LOOK & RST: GAZE-BASED OBJECT MANIPULATION IN VR

PRATIK MOHANTY, 202303457

MASTER'S THESIS

June 2025

Advisor: Ken Pfeuffer & Uta Wagner

AARHUS
UNIVERSITY
DEPARTMENT OF COMPUTER SCIENCE

LOOK & RST: GAZE-BASED OBJECT MANIPULATION IN VR

Master's thesis

Department of Computer Science
Faculty of Natural Sciences
Aarhus University

June 2025

ABSTRACT

This thesis presents an in-depth investigation into gaze-based multimodal interaction techniques for object manipulation—specifically Rotation, Scaling, and Translation (RST)—within immersive Virtual Reality (VR) environments. Motivated by the limitations of conventional hand-based VR interaction, such as physical fatigue and mental demand, this research explores the integration of eye tracking with hand gestures to enable more intuitive and sustainable control. Building upon previous gaze-based methods like *Look&Drop*[21] and *Gaze + Pinch*[17], this study introduces an application that supports all three RST operations in both 2D and 3D contexts. Three interaction techniques—*Hand Ray*, *Gaze + Pinch*, and *Look&Drop*—were implemented in a custom VR application and evaluated through a comprehensive user study with 18 participants across progressive manipulation tasks.

The study assesses usability and user experience using NASA-TLX workload assessments, user preference rankings, and qualitative feedback analysis. Statistical analysis revealed significant differences in mental demand, effort, and frustration across techniques, with *Gaze + Pinch* emerging as the optimal approach, preferred by approximately 70% of participants. Results demonstrate that the complementary pairing of gaze-based selection with hand-based manipulation achieves superior performance across all evaluation measures, effectively addressing the limitations of single-input methods while creating an interaction paradigm that reduces cognitive overhead. The findings establish that gaze-based multimodal approaches are well-suited for RST operations in VR environments, offering a promising alternative for prolonged spatial design work. The work contributes to the field of Human-Computer Interaction by providing evidence-based insights into the design of ergonomic, intuitive interaction systems in VR.

We have seen that computer programming is an art,
because it applies accumulated knowledge to the world,
because it requires skill and ingenuity, and especially
because it produces objects of beauty.

— Donald E. Knuth [11]

ACKNOWLEDGMENTS

I would like to express my sincere gratitude to my supervisor, Associate Professor Ken Pfeuffer, for his invaluable guidance, insightful advice, and unwavering support throughout this thesis. I am also deeply thankful to Dr. Uta Wagner for her fruitful contributions, which greatly enhanced the quality of this work.

My appreciation extends to the entire EH team for generously sharing their expertise in eye-hand research, fostering an enriching academic environment. I am equally grateful to my office mates, friends, and family, whose encouragement and motivation have been instrumental in completing this journey.

Pratik Mohanty
Aarhus, June 2025

CONTENTS

LIST OF FIGURES

LIST OF TABLES

LISTINGS

ACRONYMS

API Application Programming Interface

HCI Human-Computer Interaction

VR Virtual Reality

FOV Field Of View

HMD head-mounted display

RST Rotation, Scaling, and Translation

1 INTRODUCTION

Virtual Reality (VR) has emerged as a transformative technology in Human-Computer Interaction (HCI), as users now have immersive experiences that influence how we work with digital information [1]. As technology becomes better and more accessible, finding user-friendly, comfortable, and efficient ways to control digital devices is more important than before [19]. Users engage effectively in virtual environments mainly because they can control the Rotation, Scaling, and Translation (RST) of virtual objects.

As VR technology advances rapidly, interacting with objects inside VR continues to be a challenge. Typically, VR interaction is based on hand movements and waving of the hand in the air, which can make your hands feel tired and drop accuracy for detailed activities [7]. Due to the "gorilla arm syndrome" issue, which significantly limits the practical applications of VR, particularly in professional contexts where sustained interaction is necessary [15].

Gaze-based interaction offers a strong solution by relying on our natural instinct to look before we begin an activity with an item [17]. With the help of eye tracking in VR, it offers a reduction in how much we have to move our bodies while still maintaining or improving how accurately we select and control objects. Including the use of gaze along with minimal gestures could lead to VR becoming more usable by using faster and convenient eye movements for normal actions and gestures for specific task purposes. Using technology in this way might allow users to rotate, scale, or translate objects effortlessly, as it reduces the strain linked to regular ways of interacting with the object in a virtual environment.

This research explores **Gaze-Based interaction to perform rotation, scaling, and translation (RST) operations** on virtual objects in immersive environments. Building upon the foundational work of Wagner et al.[21] and Pfeuffer et al.[17], which previously showed how gaze and hand can be used together to move objects in a 3D environment, we extend these possibilities to include rotation and scaling manipulations. Although *Look&Drop* and *Gaze + Pinch* were limited to moving and positioning objects, our approach enables users to perform complete RST operations in 2D and 3D manipulation tasks within immersive environments. A main focus of the study is to see if users can perform these complex manipulations simply by looking at a virtual item with their eyes and using hand gestures. We aim to investigate whether the multimodal technique can enhance all three manipulation tasks(Rotation, Scaling, and Translation). Our approach

extends existing gaze-based selection techniques to enable comprehensive RST operations through two distinct multimodal techniques: *Gaze + Pinch* and *Look&Drop*. *Gaze + Pinch* combines eye gaze for target identification with hand gestures for manipulation control—users gaze at an object to select it, then perform a pinch gesture while moving their hand to control rotation, scaling, or translation. In contrast, *Look&Drop* relies primarily on eye movements with minimal hand input, where users direct their gaze toward an object and use only a pinch gesture (without hand movement) to trigger and control the manipulation through continued eye tracking. However, this extension presents a fundamental attention dilemma: users must simultaneously direct their gaze toward the object they wish to manipulate while also visually monitoring the transformation results to provide accurate feedback. This challenge is particularly pronounced during continuous operations like rotation and scaling, where users need to observe the object's changing state while maintaining gaze-based control. Despite this inherent complexity, evaluating the complete RST spectrum with gaze-based interaction represents a significant contribution that has not been systematically explored in previous research, making this investigation both scientifically interesting and practically relevant for advancing multimodal VR interaction design.

