric approach on average.

alized in Figure 25.

Note here, that per default every Drone-Camera starts in the user-centric scheme and participants needed to swap to the drone-centric scheme by hand for every newly spawned Drone-Camera. Also noteworthy is that during the user-study three of the participants never swapped to the drone-centric scheme at all.

No correlation between perceived embodiment and Control-Scheme could be found. The Pearson correlation resulted with a value of *-0.02098* for r and a p-value of *0.948186*.

7 Findings and Discussion

The findings of this user study provide insights into the research questions and allow for a critical review of the hypotheses.

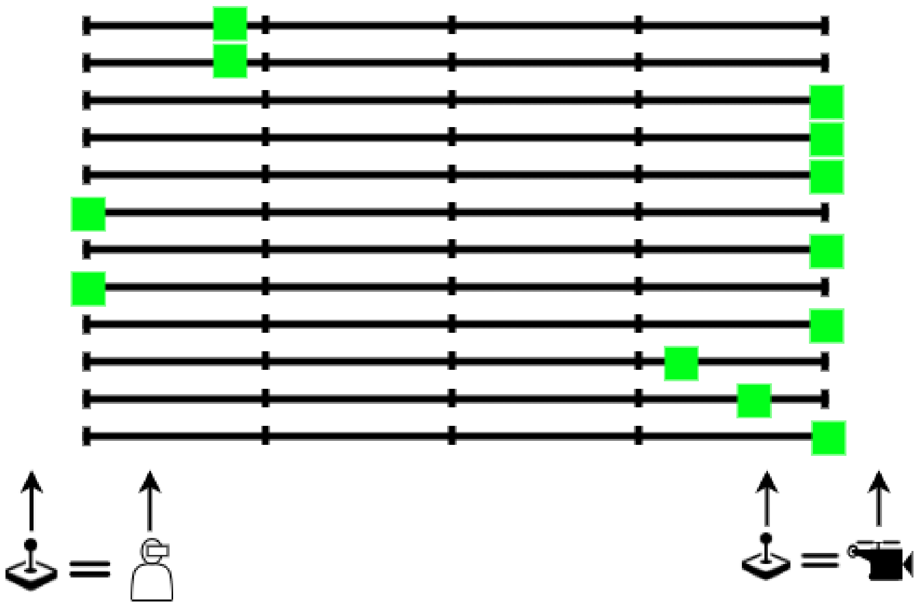

Figure 24: The participants marked their preferred Control-Schemes on a scale. Most of them prefer the drone-centric Control-Scheme.

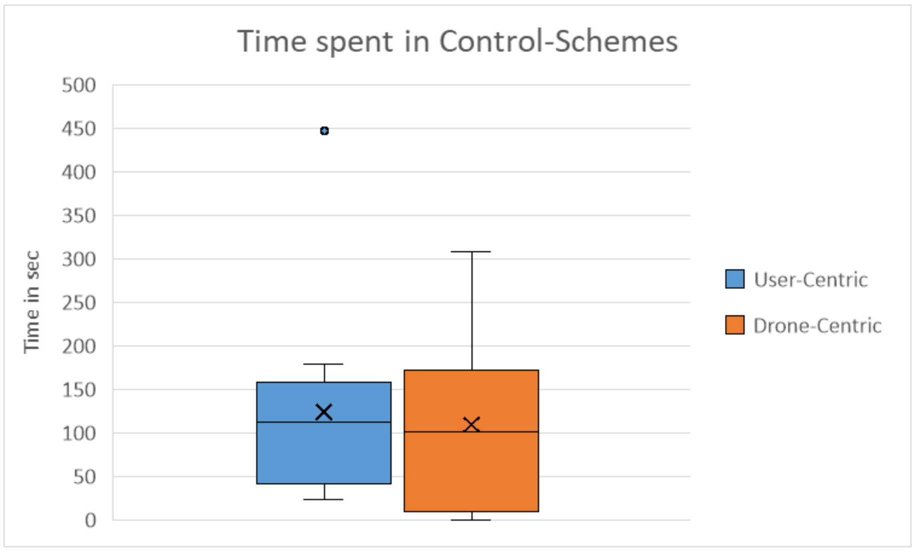

Figure 25: On average more time is spent in the user-centric scheme than in the drone-centric.

7.1 Research Questions and Hypotheses

For **RQ1**, it was hypothesized that the *HoverCam* OCE approach would be superior due to its intuitive interface, which enables quick adjustments through ray casting. However, results indicate that the *Drone Camera* was both preferred and more effective overall. Detailed results for the sub-research questions follow below.

The **sRQ1** examined which approach led to greater cybersickness. Hypothesis **H2** predicted that the *HoverCam* would cause more cybersickness, as its 'look-and-fly' feature involves abrupt movements without smooth transitions. This hypothesis is supported by the VRSQ scores (see Figure 20). On average, the *Drone Camera* scored *12.22*, indicating only mild symptoms, whereas the *HoverCam* scored *21.87*, indicating moderate symptoms.

For **sRQ2**, the aim was to assess which approach would lead to better task performance and exploration. Participants completed the task significantly faster using the *Drone Camera* (4.76min) compared to the *HoverCam* (6.77min), as shown in the completion time boxplot (Figure 18). Usability scores further support this, with the SUS results showing an average usability score of *57.9* for the *HoverCam* (indicating moderate usability), versus *73.8* for the *Drone Camera* (considered acceptable usability).

The NASA TLX results (see Table 3) also emphasize the superiority of the *Drone Camera* with lower perceived workload scores. Notably, the *HoverCam* scored a frustration level of *52.50*, significantly higher than the *Drone Camera's* score of *27.50*. During the study, participants frequently commented on the difficulty of gaining an overview of the diorama with the *HoverCam*. Additionally, ray casting was sometimes inaccurate, making it challenging to select distant objects due to occlusion by surrounding objects or the View-Panel positioned in front of the user. Elmqvist and Tsigas (2008) discuss multiple viewpoints as a solution for occlusion. While multiple viewpoints may improve the overview, they do not address selection accuracy in this case. A modification allowing users to ray cast through a View-Panel to select objects could mitigate these issues, potentially improving navigation and reducing frustration with the *HoverCam*. In addition, highlighting a hovered-over object leads to better feedback and therefore better usability, Argelaguet and Andujar (2013) confirms the importance of feedback for selection in a 3D space. For highlighting, an outline around the object could be used. Outlines have already been extensively used in this user study, if the system were to be tested in AR or MR at some point, other options for indication could be evaluated. Fuchs, Sigel, and Dörner (2016) have already tested multiple novel approaches for highlighting objects in AR.

In conclusion, for **RQ1**, the *Drone Camera* was more effective overall. It allowed for quicker task completion, reduced cybersickness, better usability and ease of use, as demonstrated by the VRSQ, SUS and NASA TLX results. Participants also expressed a clear preference for the *Drone Camera*.

sRQ3 and **sRQ4** explore the embodiment of participants during the study.

For **sRQ3**, Hypothesis **H3** suggested that no control scheme would be clearly preferred. However, a majority favored the drone-centric scheme (8 of 12 participants). Notably, participants spent more time in the user-centric scheme (123.56s on average) than in the drone-centric scheme (115.32s). This may be attributed to the fact that the *Drone Camera* defaults to the user-centric mode and despite a detailed tutorial, some participants forgot they could switch modes. Additionally, half of the participants used the *HoverCam* before the *Drone Camera*, which does not have multiple control schemes, potentially leading some to overlook this functionality in the *Drone Camera*. Three participants did not use the drone-centric scheme at all.

For **sRQ4**, no significant correlation between control scheme and SoE was found (Pearson's r = -0.02, p = 0.95). Hypothesis **H4** is therefore rejected. Despite Kilteni, Groten, and Slater (2012) suggesting that aligning physical and visual stimulation can enhance SoBO, this alignment did

not increase SoE toward the drone in the drone-centric scheme. A possible explanation is that a mere change in joystick orientation is insufficient to create a meaningful sensory link between the physical body and the virtual representation. Joystick movement itself may not create a strong enough correlation with body movement in VR, which could explain the lack of significant impact on embodiment.
Overall, the participants' embodiment was not polled sufficiently. The scale was the only measure for embodiment. It was inspired by Hoppe et al. (2022) who used a similar scale, however, they also used additional questionnaires such as the embodiment questionnaire by Gorisse et al. (2017), the IGroup's Presence Questionnaire (IPQ) and the Games Experience Questionnaire (GEQ). Since the embodiment was only considered an interesting sub-research question that emerged out of curiosity, it was not examined in greater detail. Therefore, the result of the **sRQ4** is limited and would require further research with more conclusive questionnaires for a meaningful result.

Besides this, another observation was that participants often shifted their focus between the View-Panel and the diorama, with some reporting changes in perceived embodiment over time. This raises an intriguing possibility: that mere focus shifts might influence embodiment. Further research is needed to explore this potential relationship, as it was only observed anecdotally in this study. Thus, the interaction of embodiment, control schemes and visual focus represents a promising direction for future work.

7.2 POI Search Time Distribution

The tracked data from the user study indicates the hidden POIs were not all equally well hidden. See Figure 26 for a boxplot of each POI.

In the study, users had to find three randomly selected POIs out of the 15 possible POIs. With a great enough amount of participants, this would not be a problem since overall, both drone and hover camera would end up with the same amount of "hard" and "easy" to find POIs. However, since this study only consisted of twelve participants this is not the case. Take for example the *HMD* POI. It was part of six tasks, four of them being with the drone camera and two with the hover camera. Since the *HMD* is one of the "easier" POIs, the drone camera had an advantage over the hover camera. As a comparison, also take a look at the *Boxing Gloves* POI. This POI was part of five tasks, all of which were conducted with the hover camera. Overall, due to the small amount of participants, the study is skewed.

7.3 Participant Feedback and Possible System Improvements

This section consolidates the feedback gathered from participants during the verbal interviews, as well as observations recorded throughout the user study. These insights provide a basis for evaluating the current system and identifying potential avenues for refinement. The remarks, both spontaneous and prompted, are analyzed here to suggest enhancements for the system and address areas that presented challenges or opportunities for improvement.

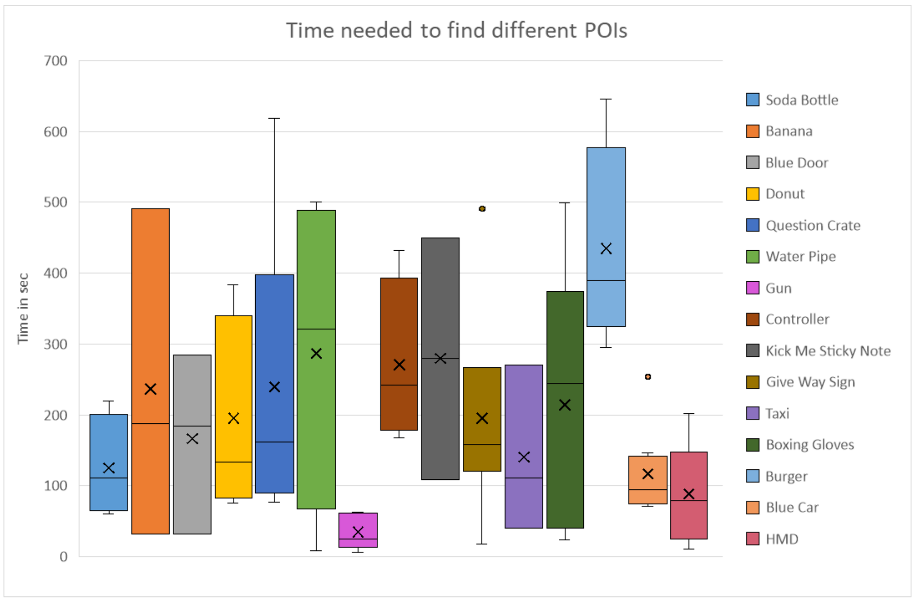

Figure 26: The tracked data from the user study indicates the hidden POIs were not all equally well hidden.

7.3.1 HUD Feedback

Participants were introduced to the ability to transfer View-Panels to the HUD, offering a potentially more convenient means of panel management. Despite the feature being explained during the training phase, its usage was minimal. Only one participant utilized the HUD functionality during the task and another explicitly described it as ”useless.” This feedback points to either a lack of relevance for the task at hand or a need to rethink how the feature is presented and integrated.

The limited adoption of the HUD suggests that participants did not perceive it as valuable or intuitive. Future iterations of the system could benefit from clearer integration of the feature within the task flow, potentially by emphasizing scenarios during training where the HUD provides a tangible advantage. Alternatively, the feature's design could be simplified or omitted entirely to reduce cognitive load and streamline the system.

7.3.2 Camera Feedback and Customization Needs

Hover Camera Participants commonly expressed dissatisfaction with the hover camera when tasked with navigating the large diorama. While it was noted that the hover camera was effective for inspecting objects at close range, several limitations became evident:

- **Restricted Field of View:** The hover camera's perspective was not advantageous to identifying hidden objects in the expansive diorama. Some participants attempted to zoom out or position the camera higher in an effort to gain a broader overview, but these adjustments often fell short.

- **Navigational Limitations:** The hover camera performed poorly in tight or cluttered areas, such as corners, where visibility and maneuverability were constrained.
- **Focus Discrepancies:** Participants frequently zoomed out for an enhanced overview but, in doing so, ignored or lost focus on the object that the camera was originally targeting.

The prime problem of the hover cam was the selection of distant objects. The ray cast made it challenging to discern which object was currently being targeted. Users expressed that having additional visual feedback would be beneficial in this context. Adding an outline around the hovered object could help clarify selection, reducing frustration and improving accuracy when interacting with far-off objects. This feature would likely enhance the intuitiveness and usability of the selection process.