To guide this investigation, the study addresses the following research questions.

1. **How can Multimodal interaction be optimally integrated to support the full spectrum of rotation, scaling, and translation (RST) tasks in virtual environments?**

2. **In what ways do multimodal gaze and hand-based manipulation techniques complement each other within the practical context of spatial design applications?**

3. **What are the usability implications and performance outcomes when implementing gaze-enhanced interaction techniques in spatial design tools compared to conventional methods?**

To explore these research questions, this study uses an application that supports object manipulation tasks in both 2D and 3D spatial design contexts. While gaze-based input may not offer the same level of precision as traditional hand-based techniques, the goal is to investigate how gaze, when combined with minimal gestures, can support efficient, low-effort interaction. This setup allows us to examine multimodal input in scenarios that require careful placement and continuous engagement, helping us understand not only performance but also how users experience comfort, intuitiveness, and overall usability during complex manipulation tasks.

With the help of this, the application offers an interactive VR environment designed to support object manipulation and creative tasks,

inspired by the simplicity and flexibility of PowerPoint. Users can place and arrange shapes, customize visual elements, and interact with objects using intuitive gestures. The tool set enables efficient layout, annotation, and editing workflows in both 2D and 3D contexts. A detailed description of the individual features is provided in section 3.5.

From the user study, we found that the users responded very positively to the application, especially appreciating how it showcased the full potential of combining RST (Rotate, Scale, Translate) with different interaction techniques and creative tools. After the tests, overall, users found the Gaze + Pinch method to be the most intuitive and easy to use for object manipulation. When looking at individual tasks, for the translation task, both of the gaze-based techniques were highly preferred(approximately 90% of the users)for it's fast selection and moving an object. While in the scaling task, preferences were split evenly between Gaze + Pinch and Hand Ray. Many users noted that *Look&Drop* felt mentally demanding for scaling, and they had to put a lot of effort, as it required them to focus on the handle without giving them clear control over how the object looked during the adjustment. In contrast, for the full RST task, Gaze + Pinch stood out as the top choice with approx 60%. These results reflect a strong interest of the users in gaze-based interaction, especially for more complex manipulation tasks.

These findings highlight the immense potential of gaze-based interactions in shaping the future of XR applications. By offering an intuitive and natural way to interact with virtual environments, gaze-based techniques can redefine how users engage with digital spaces. Combining gaze with gestures, like pinch, opens up opportunities to create more immersive, efficient, and accessible tools for creative and complex tasks, setting a new standard for user-centric design in XR.

2 RELATED WORK

This chapter positions our research within the landscape of VR interaction techniques and contextualizes our contributions by exploring relevant prior work that has shaped the design of our technique and our evaluation methodology. We structure this discussion around essential research domains that have impacted our strategy for gaze-based object manipulation in VR settings.

2.1 MULTIMODAL INTERACTION APPROACHES

Eye tracking has emerged as a promising approach to address the physical limitations of hand-based interaction. Early work by Zhai, Morimoto, and Ihde [26] introduced the MAGIC (Manual And Gaze Input Cascaded) pointing technique, which used gaze to reposition the cursor near the area of visual attention, reducing the physical distance for manual refinement. This principle of combining gaze for coarse positioning with manual input for refinement has informed numerous subsequent interaction techniques.

Based on this idea, Pfeuffer et al. [17] extended this concept to VR environments with their "*Gaze + Pinch*" technique, using eye gaze for targeting and a simple pinch gesture for selection and manipulation. From the evaluation, we learned that this hand-eye system uses human eyes well while keeping the hand from moving too much, though certain limitations with small or distant targets remained.

Continuing on this, research has increasingly focused on combining gaze with complementary input modalities to enhance interaction capabilities. Chatterjee, Xiao, and Harrison [6] created the "Gaze + Gesture" technique employing both eye and hand tracking, where a pinch gesture and subsequent hand movement function as a cursor. Their evaluation found no significant performance difference between this technique and traditional mouse input, despite the accuracy limitations of eye tracking.

Expanding on this multimodal perspective,Jeong et al.'s GazeHand system [10] explores different combinations of eye tracking and hand gestures to optimize various manipulation tasks. Their evaluation reveals that different tasks benefit from different interaction combinations, suggesting the importance of context-appropriate technique selection.

Similarly,Turner et al. [20] explored the combination of gaze and hand input for cross-device interactions, finding that this multimodal

approach reduces physical effort while maintaining or improving task performance.

Our research builds on these multimodal approaches by evaluating the *Look & Drop* and the other technique across varied manipulation tasks, with particular attention to how the combination of gaze and minimal hand input affects physical comfort during extended use and usability.

2.2 3D OBJECT MANIPULATION: ROTATION, SCALING, AND TRANSLATION (RST)

The manipulation of virtual objects through RST operations represents a core functionality in VR environments and is the primary focus of our research. Prior work in this area has explored various input modalities and interaction techniques to optimize these fundamental operations.

Building on this, Pathmanathan et al. [16] conducted a comparative study between eye gaze and head gaze for RST operations in VR. Their findings demonstrated that while head-gaze offers more stability in certain contexts, eye-gaze provides superior speed and reduced physical effort, qualities particularly valuable for manipulation tasks.