These observations underscore that while the hover camera is well-suited for detailed object interactions, its application in larger, more exploratory tasks is limited. Adjustments to its field of view or navigational mechanics could make it more versatile.

Drone Camera The drone camera was preferred by participants for tasks requiring broader scene exploration. However, several participants highlighted the need for greater control customization to better suit their individual preferences. Desired adjustments included:

- Customizable movement speed and zoom sensitivity.
- The ability to invert axis controls for look rotation.
- Setting a default mode for the drone camera that aligns with user preferences.

Additionally, participants suggested reconfiguring joystick functions to make control more intuitive. Specifically, assigning the right joystick to control movement while allowing the left joystick to adjust vertical positioning was recommended.

Shared Feedback With both the drone camera and the hover camera it is possible to navigate the camera into objects and see the meshes from the inside. These viewpoints are often disorienting and lead to losing overview of where the camera is located at. A possible solution for this is to not allow the camera to fly into meshes. For instance Ortega, Stuerzlinger, and Scheurich (2015) introduced *SHOCam* which is an improved version of the HoverCam with this exact feature.

Participants remarked that the perceived embodiment changed over time. Depending on their current task and focus they either felt more like the camera or the operator of it. They especially remarked on this when they had to mark their perceived embodiment on the embodiment scale during the questionnaire. Besides this one participant also remarked that the preferred camera mode changed as well. Depending on if they see the object they focus on, they prefer the hover or drone camera. Preferring the hover camera when the object is close and visible and the drone camera if not.

7.3.3 View-Panel Organization and Interaction Patterns

Participants displayed a variety of strategies for organizing and managing View-Panels during the study. Most participants followed a sequential workflow:

1. Spawn a View-Pair.
2. Relocate the View-Panel to a desired spot.
3. Locate a POI.
4. Move the View-Panel aside.
5. Continue from 1. until all POIs are found.

While this approach was prevalent, some participants adopted unique methods:

- **Head Movement for Placement:** One participant preemptively moved their head before spawning a new View-Panel to control where the panel appeared.
- **Sky Placement:** Another participant preferred positioning panels high above the scene to keep them visually unobtrusive.
- **Symmetrical Organization:** Several participants consistently placed panels symmetrically to their left and right, typically at chest level for easy access.

These diverse approaches suggest a need for more advanced View-Panel management tools. Features such as automated organization, stackable panels, or a grouping function could provide participants with a more efficient way to handle multiple panels simultaneously.

7.3.4 Outline and Highlighting Features

The system employed outline effects to highlight important objects and cameras:

- **Blue Outline:** Indicates the focused object of the hover camera.
- **Green Outline:** Highlights a selected View-Camera.
- **Yellow Outline:** Marks discovered POI.

While these outlines proved useful for object identification, participants noted instances of confusion caused by overlapping outlines. For example, when a focused object was simultaneously visible through a View-Panel and directly within the user's field of view, duplicate outlines created visual clutter. Figure 27 demonstrates this issue.

To address this, the system could be modified to allow View-Panels to occlude outline effects. This adjustment would reduce visual noise, making the highlighting system more intuitive and less distracting.

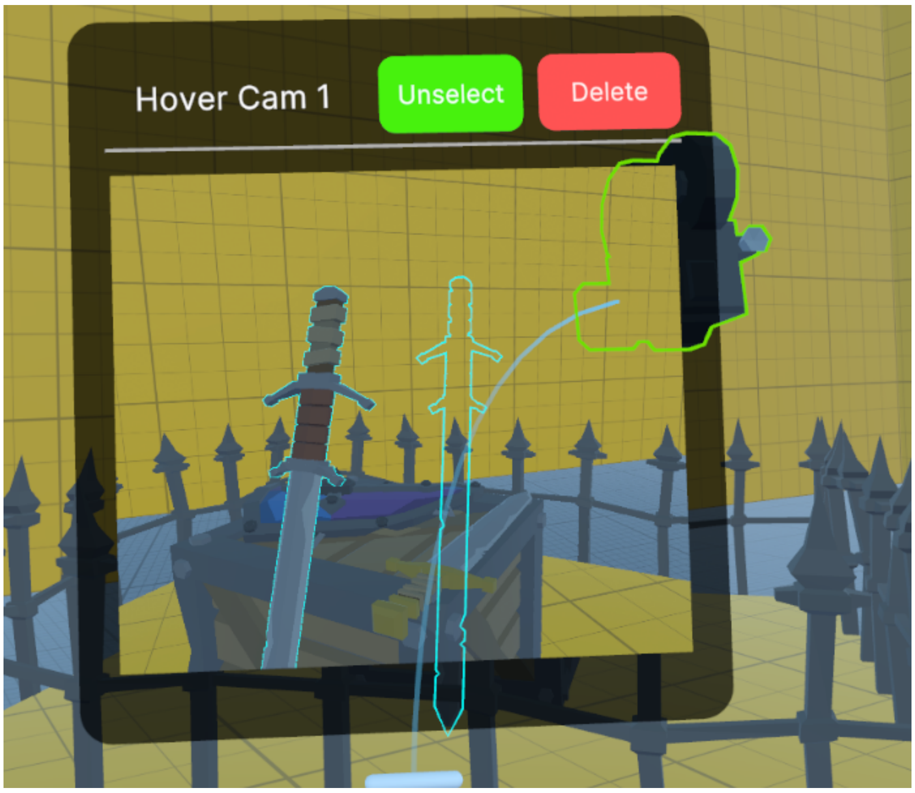

Figure 27: The hover camera shows a blue outline around its focused object. A selected View-Camera is also indicated with a green outline. These outlines shine through objects. This can lead to confusion if they shine through the View-Panel, which in turn also displays the blue outline the View-Camera sees.

7.3.5 Button Mapping and Interaction Issues

Participants often struggled with the button mappings, particularly when interacting with UI buttons. Many mistakenly used the ‘A’ button, which was assigned to spawn a new View-Pair, instead of the Trigger button designated for interaction. This confusion interrupted the workflow, especially when participants had to clean up unintended View-Pairs.

In part, the issue might be caused by the hover camera’s ‘look-and-fly’ feature, which also used the ‘A’ button. Participants were already used to interacting with the ’A’ button due to this. Reassigning this functionality to the Trigger button would better align with user expectations of having a single interaction button, improving usability and minimizing frustration.

Notably, many participants appeared to forget the existence of the ’look-and-fly’ feature, despite it being thoroughly explained during the training phase. They kept on deleting their newly spawned drone camera to then spawn a new camera on a new object. A potential explanation for this is the study’s within-subject design. Half of the participants used the drone camera before switching to the hover camera. As the drone camera lacks the ’look-and-fly’ feature, participants may have been primed to disregard it. Conversely, participants who used the hover camera first did not have a problem when spawning and moving the drone camera.