Expanding on gaze-based interaction, Liu, Orlosky, and Plopski [13] introduced multiple gaze-driven rotation methods (RotBar, RotPlane, RotBall) for head-mounted display (HMD), optimizing single-axis and multiaxis rotations. Their work complements head-based manipulation techniques (e.g., head rotation mapping) and hybrid approaches combining gaze/head inputs for hands-free interaction. RotBall notably improved multiaxis efficiency, while RotBar/RotPlane enhanced precision for simpler rotations.

In a related direction, Liu, Plopski, and Orlosky[14] provides an innovative approach using gaze to control movement along orthogonal planes. Their application interprets gaze direction to determine the plane of movement, simplifying 3D positioning tasks while maintaining user control.

Further integrating gaze with other inputs, Yu et al. [25] investigated the integration of gaze and hand input for 3D object manipulation in VR. Their study explored coordination and transition strategies between gaze and manual input, revealing that combining these modalities can enhance user performance and reduce physical strain during complex manipulation tasks.

Beyond gaze-based approaches, finger-based manipulation techniques have been explored. For example, in their research on the FingerOscillation method, Wu, Chellali, and Otmane [24] examined different types of finger gestures for manipulation tasks, discovering

that even small variations in gesture design have a significant impact on both performance and comfort during prolonged VR sessions.

Extending the exploration of hybrid techniques, Wagner et al. [21] introduced the "*Look & Drop*" technique, which integrates precise gaze targeting with minimal hand movement for object placement and manipulation. The key benefit of *Look&Drop* is that gaze takes over the xy movement of an object while the hand is only responsible for the z movement. While primarily focused on translation, their approach demonstrates the potential for extending gaze-based interaction to rotation and scaling operations.

Our research extends these investigations by comprehensively evaluating multimodal techniques across all three RST operations with a self-developed 2D and 3D tool inspired by PowerPoint.

2.3 PHYSICAL CONSIDERATIONS IN VR INTERACTION

A fundamental challenge in VR interaction is the physical strain associated with mid-air interactions. Hincapié-Ramos et al. [7] introduced "consumed endurance" as a quantitative metric for arm fatigue during mid-air gestures, establishing that continuous hand tracking induces significant physical strain. This phenomenon, often termed "gorilla arm syndrome", represents a critical barrier to extended use of VR systems.

Jang et al. [9] developed a model to quantify cumulative arm muscle fatigue during mid-air interactions, considering both perceived exertion and arm motion kinetics. Their findings highlight the necessity of designing VR interactions that account for users' physical limitations to prevent fatigue accumulation over time.

When we discuss physical fatigue, it directly helps our case because the *Look & Drop* technique addresses arm fatigue by letting users select targets with their gaze. Since we leave spatial control to the eyes, whose motions cause the least strain, we try to handle the main hurdle faced during standard manipulation tasks.

2.4 EVALUATION METHODOLOGIES

Argelaguet and Andujar [2] carried out a thorough review of techniques for selecting 3D objects within virtual settings, providing detailed discussions on factors influencing selection effectiveness. Their characterization of user fatigue as "one of the most known issues in virtual reality applications" directly informs our evaluation approach, which prioritizes reduced physical effort alongside task performance.

Bergström et al. [3] offered guidelines for evaluating object selection and manipulation techniques in VR, addressing the challenge of comparing techniques between studies. Their recommendations for study design and reporting protocols have informed our evaluation methodology, particularly in our approach to task design and performance measurement.

Building on established evaluation frameworks, we also draw on Buxton's taxonomy of interaction techniques [4], which provides a systematic approach to categorizing and comparing different interaction modalities. This taxonomy helps us position our evaluation within the broader context of 3D interaction research and ensures comprehensive coverage of relevant interaction dimensions.

We use these guidelines as a base, concentrating on how well the gaze-based technique handles physical ease and workflow efficiency in a range of tasks. Using standard evaluation measures, we plan to prove the usefulness of our approach compared to other methods in the field.

2.5 RESEARCH GAPS AND OUR CONTRIBUTION

While prior research highlights the potential of gaze-based interaction in VR, opportunities remain for broader evaluations across a wider range of object manipulation tasks. In particular, further exploration is needed to understand how these techniques perform across specific RST operations and in more realistic usage scenarios.

Our research addresses these gaps by:

1. Systematically evaluating all techniques across RST operations. Comparison of the effectiveness of different interaction techniques.

2. Focusing specifically on the balance between physical comfort and ease.

3. Investigating how the technique performs in both 2D and 3D contexts.

This comprehensive evaluation aims to determine whether gaze-based techniques can offer a practical alternative to hand-based methods for object manipulation applications requiring extended use.

3 DESIGN CHOICES

This chapter explains the key elements and methods behind the following techniques: *Hand Ray*, *Gaze + Pinch*, and *Look&Drop*. Each technique describes a different way to handle object manipulation in virtual reality.