Another issue observed during the study was the unintentional movement of the hover camera while using the 'look-and-fly' feature. When participants identified a POI, their typical intention was to leave the current camera in place and spawn a new one to continue exploring. However, participants often forgot to deselect the hover camera before using the 'look-and-fly' feature, inadvertently moving it away from the identified POI. This forced them to backtrack and reposition the camera, interrupting their workflow.

This problem may also stem from inconsistencies in camera control mechanics between the two systems. For example, the drone camera allows spawning a new camera with the 'A' button, regardless of whether a camera is currently selected. In contrast, the hover camera spawns a new camera only if no other camera is selected. This discrepancy may have contributed to participant confusion.

To address these issues, a potential improvement could involve assigning distinct buttons for the 'look-and-fly' feature and the spawning of new cameras. This adjustment could help reduce confusion and improve the overall user experience.

7.3.6 Proposed Combined Camera System

Based on participant feedback, a hybrid camera system is proposed to capitalize on the strengths of both the drone and hover cameras:

- **Primary Functionality:** The system operates as a drone camera, enabling users to navigate large environments effectively.
- **Object Selection:** Users can ray cast through the View-Panel to highlight objects of interest, with hovered objects outlined for clarity. Fuchs, Sigel, and Dörner (2016) researched methods for highlighting objects in AR. Their conclusion lists *Hotplate* as a suitable option for highlighting. *Hotplate* being a technique that displays swirling upwards-moving particles originating from the base of the object.
- **Seamless Transition:** Selecting an object initiates a smooth transition into hover camera mode, allowing detailed inspection of the object without abrupt viewpoint changes.
- **Mode Switching:** A button within the View-Panel's UI allows users to return to drone camera mode, facilitating continued exploration.

This combined system balances the drone camera's expansive navigational capabilities with the hover camera's precision for object-focused tasks. Incorporating additional improvements, such as customizable controls, refined button mappings and enhanced highlighting, would further elevate the system's usability and overall participant satisfaction.

Kim and Xiong (2022) researched a similar approach for tackling the problem of distant object selection. They proposed and tested *ViewfinderVR* to overcome the limitations of ray casting. *ViewfinderVR* utilizes a virtual viewfinder panel which is a modern adaptation of the through-the-lens metaphor (Gleicher and Witkin 1992). The viewfinder panel acts as a virtual mirror that can be placed and manipulated. The mirror allows for the selection of objects via ray casts or touch that are seen in its reflection. A Fitts' law-based test (Fitts 1954) and NASA TLX results show the *ViewfinderVR* outperforms regular ray-cast selection. It is expected that similar improvement can be achieved with the proposed combined camera system.

8 Conclusion and Future Work

This thesis explored the manipulation of multiple viewpoints in VR for a museum setting, investigating how this could offer novel ways to exhibit objects. While conducted in VR, technologies such as those described by Lindlbauer and Wilson (2018) could make similar experiences possible in MR. The scenario of a large-scale diorama was used as an example. A user study compared two distinct viewpoint manipulation methods to assess whether a restrictive automated OCE approach or a free-flying drone approach would be more effective.

Participants were asked to find hidden POI objects in the diorama using either a freely controllable drone camera or a modified HoverCam (Khan et al. 2005) with a 'look-and-fly' feature (McCrae et al. 2009). The effectiveness of each approach was measured by cybersickness, usability, task completion time and overall participant feedback. Results showed a preference for the *drone approach*, which also had better outcomes in terms of cybersickness reduction, usability and task completion time. However, participants suggested that the hover camera might be superior for detailed inspection, indicating an area for further study.

Beyond manipulation effectiveness, this study briefly explored embodiment, emerging from two different control schemes used for the drone camera. It was hypothesized that participants' preferred control scheme might correlate with their perceived embodiment—feeling as though they were either controlling the drone or embodying it. This hypothesis was based on Kilteni, Groten, and Slater (2012), who define SoE as including SoBO, which is influenced by avatar alignment with user actions. No significant correlation was found, potentially due to frequent focus shifts. The influence of control schemes on SoE, as well as the impact of focus shifts on embodiment, require further investigation.

A suggested improvement to the system is a combination of the drone camera for scene overview and the hover camera for detailed inspection.

This study focuses on large-scale dioramas, but future research could explore larger environments, such as a historical site for instance where users navigate a virtual castle. With AR, users could view the original state of damaged areas, enhancing historical understanding. Keil et al. (2011) have done something similar already but on a smaller scale, they overlay a historical 3D representation of a building on a mobile phone. Zhang et al. (2021) have used drones in order to create live photogrammetry of objects. Their system is limited by the latency of sending data from drones to a ground station however, it could enable AR viewpoints on a larger scale. Although this thesis focuses on viewpoint manipulation, the potential applications in MR and AR are vast and worthy of further exploration.

References

Argelaguet, Ferran, and Carlos Andujar. 2013. "A survey of 3D object selection techniques for virtual environments." *Computers & Graphics* 37 (3): 121–136. ISSN: 0097-8493. https://doi.org/https://doi.org/10.1016/j.cag.2012.12.003. https://www.sciencedirect.com/science/article/pii/S0097849312001793.

Boubekeur, Tamy. 2014. "ShellCam: Interactive geometry-aware virtual camera control." In *2014 IEEE International Conference on Image Processing (ICIP),* 4003–4007. https://doi.org/10.1109/ICIP.2014.7025813.

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

Cao, Jiaxun, Qingyang He, Zhuo Wang, RAY LC, and Xin Tong. 2023. "DreamVR: Curating an Interactive Exhibition in Social VR Through an Autobiographical Design Study." In *Proceedings of the 2023 CHI Conference on Human Factors in Computing Systems.* CHI '23. Hamburg, Germany: Association for Computing Machinery. ISBN: 9781450394215. https://doi.org/10.1145/3544548.3581362. https://doi.org/10.1145/3544548.3581362.

Chae, Han Joo, Jeong-in Hwang, and Jinwook Seo. 2018. "Wall-based Space Manipulation Technique for Efficient Placement of Distant Objects in Augmented Reality." In *Proceedings of the 31st Annual ACM Symposium on User Interface Software and Technology,* 45–52. UIST '18. ¡conf-loc¿, ¡city¿Berlin¡/city¿, ¡country¿Germany¡/country¿, ¡/conf-loc¿: Association for Computing Machinery. ISBN: 9781450359481. https://doi.org/10.1145/3242587.3242631. https://doi.org/10.1145/3242587.3242631.

Cmentowski, Sebastian, Sukran Karaosmanoglu, Fabian Kievelitz, Frank Steinicke, and Jens Krüger. 2023. "A Matter of Perspective: Designing Immersive Character Transitions for Virtual Reality Games." *Proc. ACM Hum.-Comput. Interact.* (New York, NY, USA) 7, no. CHI PLAY (October). https://doi.org/10.1145/3611023. https://doi.org/10.1145/3611023.