3.1 DESIGN PRINCIPLES

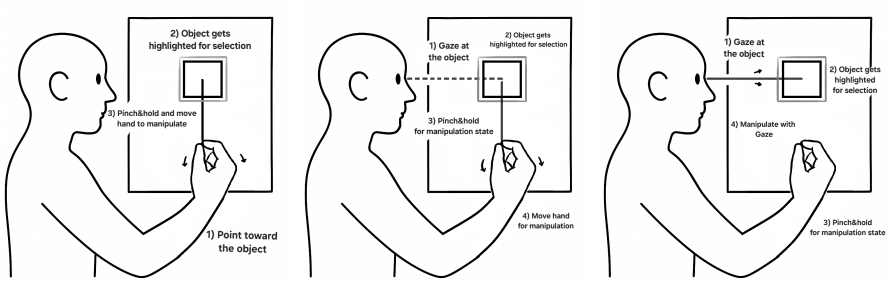
It's important to consider several aspects when designing ways to interact in VR. Among the most important aspects influencing users and tasks are visibility and responsiveness:

1. **Physical Comfort**: As highlighted in the related work section, VR interactions often induce significant physical strain during extended use. Our design choices prioritize techniques that minimize the "gorilla arm syndrome" identified by Hincapié-Ramos et al. [8].
2. **Intuitiveness**: Effective VR interactions allow users to make sense of them using what they already know and can do by just looking at the application's interface.
3. **Consistency**: Each technique must support all fundamental RST operations (Rotation, Scaling, Translation) across both 2D and 3D manipulation.
4. **Visual Feedback**: Clear visual indicators for selection, highlighting, and manipulation state are essential for user orientation and task completion.

With these considerations in mind, we used three distinct interaction techniques, each representing a different balance of gaze and hand input for object manipulation.

3.2 INTERACTION TECHNIQUES

We chose the interaction methods based on this goal since we wanted to explore the differences in integration and control of movements in VR. They let us examine how users perform RST tasks with their hands, eyes or both(see Figure 3.1), providing a detailed understanding of how different interaction methods affect task performance.



(a) *Hand Ray* **(b)** *Gaze + Pinch* **(c)** *Look&Drop*

Figure 3.1: Overview of the Interaction Techniques for object selection and manipulation

3.2.1 *Hand Ray*

The *Hand Ray* technique represents the conventional approach to manipulating an object in VR environments. This technique uses a ray directed from the user's hand to interact with virtual objects (see Figure 3.1a). It gives a direct spatial mapping between hand movement and object manipulation, connecting with a ray, which is intuitive for many users. As it requires the ray from the palm, we need to keep our hand extended in mid-air, which can lead to physical fatigue during extended use, as noted by Jang et al. [9].

3.2.2 *Gaze + Pinch*

The *Gaze + Pinch* technique by Pfeuffer et al.[18], combines gaze with pinch for selection and manipulation of the target object (see Figure 3.1b). This approach uses an adequate amount of hand movement while controlling how things are manipulated with the help of your hand. With this technique, you will be using your eyes as a cursor for preselection, and the hand confirms the preselection with a pinch gesture. After that, the hand manipulates the object in XYZ. However, users still need to do some hand movement to adjust the object in both 3d and 2d space.

3.2.3 *Look&Drop*

The *Look&Drop* technique, inspired by the work of Wagner et al. [21], represents the most gaze-centric approach explored in this work. In enables users to perform most interface tasks using only their eyes, requiring minimal hand input for occasional confirmations (see Figure 3.1c). The novel aspect lies in its division of task control: gaze is used to navigate along the X and Y axes, while the hand controls movement along the Z axis. For tasks demanding precise control in 3D space, however, the hand takes over full manipulation across all three

axes. Compared to other techniques, *Look&Drop* significantly reduces physical effort by relying primarily on eye movements combined with a simple pinch gesture.

3.3 RST BASED ON INTERACTION TECHNIQUES

This section examines how the chosen interaction techniques support core tasks such as selection, translation, rotation, and scaling in 2D and 3D, as shown in Table 3.1 and Table 3.2.

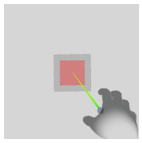
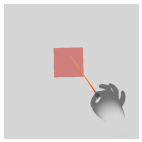
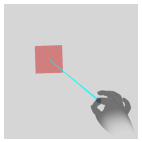
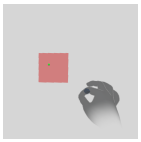
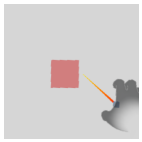
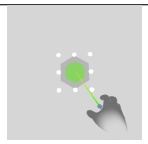
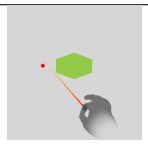
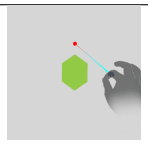
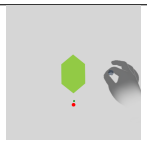
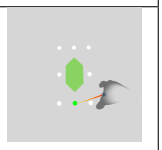
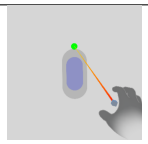
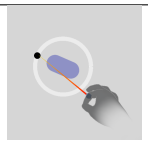
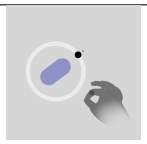
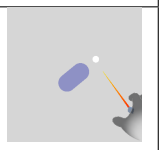
	Selection	***Hand Ray***	***Gaze + Pinch***	***Look&Drop***	Release
Translation					
Scaling					
Rotation					

Table 3.1: Visual breakdown of interaction stages of 2D object across the three techniques. The first column shows the selection state required before manipulation begins and the end state for task completion. This is followed by translation, scaling, and rotation implementations using *Hand Ray*, *Gaze + Pinch*, and *Look&Drop*.

3.3.1 *Hand Ray*

Selection Mechanism: Users point a ray from their palm toward the target object. When the ray intersects an object, it enters a hover state, visually highlighting the object to indicate it is selectable. To confirm the selection, users perform a pinch gesture, which activates the transformation handles. These handles can then be individually selected using the same point-and-pinch method.

Translation: Once an object is selected, it is translated by aligning its position with the endpoint of the *Hand Ray*. In 2D contexts, objects move along the plane they reside on (X and Y axes). In 3D environments, planar translation behaves identically, however, depth manipulation along the Z-axis is achieved by moving the hand for-

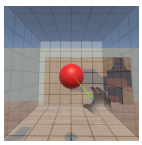
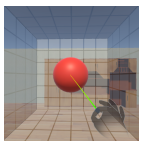
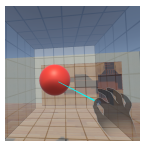
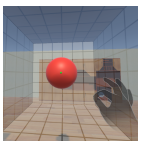
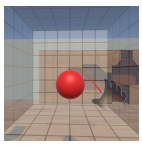
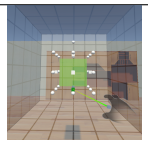
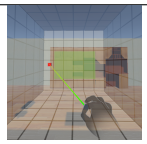
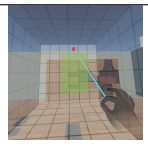
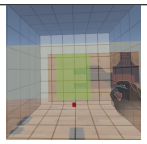
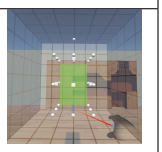
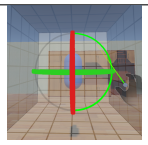
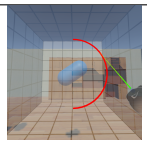
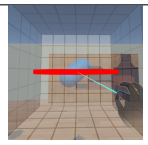
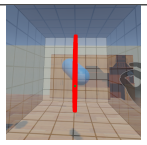
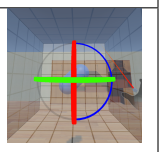
	Selection	*Hand Ray*	*Gaze + Pinch*	*Look&Drop*	Release
Translation					
Scaling					
Rotation					

Table 3.2: Illustration of 3D object manipulation workflows using the three interaction techniques. Each manipulation task is preceded by a distinct selection state and end state, highlighting the procedural nature of interaction.

ward or backward. The object continuously updates its position to match the 3D endpoint of the ray, enabling intuitive spatial control across all dimensions.

Rotation: When an object is selected, axis-specific rotation handles are displayed around it. Users direct the *Hand Ray* at a desired rotation handle and engage a pinch-and-drag gesture to rotate the object around its corresponding axis. The rotation angle is dynamically computed based on the handle's angular displacement, allowing for precise and continuous alignment of the object's orientation with the handle's position.

Scaling: Similar to rotation, axis-aligned handles appear at the edges of the selected object, allowing users to modify its dimensions with precision. By targeting a handle using the *Hand Ray* and performing a pinch-and-drag gesture, users can resize the object along the corresponding axis. Moving the handle away from the object's centre increases its size, while dragging it inward reduces it. This interaction supports intuitive, axis-specific dimension adjustment.

3.3.2 *Gaze + Pinch*

Selection Mechanism: Functionally similar to *Hand Ray*, *Gaze + Pinch* leverages gaze for target acquisition combined with a close-open pinch gesture to confirm selection.

Translation: Once an object is selected, its position is controlled relative to the hand's movement from the initial pinch point. Hor-

izontal and vertical translation (X and Y axes) corresponds to the displacement of the hand from the pinch origin, visually reinforced by a line connecting the hand and the object. For 3D, Z-axis (depth) manipulation requires moving the hand forward and backward from the initial position.

Rotation: Rotation handles are targeted through gaze and engaged by a pinch-and-hold gesture. The angle of rotation depends on the hand movement relative to the object's center, with the gaze determining which axis or handle is being manipulated in both 2D and 3D.

Scaling: Scaling is performed by first selecting the object and then targeting scaling handles with gaze, followed by a pinch gesture to engage. The scaling magnitude depends on the hand's movement as the hand move away (enlarging) or toward (reducing) the initial pinch point, the object rescales in the direction the handle is selected.

3.3.3 *Look&Drop*

Selection Mechanism: Similar to *Gaze + Pinch*, users look at objects or handles to highlight them, followed by a pinch gesture for selection. This maintains the natural eye-hand coordination pattern where the user looks at objects before interacting with them.

Translation: After selection, the object follows the user's gaze direction in the X and Y axes —the object moves to where the user is looking within the interaction plane while holding the pinch. For movement in 3D environments, micro-movements of the fingertips (subtle changes in pinch width) control depth. As you move your pinched finger back and forth, the depth will adjust accordingly.

Rotation: For 2D rotation, interaction mirrors that of the *Hand Ray* technique. In 3D, the user selects one of three rotation handles by gaze and aligns their view along the chosen handle's axis to initiate rotation. Fine adjustments can be made using minimal finger movements during the pinch.

Scaling: Scaling operations use gaze distance from the object's initial handle position to determine magnitude. Looking further from the initial handle position increases the size, while looking closer decreases it. For 3D, micro-movements of the pinched fingers provide control over the scaling in the Z-axis.

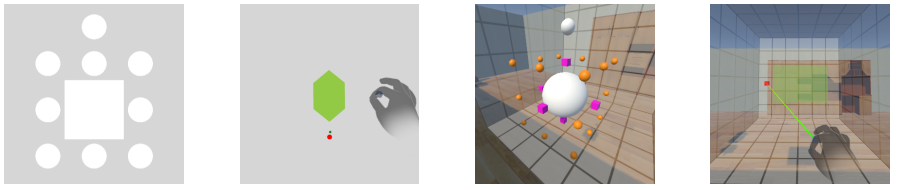
3.4 VISUAL FEEDBACK

Consistent visual feedback mechanisms were implemented across all three interaction techniques to support user awareness and improve interaction performance. These mechanisms help users quickly under-