Delgado, Johnny, Rachel West, Angelos Barmpoutis, Seung Hyuk Jang, Edward Stanley, and Hyo Kang. 2024. "Enhancing Museum Experience with VR by Situating 3D Collections in Contex." In *Proceedings of the 23rd Annual ACM Interaction Design and Children Conference,* 670–675. IDC '24. Delft, Netherlands: Association for Computing Machinery. ISBN: 9798400704420. https://doi.org/10.1145/3628516.3659372. https://doi.org/10.1145/3628516.3659372.

Desvallées, André, and François Mairesse. 2010. *Key Concepts of Museology.* Paris, France: Armand Colin. ISBN: 9782200253981.

Dufresne, Florian, Charlotte Dubosc, Geoffrey Gorisse, and Olivier Christmann. 2024. "Understanding the Impact of Coherence between Virtual Representations and Possible Interactions on Embodiment in VR: an Affordance Perspective." In *Extended Abstracts of the 2024 CHI Conference on Human Factors in Computing Systems.* CHI EA '24. Association for Computing Machinery. ISBN: 9798400703317. https://doi.org/10.1145/3613905.3650752. https://doi.org/10.1145/3613905.3650752.

Elmqvist, Niklas, and Philippas Tsigas. 2008. "A Taxonomy of 3D Occlusion Management for Visualization." *IEEE Transactions on Visualization and Computer Graphics* 14 (5): 1095–1109. https://doi.org/10.1109/TVCG.2008.59.

Erat, Okan, Werner Alexander Isop, Denis Kalkofen, and Dieter Schmalstieg. 2018. "Drone-Augmented Human Vision: Exocentric Control for Drones Exploring Hidden Areas." *IEEE Transactions on Visualization and Computer Graphics* 24 (4): 1437–1446. https://doi.org/10.1109/TVCG.2018.2794058.

Fitts, P. M. 1954. "The information capacity of the human motor system in controlling the amplitude of movement." *Journal of Experimental PSychology* 74:381–391.

Fribourg, Rebecca, Ferran Argelaguet, Anatole Lécuyer, and Ludovic Hoyet. 2020. "Avatar and Sense of Embodiment: Studying the Relative Preference Between Appearance, Control and Point of View." *IEEE Transactions on Visualization and Computer Graphics* 26 (5): 2062–2072. https://doi.org/10.1109/TVCG.2020.2973077.

Fuchs, Sebastian, Mario Sigel, and Ralf Dörner. 2016. "Highlighting Techniques for Real Entities in Augmented Reality." In *Proceedings of the 11th Joint Conference on Computer Vision, Imaging and Computer Graphics Theory and Applications - Volume 1: GRAPP, (VISIGRAPP 2016),* 259–270. INSTICC, SciTePress. ISBN: 978-989-758-175-5. https://doi.org/10.5220/0005674002570268.

Gallagher, Shaun. 2000. "Gallagher, S. 2000. Philosophical conceptions of the self: implications for cognitive science." *Trends in Cognitive Sciences* 4 (January): 14–21.

Gleicher, Michael, and Andrew Witkin. 1992. "Through-the-lens camera control." In *Proceedings of the 19th Annual Conference on Computer Graphics and Interactive Techniques,* 331–340. SIGGRAPH '92. New York, NY, USA: Association for Computing Machinery. ISBN: 0897914791. https://doi.org/10.1145/133994.134088. https://doi.org/10.1145/133994.134088.

Gorisse, Geoffrey, Olivier Christmann, Etienne Amato, and Simon Richir. 2017. "First- and Third-Person Perspectives in Immersive Virtual Environments: Presence and Performance Analysis of Embodied Users." *Frontiers in Robotics and AI* 4 (July): 33. https://doi.org/10.3389/frobt.2017.00033.

Hart, Sandra G. 2006. "Nasa-Task Load Index (NASA-TLX); 20 Years Later." *Proceedings of the Human Factors and Ergonomics Society Annual Meeting* 50 (9): 904–908. https://doi.org/10.1177/154193120605000909. eprint: https://doi.org/10.1177/154193120605000909. https://doi.org/10.1177/154193120605000909.

Hart, Sandra G., and Lowell E. Staveland. 1988. “Development of NASA-TLX (Task Load Index): Results of Empirical and Theoretical Research.” In *Human Mental Workload,* edited by Peter A. Hancock and Najmedin Meshkati, 52:139–183. Advances in Psychology. North-Holland. https://doi.org/https://doi.org/10.1016/S0166-4115(08)62386-9. https://www.sciencedirect.com/science/article/pii/S0166411508623869.

Herrera, Fernanda, Soo Youn Oh, and Jeremy N. Bailenson. 2018. “Effect of Behavioral Realism on Social Interactions Inside Collaborative Virtual Environments.” *Presence: Teleoperators and Virtual Environments* 27, no. 2 (February): 163–182. https://doi.org/10.1162/pres_a_00324. eprint: https://direct.mit.edu/pvar/article-pdf/27/2/163/2003610/pres_a_00324.pdf. https://doi.org/10.1162/pres%5C_a%5C_00324.

Hettinger, Lawrence J., and Gary E. Riccio. 1992. “Visually Induced Motion Sickness in Virtual Environments.” *Presence: Teleoperators and Virtual Environments* 1, no. 3 (August): 306–310. https://doi.org/10.1162/pres.1992.1.3.306. eprint: https://direct.mit.edu/pvar/article-pdf/1/3/306/1622390/pres.1992.1.3.306.pdf. https://doi.org/10.1162/pres.1992.1.3.306.

Hoppe, Matthias, Andrea Baumann, Patrick Chofor Tamunjoh, Tonja-Katrin Machulla, Paweł W. Woźniak, Albrecht Schmidt, and Robin Welsch. 2022. “There Is No First- or Third-Person View in Virtual Reality: Understanding the Perspective Continuum.” In *Proceedings of the 2022 CHI Conference on Human Factors in Computing Systems.* CHI ’22. New Orleans, LA, USA: Association for Computing Machinery. ISBN: 9781450391573. https://doi.org/10.1145/3491102.3517447. https://doi.org/10.1145/3491102.3517447.

Inoue, Maakito, Kazuki Takashima, Kazuyuki Fujita, and Yoshifumi Kitamura. 2023. “BirdViewAR: Surroundings-aware Remote Drone Piloting Using an Augmented Third-person Perspective.” In *Proceedings of the 2023 CHI Conference on Human Factors in Computing Systems.* CHI ’23. Hamburg, Germany: Association for Computing Machinery. ISBN: 9781450394215. https://doi.org/10.1145/3544548.3580681. https://doi.org/10.1145/3544548.3580681.

Inoue, Yasuyuki, and Michiteru Kitazaki. 2021. “Virtual Mirror and Beyond: The Psychological Basis for Avatar Embodiment via a Mirror.” *Journal of Robotics and Mechatronics* 33 (October): 1004–1012. https://doi.org/10.20965/jrm.2021.p1004.

Jankowski, Jacek, and Martin Hachet. 2013. “A Survey of Interaction Techniques for Interactive 3D Environments.” In *Eurographics 2013 - STAR.* Girona, Spain, May. https://inria.hal.science/hal-00789413.