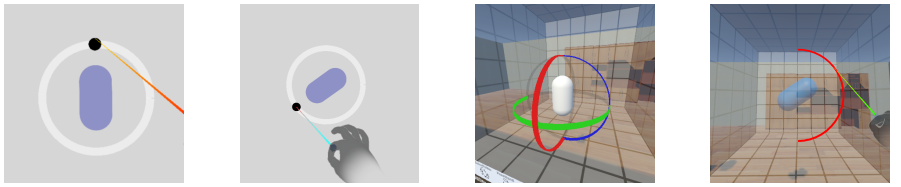
stand the state of the system and receive immediate responses to their actions:

(a) Target Highlight (b) Tool Highlight (c) Handle Highlight

Figure 3.2: Object highlighting when a user points a ray or gazes at an object/a tool/a handle to get highlighted or change of color



(a) 2D Selection (b) 2D Scaling State (c) 3D Selection (d) 3D Scaling State



(e) 2D Rotation Selection (f) 2D Rotaion State (g) 3D Rotation Selection (h) 3D Rotation State

Figure 3.3: Object selection and manipulation states, where pinching on an object reveals handles for scaling or rotating using the top circular handle.

1. **Object Highlighting**: Interactable objects are visually emphasized when targeted via ray or gaze. A subtle mask appears beneath the object, manipulation handles turn green, and UI icons are outlined with a highlight to indicate the current pointing direction as shown in Figure 3.2.

2. **Selection and Manipulation State**: The selection state for the object will let the handle appear, which shows the object is selected.Once the object is selected, the standard handles appear for rotation and scaling operations, with consistent coloring (e.g., red for x-axis, green for y-axis, blue for z-axis) as shown in Figure 3.3. Upon selecting, they turn red in 3D, and for 2D, the

Rot handle turns black and a white circular part appears. Once released, the handles return to their original colors, such that the user can reuse them.

3. **Gaze Indicator**: In gaze-based techniques, a subtle marker shows the user's current gaze point. The indicator dynamically changes color depending on the surface it intersects and turns green upon a successful pinch gesture, confirming selection.

4. **Ray Indicator**: In hand-based methods, a red ray visualizes the pointing direction. When the ray intersects with a target, it turns green to indicate a successful hit.

3.5 USER INTERFACE COMPONENTS

To provide a versatile interaction environment, we developed a unified interface that supports a wide range of features and tasks, including object creation, manipulation, and exploration. Our goal was to create a consistent foundation for both the implementation and demonstration of the three interaction techniques in a realistic and application-oriented setting. Inspired by familiar applications such as Microsoft PowerPoint, we integrated several tool panels offering essential functionality, as shown in Figure 3.4. These panels were designed to be accessible through all three interaction techniques while ensuring a consistent visual appearance and interaction behavior.

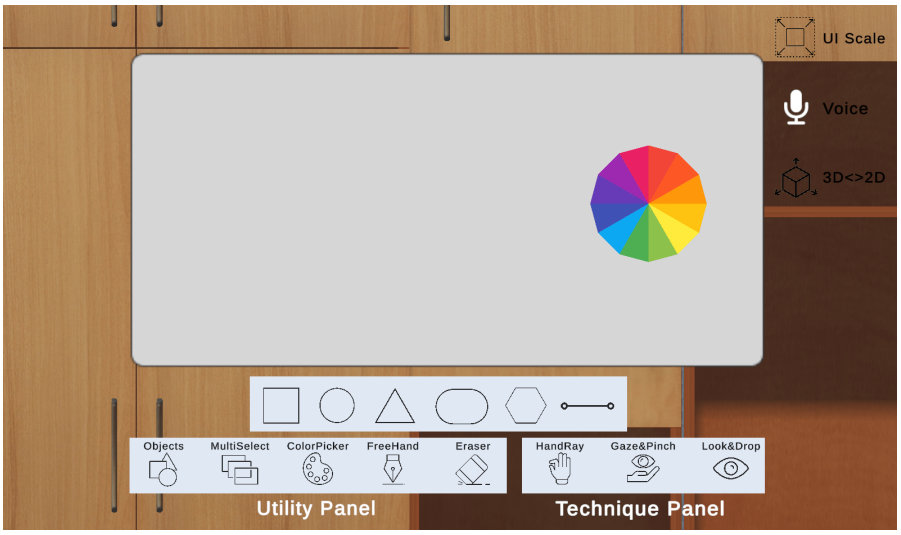


Figure 3.4: User Interface Overview with Utility Panel Tools (Objects, Multi-select, Color Picker, Freehand, Erase), Technique Selection Panel, and Side Panel Features

3.5.1 Utility Panel

The **Utility Panel** provides core tools essential for the application and is strategically positioned within the user's field of view to ensure easy access. Visual highlighting provides immediate feedback when a user points at a button, helping them stay oriented and supporting efficient interaction.

1. **Objects Tool**: This tool presents a popup menu featuring basic 2D and 3D geometric shapes (e.g., cubes, spheres, cylinders) as well as line connectors. Users can select a shape from the menu and instantiate it on the canvas by performing a pinch-and-drag gesture. Once placed, the object can be manipulated using the currently active interaction technique, as described in Section 3.2

2. **Multi-select Tool**: This tool enables users to select and manipulate multiple objects at once by drawing a selection mask over a specific area on the canvas. When the mode is activated, users can drag to define a selection region that highlights all objects within its bounds. Selected objects can then be translated as a group using any of the three interaction techniques. In 3D mode, a volumetric selection mask is created instead of a 2D region, allowing grouped manipulation of the enclosed objects.

3. **Color Picker**: Upon selecting an object, users can activate the circular color palette to modify its appearance of the selected object. The palette offers a range of standard colors, and visual feedback immediately previews the chosen color on the object. A pinch gesture applies the selected color.