Keil, Jens, Michael Zollner, Mario Becker, Folker Wientapper, Timo Engelke, and Harald Wuest. 2011. “The House of Olbrich — An Augmented Reality tour through architectural history.” In *2011 IEEE International Symposium on Mixed and Augmented Reality - Arts, Media, and Humanities,* 15–18. https://doi.org/10.1109/ISMAR-AMH.2011.6093651.

Khan, Azam, Ben Komalo, Jos Stam, George Fitzmaurice, and Gordon Kurtenbach. 2005. "HoverCam: interactive 3D navigation for proximal object inspection." In *Proceedings of the 2005 Symposium on Interactive 3D Graphics and Games,* 73–80. I3D '05. Washington, District of Columbia: Association for Computing Machinery. ISBN: 1595930132. https://doi.org/10.1145/1053427.1053439. https://doi.org/10.1145/1053427.1053439.

Kilteni, Konstantina, Raphaela Groten, and Mel Slater. 2012. "The Sense of Embodiment in Virtual Reality." *Presence Teleoperators & Virtual Environments* 21 (November). https://doi.org/10.1162/PRES_a_00124.

Kim, Hyun K., Jaehyun Park, Yeongcheol Choi, and Mungyeong Choe. 2018. "Virtual reality sickness questionnaire (VRSQ): Motion sickness measurement index in a virtual reality environment." *Applied Ergonomics* 69:66–73. ISSN: 0003-6870. https://doi.org/https://doi.org/10.1016/j.apergo.2017.12.016. https://www.sciencedirect.com/science/article/pii/S000368701730282X.

Kim, Minju, Yuhyun Lee, and Jungjin Lee. 2022. "Multi-view Layout Design for VR Concert Experience." In *Proceedings of the 30th ACM International Conference on Multimedia,* 818–826. MM '22. Lisboa, Portugal: Association for Computing Machinery. ISBN: 9781450392037. https://doi.org/10.1145/3503161.3548347. https://doi.org/10.1145/3503161.3548347.

Kim, Woojoo, and Shuping Xiong. 2022. "ViewfinderVR: configurable viewfinder for selection of distant objects in VR." *Virtual Reality* 26, no. 4 (May): 1573–1592. ISSN: 1434-9957. https://doi.org/10.1007/s10055-022-00649-z. http://dx.doi.org/10.1007/s10055-022-00649-z.

Kusunoki, Mikiya, Ryo Furuhama, Ryusuke Toshima, Hazuki Mori, Haoran Xie, Tzu-Yang Wang, Takaya Yuizono, Toshiki Sato, and Kazunori Miyata. 2023. "MultiBrush: 3D Brush Painting Using Multiple Viewpoints in Virtual Reality." In *2023 9th International Conference on Virtual Reality (ICVR),* 481–486. https://doi.org/10.1109/ICVR57957.2023.10169798.

Latoschik, Marc Erich, and Carolin Wienrich. 2022. "Congruence and Plausibility, Not Presence: Pivotal Conditions for XR Experiences and Effects, a Novel Approach." *Frontiers in Virtual Reality* 3. ISSN: 2673-4192. https://doi.org/10.3389/frvir.2022.694433. https://www.frontiersin.org/journals/virtual-reality/articles/10.3389/frvir.2022.694433.

Lindlbauer, David, and Andy D. Wilson. 2018. "Remixed Reality: Manipulating Space and Time in Augmented Reality." In *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems,* 1–13. CHI '18. Montreal QC, Canada: Association for Computing Machinery. ISBN: 9781450356206. https://doi.org/10.1145/3173574.3173703. https://doi.org/10.1145/3173574.3173703.

McCrae, James, Igor Mordatch, Michael Glueck, and Azam Khan. 2009. "Multiscale 3D navigation." In *Proceedings of the 2009 Symposium on Interactive 3D Graphics and Games,* 7–14. I3D '09. Boston, Massachusetts: Association for Computing Machinery. ISBN: 9781605584294. https://doi.org/10.1145/1507149.1507151. https://doi.org/10.1145/1507149.1507151.

Narkar, Anish S., Jan J. Michalak, Candace E. Peacock, and Brendan David-John. 2024. "GazeIntent: Adapting Dwell-time Selection in VR Interaction with Real-time Intent Modeling." *Proc. ACM Hum.-Comput. Interact.* (New York, NY, USA) 8, no. ETRA (May). https://doi.org/10.1145/3655600. https://doi.org/10.1145/3655600.

Norman, Donald A. 2002. *The Design of Everyday Things.* USA: Basic Books, Inc. ISBN: 9780465067107.

Ortega, Michael, Wolfgang Stuerzlinger, and Doug Scheurich. 2015. "SHOCam: A 3D Orbiting Algorithm." In *Proceedings of the 28th Annual ACM Symposium on User Interface Software & Technology,* 119–128. UIST '15. Charlotte, NC, USA: Association for Computing Machinery. ISBN: 9781450337793. https://doi.org/10.1145/2807442.2807496. https://doi.org/10.1145/2807442.2807496.

Otono, Riku, Adélaïde Genay, Monica Perusquía-Hernández, Naoya Isoyama, Hideaki Uchiyama, Martin Hachet, Anatole Lécuyer, and Kiyoshi Kiyokawa. 2023. "I'm Transforming! Effects of Visual Transitions to Change of Avatar on the Sense of Embodiment in AR." In *2023 IEEE Conference Virtual Reality and 3D User Interfaces (VR),* 83–93. https://doi.org/10.1109/VR55154.2023.00024.

Pierce, Jeffrey S., Brian C. Stearns, and Randy Pausch. 1999. "Voodoo dolls: seamless interaction at multiple scales in virtual environments." In *Proceedings of the 1999 Symposium on Interactive 3D Graphics,* 141–145. I3D '99. Atlanta, Georgia, USA: Association for Computing Machinery. ISBN: 1581130821. https://doi.org/10.1145/300523.300540. https://doi.org/10.1145/300523.300540.

Pohl, Henning, Klemen Lilija, Jess McIntosh, and Kasper Hornbæk. 2021. "Poros: Configurable Proxies for Distant Interactions in VR." In *Proceedings of the 2021 CHI Conference on Human Factors in Computing Systems.* CHI '21. Yokohama, Japan: Association for Computing Machinery. ISBN: 9781450380966. https://doi.org/10.1145/3411764.3445685. https://doi.org/10.1145/3411764.3445685.

Pointecker, Fabian, Judith Friedl, Daniel Schwajda, Hans-Christian Jetter, and Christoph Anthes. 2022. "Bridging the Gap Across Realities: Visual Transitions Between Virtual and Augmented Reality." In *2022 IEEE International Symposium on Mixed and Augmented Reality (ISMAR),* 827–836. https://doi.org/10.1109/ISMAR55827.2022.00101.