4. **Freehand Tool**: This tool allows users to create custom shapes by placing a sequence of connected points in space. As each point is defined, the system automatically draws edges between consecutive points. The completed shape becomes a manipulable object, supporting the same transformations as predefined shapes.

5. **Erase Tool**: This tool provides functionality for removing objects from the scene. When activated, users can select individual objects for deletion, with immediate visual feedback confirming their removal.

3.5.2 Technique Panel

The **Technique Panel** allows users to switch between the three implemented interaction modes described in Section 3.2:

1. ***Hand Ray***: Activates the *Hand Ray* technique, using ray casting from the palm for selection and manipulation.

2. ***Gaze + Pinch***: Switches to a hybrid technique that combines eye gaze for targeting, pinch gestures for selection, and hand movement for manipulation.

3. ***Look&Drop***: Enables the most gaze-driven techniques, requiring minimal hand movement by relying primarily on eye gaze input.

The currently active technique is visually highlighted in the panel to provide clear feedback. By default, the system starts in *Hand Ray* mode. Users can switch techniques by pointing at the corresponding icon in the panel and performing a pinch gesture, ensuring consistency with the overall interaction design.

3.5.3 Side Panel

The **Side Panel** contains additional utility functions that enhance the overall interaction experience:

1. **UI Scale**: Enables users to adjust the overall size of the interface by performing a pinch-and-drag gesture on the scaling icon. his feature enhances accessibility by allowing users to customize the UI according to their preferences and interaction distance, ensuring comfortable use across different physical setups.

2. **Voice Annotation**: Allows users to attach textual labels to selected objects using voice input. When activated, the system captures voice input and converts it into on-screen text attached directly to the object, supporting quick and hands-free annotations.

3. **Perspective Toggle (2D-3D)**: Enables users to switch between 2D and 3D interaction perspective. In 2D mode, object movement is constrained to a single plane, facilitating precise layout tasks. In 3D mode, objects can be manipulated with full 6DOF. All interaction techniques dynamically adapt to the selected dimensional context.

All interface elements were designed with consistent visual styling and interaction behaviors to minimise cognitive overload and ensure a smooth user experience when switching between different tools and techniques. Selection highlights, activation feedback, and tool transitions adhere to a unified visual language across the system.

The specific details about the implementation of these tasks and the evaluation methodology are detailed in the following chapter 4 and later.

4 IMPLEMENTATION

This chapter elaborates on the practical application of the three interaction techniques introduced in Chapter 3. Each method was implemented to ensure compatibility and functionality across the respective interaction styles.

4.1 SOFTWARE

The application developed for this user case study was implemented using Unity version 60000.0.43f1 on a Windows 11 system. The core functionality was programmed in C#, leveraging Unity's scripting API. To enable multi-modal interaction, we integrated the Meta XR All-in-One SDK package, which supports eye tracking, hand tracking, and voice command recognition through the deployed HMD. However, our implementation only accessed eye tracking data, voice data, and hand pose detection via the OVR Application Programming Interface (API). The remaining functionality utilized Unity's built-in packages: `Vector3` and `Quaternion` classes handled spatial transformations, `Mathf` performed mathematical operations, `System.Collections` and `System.Collections.Generic` managed data structures, `System.Linq` facilitated collection manipulation, and `System.IO` with `System.Threading` enabled file operations and asynchronous processing. This approach allowed us to create a comprehensive VR application while maintaining compatibility with Unity's native ecosystem.

4.2 HARDWARE

The work was created and tested on the Meta Quest Pro headset, a device that has integrated eye tracking and hand tracking. With a 106° × 95.57° Field Of View (FOV), a per-eye resolution of 1800 × 1920 pixels at a 90Hz refresh rate, and offers six degrees of freedom tracking through five built-in cameras. Like eye tracking, hand tracking is done with the built-in camera system, while dedicated eye-tracking sensors are placed by the lenses.

The eye tracking on the Meta Quest Pro is shown by Wei, Bloemers, and Rovira[22] to have a median accuracy of roughly 1.635° and a median precision of 0.702° (measured using standard deviation). These requirements became the basis for our approach to eye-controlled interaction.

4.3 EYE TRACKING

Using reliable eye-tracking interactions presented several challenges. Minor shifts in the headset position can cause slippage issues that affect eye-tracking quality. Furthermore, involuntary eye movements-particularly high-velocity saccades and low-amplitude fixational jitter-introduced significant noise into the gaze signal, complicating object selection and precise targeting in both 2D and 3D spaces.

To mitigate these issues, we used several solutions. First, we applied temporal smoothing to the raw gaze data using the 1 € Filter with parameters $fcmin = 0.5$ and $\beta = 1$, as recommended by Casiez, Roussel, and Vogel[5]. Second, we enhanced gaze detection by replacing standard ray casting with CircleCast and SphereCast for 2D and 3D object detection. This approach emit a volumetric probe- circular in 2D and spherical in 3D- from the eye position in the gaze direction, making selection easier as it covers a larger surface compared to simple ray casting. The radius was set to approximately 0.1° visual angle to balance sensitivity and precision, minimizing false positives. Finally, we implemented visual feedback by adding a subtle glow to objects intersected by the gaze cast. This immediate highlighting response supported user awareness and gaze-based targeting.

4.4 HAND TRACKING

Hand tracking implementation faced its own set of challenges related to spatial accuracy, temporal stability, and robustness under real-world conditions. Ray instability is one of the primary issues, as the pointing vector derived from hand orientation is highly susceptible to minor, unintentional hand movements. These micro movements can cause endpoint jitter, impairing precise selection of small targets or distant UI elements. Occlusion presents another challenge, especially during complex gestures like pinching. When thumb and index finger occlude each other or fall outside the field of view, leading to unreliable gesture recognition or false negative. The Heisenberg effect creates additional complications, as while performing a pinch gesture, there is a possibility that the ray will move away from the desired object since the hand also moves during the pinch gesture, moving the ray from the object [23]. Furthermore, selection precision becomes challenging when targeting small objects or manipulation handles with a ray from the hand, especially for users with hand tremors or when working at a distance.