Poupyrev, Ivan, Mark Billinghurst, Suzanne Weghorst, and Tadao Ichikawa. 1996. "The go-go interaction technique: non-linear mapping for direct manipulation in VR." In *Proceedings of the 9th Annual ACM Symposium on User Interface Software and Technology,* 79–80. UIST '96. Seattle, Washington, USA: Association for Computing Machinery. ISBN: 0897917987. https://doi.org/10.1145/237091.237102. https://doi.org/10.1145/237091.237102.

Praetorius, Anna Samira, and Daniel Görlich. 2020. "How Avatars Influence User Behavior: A Review on the Proteus Effect in Virtual Environments and Video Games." In *Proceedings of the 15th International Conference on the Foundations of Digital Games.* FDG '20. Bugibba, Malta: Association for Computing Machinery. ISBN: 9781450388078. https://doi.org/10.1145/3402942.3403019. https://doi.org/10.1145/3402942.3403019.

Prouzeau, Arnaud, Antoine Lhuillier, Barrett Ens, Daniel Weiskopf, and Tim Dwyer. 2019. "Visual Link Routing in Immersive Visualisations." In *Proceedings of the 2019 ACM International Conference on Interactive Surfaces and Spaces,* 241–253. ISS '19. Daejeon, Republic of Korea: Association for Computing Machinery. ISBN: 9781450368919. https://doi.org/10.1145/3343055.3359709. https://doi.org/10.1145/3343055.3359709.

Ryu, Jun, Seunghoon Park, and Gerard Jounghyun Kim. 2023. "Sickness Reduction in FPV Drone Control: Improved Effects of Reverse Optical Flow with Static Landmarks Only." In *Proceedings of the 29th ACM Symposium on Virtual Reality Software and Technology.* VRST '23. Christchurch, New Zealand: Association for Computing Machinery. ISBN: 9798400703287. https://doi.org/10.1145/3611659.3617219. https://doi.org/10.1145/3611659.3617219.

Schubert, Thomas, Frank Friedmann, and Holger Regenbrecht. 2001. "The Experience of Presence: Factor Analytic Insights." *Presence: Teleoper. Virtual Environ.* (Cambridge, MA, USA) 10, no. 3 (June): 266–281. ISSN: 1054-7460. https://doi.org/10.1162/105474601300343603. https://doi.org/10.1162/105474601300343603.

Shen, Chenxinran, Joanna Mcgrenere, and Dongwook Yoon. 2024. "LegacySphere: Facilitating Intergenerational Communication Through Perspective-Taking and Storytelling in Embodied VR." In *Proceedings of the 2024 CHI Conference on Human Factors in Computing Systems.* CHI '24. Honolulu, HI, USA: Association for Computing Machinery. ISBN: 9798400703300. https://doi.org/10.1145/3613904.3641923. https://doi.org/10.1145/3613904.3641923.

Tatzgern, Markus, Raphael Grasset, Eduardo Veas, Denis Kalkofen, Hartmut Seichter, and Dieter Schmalstieg. 2015. "Exploring real world points of interest: Design and evaluation of object-centric exploration techniques for augmented reality." *Pervasive and Mobile Computing* 18:55–70. ISSN: 1574-1192. https://doi.org/https://doi.org/10.1016/j.pmcj.2014.08.010. https://www.sciencedirect.com/science/article/pii/S1574119214001564.

Unruh, Fabian, Jean-Luc Lugrin, and Marc Erich Latoschik. 2024. "Out-Of-Virtual-Body Experiences: Virtual Disembodiment Effects on Time Perception in VR." In *Proceedings of the 30th ACM Symposium on Virtual Reality Software and Technology.* VRST '24. Trier, Germany: Association for Computing Machinery. ISBN: 9798400705359. https://doi.org/10.1145/3641825.3687717. https://doi.org/10.1145/3641825.3687717.

Wu, Fei, and Evan Suma Rosenberg. 2022. “Adaptive Field-of-view Restriction: Limiting Optical Flow to Mitigate Cybersickness in Virtual Reality.” In *Proceedings of the 28th ACM Symposium on Virtual Reality Software and Technology.* VRST ’22. Tsukuba, Japan: Association for Computing Machinery. ISBN: 9781450398893. https://doi.org/10.1145/3562939.3565611. https://doi.org/10.1145/3562939.3565611.

Zaman, Faisal, Craig Anslow, and Taehyun James Rhee. 2023. “Vicarious: Context-aware Viewpoints Selection for Mixed Reality Collaboration.” In *Proceedings of the 29th ACM Symposium on Virtual Reality Software and Technology.* VRST ’23. Christchurch, New Zealand: Association for Computing Machinery. ISBN: 9798400703287. https://doi.org/10.1145/3611659.3615709. https://doi.org/10.1145/3611659.3615709.

Zeleznik, Robert, and Andrew Forsberg. 1999. “UniCam—2D gestural camera controls for 3D environments.” In *Proceedings of the 1999 Symposium on Interactive 3D Graphics,* 169–173. I3D ’99. Atlanta, Georgia, USA: Association for Computing Machinery. ISBN: 1581130821. https://doi.org/10.1145/300523.300546. https://doi.org/10.1145/300523.300546.

Zhang, Di, Feng Xu, Chi-Man Pun, Yang Yang, Rushi Lan, Liejun Wang, Yujie Li, and Hao Gao. 2021. “Virtual Reality Aided High-Quality 3D Reconstruction by Remote Drones.” *ACM Trans. Internet Technol.* (New York, NY, USA) 22, no. 1 (September). ISSN: 1533-5399. https://doi.org/10.1145/3458930. https://doi-org.ezproxy.fh-salzburg.ac.at/10.1145/3458930.

```
[Serializable]
public class ViewConfig<T> where T : BaseViewPair
{
    public String panelTitle;

    public T prefab;

    [HideInInspector]
    public T instance = null;
}

public abstract class BaseViewModeHandler<T> : MonoBehaviour where T :
    BaseViewPair
{
    public List<ViewConfig<T>> viewConfigs;

    public int CurrentActiveViewCount
    {
        get
        {
            return viewConfigs.Count(x => x.instance != null);
        }
    }

    /// <summary>
    /// Deletes all active view pairs.
    /// </summary>
    /// <returns>The number of deleted view pairs.</returns>
    public int DeleteAllActiveViews()
    {
        var count = 0;
        viewConfigs.ForEach(x =>
        {
            if (x.instance != null)
            {
                x.instance.DeleteViewPair();
                x.instance = null;
                count++;
            }
        });
        return count;
    }

    public abstract T SpawnViewPair();

    public abstract void Activate();

    public virtual void Deactivate()
    {
        DeleteAllActiveViews();
    }
}
```

Listing 4: The BaseViewModeHandler class provides a blueprint for ViewModeHandlers, supporting flexible additions of new View-Modes.

```
    public class DroneViewPair : BaseViewPair
    {
        public DroneCamController droneCamController;

        public override void Awake()
        {
            base.Awake();
            droneCamController.viewPair = this;
        }

        public override void ReceiveSelect()
        {
            droneCamController.IsSelected = !droneCamController.IsSelected;
        }
    }
```

Listing 5: The DroneViewPair class extends BaseViewPair with additional functionality, including a DroneCamController.

```
...
    public float moveSpeed = 25f;
    public float pitchSpeed = 75f;
    public float yawSpeed = 75f;
    public float heightSpeed = 10f;
...
```

Listing 6: The DroneCamController defines movement speed fields for translational and rotational motion.

```
[Serializable]
public class DroneActions
{
    public InputActionReference moveAction;
    public InputActionReference yawPitchAction;
    public InputActionReference upAction;
    public InputActionReference downAction;

    public void EnableAllActions()
    {
        moveAction.action.Enable();
        yawPitchAction.action.Enable();
        upAction.action.Enable();
        downAction.action.Enable();
    }
    public void DisableAllActions()
    {
        moveAction.action.Disable();
        yawPitchAction.action.Disable();
        upAction.action.Disable();
        downAction.action.Disable();
    }
}
```

Listing 7: The DroneActions class manages input actions for controlling the drone's movement and orientation.

```
    /// <summary>
    /// Attempts to locate the LocomotionSystem within the scene.
    /// </summary>
    /// <returns>True if the locomotion system was found; otherwise, false
       .</returns>
    private bool TryToFindLocomotion()
    {
        if (_locomotionSystem != null) return true;
        _locomotionSystem = FindObjectOfType<LocomotionSystem>();
        _locomotionControllerManagers = FindObjectsOfType<
           ActionBasedControllerManager>().ToList();
        return _locomotionSystem != null;
    }

    public void SetLocomotionEnabled(bool enabled)
    {
        if (TryToFindLocomotion())
        {
            _locomotionSystem.gameObject.SetActive(enabled);
            foreach (var c in _locomotionControllerManagers)
            {
                c.enabled = enabled;
            }
        }
    }
```

Listing 8: The LocomotionSystem component is located and toggled to control user movement. The script searches for the component and sets its GameObject's active state.

Appendices

A AI Methodology

During the writing of this thesis ChatGPT has been used as a tool for paraphrasing and structuring of text and in the search for scientific sources. Exemplary prompts that have been used are for example:

```
what is a good structure for a user study in a master thesis
```

or for getting feedback on first drafts of paragraphs which gave feedback on structure and wording, the following prompt was given to ChatGPT:

```
what do you say to this introduction written in latex?
\section{User Study}
In order to answer the research question what method is more effective in a
     museum setting a user study was conducted. The user study includes a
    task where users have to find \ac{POI}s in a diorama. The users are
    given this task with both methods in order to compare them. Data is
    collected during the task to quantify this.
\\
The study is followed by a questionnaire and an interview in order to also
    answer the questions on whether the multiple viewpoints lead to better
    exploration or greater simulator sickness. In addition the questionnaire
     and interview are used to evaluate the embodiment of the users and the
    existence of a Perspective Continuum \autocite{PerspectiveContinuum
    :2022} in this setting.
\\
The hypothesise hereby is that the HoverCam \autocite{HoverCam:2005} \ac{
    OCE} approach leads to faster exploration due to the easier usage and
    quicker change of look-at-targets via a simple ray cast. Furthermore it
    is expected that multiple viewpoints lead to greater simulator sickness
    since users have the additional task of managing said viewpoints whilst
    also leading to better exploration since the extra viewpoints give a
    greater overview of the environment. Finally, a Perspective Continuum is
     expected to exist, however due to the constant switching between
    viewpoints during the exploration a similar switching between embodiment
     is expected as well.
```

As a last example, ChatGPT was used as aid in the search for scientific papers, this prompt was given to the AI:

```
i am looking to find a scientific paper on control schemes for virtual
    drones in VR. im mostly interested in whether the controls use the users
     forward direction or the drones
```

ChatGPT responded with useful keywords to search for.

B git-Repository

The source code of this thesis can be found on the following git repository:
https://gitlab.mediacube.at/fhs44512/mmt-masterarbeit-david-maerzendorfer

This work has the following word count (counted by texcount):

```
File: body.tex
Encoding: utf8
Sum count: 17225
Words in text: 16285
Words in headers: 133
Words outside text (captions, etc.): 807
Number of headers: 55
Number of floats/tables/figures: 30
Number of math inlines: 0
Number of math displayed: 0
Subcounts:
  text+headers+captions (#headers/#floats/#inlines/#displayed)
  552+1+4 (1/0/0/0) Section: Introduction
  126+2+0 (1/0/0/0) Subsection: Object-Centric Exploration
  349+2+0 (1/0/0/0) Subsection: Free Exploration
  482+2+4 (1/0/0/0) Subsection: Museum Setting
  100+2+0 (1/0/0/0) Subsection: Research Questions
  42+2+0 (1/0/0/0) Section: Related Work
  481+1+0 (1/0/0/0) Subsection: Viewpoints} \label{RelatedWorkViewpoints
  438+2+0 (1/0/0/0) Subsection: Object-Centric Exploration
  447+2+0 (1/0/0/0) Subsection: Distant Interaction
  899+1+54 (1/2/0/0) Subsection: Embodiment
  501+1+0 (1/0/0/0) Subsection: Drones
  387+2+0 (1/0/0/0) Section: System Design
  282+3+63 (1/1/0/0) Subsection: OCE Approach: HoverCam
  222+3+55 (1/1/0/0) Subsection: Free Drone Approach
  183+1+66 (1/1/0/0) Subsection: View-Panels
  116+1+4 (1/0/0/0) Section: Implementation
  224+1+0 (1/0/0/0) Subsection: View-Pair
  213+1+0 (1/0/0/0) Subsection: View-Camera
  300+1+0 (1/0/0/0) Subsection: View-Panel
  697+7+0 (3/0/0/0) Subsection: View-Manager
  635+7+0 (3/0/0/0) Subsection: Drone Implementation} \label{droneImpleme
  1671+11+167 (5/6/0/0) Subsection: HoverCam Implementation} \label{hover
  614+1+19 (1/0/0/0) Subsection: Avatar}\label{Avatar
  609+3+0 (1/0/0/0) Subsection: Point of Interest}\label{sec:POI
  333+1+28 (1/1/0/0) Subsection: Control-Manager}\label{sec:controlManage
  439+17+8 (6/0/0/0) Subsection: Study-Manager}\label{sec:studyManager
  1346+2+111 (1/6/0/0) Section: User Study
  537+1+159 (1/10/0/0) Section: Results
  21+3+0 (1/0/0/0) Section: Findings and Discussion
  823+4+0 (1/0/0/0) Subsection: Research Questions and Hypotheses
```

```
175+4+17 (1/1/0/0) Subsection: POI Search Time Distribution
1626+37+48 (10/1/0/0) Subsection: Participant Feedback and Possible Sys
415+4+0 (1/0/0/0) Section: Conclusion and Future Work
```