To address these issues, we use the same filtering method as the eye tracking. Here we applied the 1 € Filter to hand position data with parameters set at $fcmin = 1.5$ and $\beta = 1$, which were determined through testing to provide optimal smoothing without introducing

noticeable latency. To minimize the Heisenberg effect [23], we selected specific tracking points on the hand, specifically the point between the convergence point of the thumb and index finger, as reference points for hand-based manipulations. Similarly to the eye-tracking approach, to enhance detection by replacing raycasting with SphereCast and CircleCast for 2D and 3D interactions, a spherical detection volume is used. This improved selection reliability, especially for small handles and distant objects, while still allowing for positioning once selected.

4.5 TECHNICAL ASPECTS OF INTERACTION TECHNIQUES

This section is included to provide a deeper understanding of the underlying technical implementation of each interaction technique mentioned in Section 3.2, highlighting how raw eye and hand tracking data are processed and stabilized to ensure smooth and reliable user interactions. This also serves as a technical reference.

Look&Drop Implementation

Understanding the implementation of eye tracking for *Look&Drop*, which is crucial to demonstrate how our system achieves reliable eye-based selection despite the inherent noise in the eye tracking data. It captures individual eye positions and orientations, then combines them using spherical linear interpolation for optimal accuracy:

```
Vector3 leftEyePos = LeyeGaze.transform.position;
Vector3 rightEyePos = ReyeGaze.transform.position;
Quaternion leftEyeRot = LeyeGaze.transform.rotation;
Quaternion rightEyeRot = ReyeGaze.transform.rotation;

Vector3 rayOrigin = Vector3.Lerp(leftEyePos, rightEyePos, eyeWeightBias);
Quaternion weightedRotation = Quaternion.Slerp(leftEyeRot, rightEyeRot,
    eyeWeightBias);
Vector3 rawRayDirection = weightedRotation * Vector3.forward;
```

Listing 4.1: Weighted Gaze Ray Calculation

The combination process uses `Vector3.Lerp()` for position interpolation and `Quaternion.Slerp()` for rotation blending. The `eyeWeightBias` parameter allows for dynamic weighting between the dominant and non-dominant eye, accommodating individual user characteristics and improving tracking stability.

The filtering mechanism is to address the jittery noise in gaze signals. Using the One Euro Filter provides frequency-adaptive smoothing that maintains responsiveness while eliminating jitter:

```
InitializeFilters()
```

```
    {
        positionFilters = new OneEuroFilter[3];
        directionFilters = new OneEuroFilter[3];

        for (int i = 0; i < 3; i++)
        {
            positionFilters[i] = new OneEuroFilter(filterMinCutoff,
                filterBeta, filterDCutoff);
            directionFilters[i] = new OneEuroFilter(filterMinCutoff,
                filterBeta, filterDCutoff);
        }
    }

filteredRayOrigin = new Vector3(
        positionFilters[0].Filter(rayOrigin.x, currentTime),
        positionFilters[1].Filter(rayOrigin.y, currentTime),
        positionFilters[2].Filter(rayOrigin.z, currentTime)
    );

filteredRayDirection = new Vector3(
        directionFilters[0].Filter(rawRayDirection.x, currentTime),
        directionFilters[1].Filter(rawRayDirection.y, currentTime),
        directionFilters[2].Filter(rawRayDirection.z, currentTime)
    ).normalized;
```

Listing 4.2: Initialization and using One Euro Filter for jitter

The filtering process operates independently of each spatial dimension, allowing for anisotropic noise characteristics. Position and direction vectors receive separate filter treatment, with direction vectors requiring normalization post-filtering to maintain unit vector properties essential for ray calculations.

Gaze + Pinch Implementation

Documenting the implementation is *Gaze + Pinch* essential to showcase how we seamlessly integrate eye gaze selection with hand-based manipulation, creating a hybrid approach that leverages the strengths of both modalities. The technical implementation is similar to the *Look&Drop* approach for selection while maintaining separate processing pathways for manipulation using the position of the hand.

```
// Calculate hand movement delta
Vector3 handMovementDelta = palm.position - initialPalmPosition;

// Apply movement multiplier to make an object move faster than the hand
handMovementDelta *= handMovementMultiplier;

// Create a movement delta that only affects X and Y axes (keeps Z
    constant)
Vector3 adjustedMovement = new Vector3(
    handMovementDelta.x,
    handMovementDelta.y,
    0  // No movement on Z axis
```

```
);
```

Listing 4.3: Hand posistion calculation for manipulation

Hand Ray Implementation

Hand-based ray tracking provides direct mapping between hand orientation and interaction ray direction. The technical implementation uses the palm transform as the primary reference frame:

```
Vector3 rayOrigin = palmTransform.position;
Vector3 rawRayDirection = palmTransform.forward;

filteredRayDirection = new Vector3(
    directionFilters[0].Filter(rawRayDirection.x, currentTime),
    directionFilters[1].Filter(rawRayDirection.y, currentTime),
    directionFilters[2].Filter(rawRayDirection.z, currentTime)
).normalized;
```

Listing 4.4: Filtering jitter of HandRay Using 1 Euro Filter

The hand-tracking approach requires only directional filtering, as the position of the hand shows lower noise characteristics compared to eye tracking. The palm's forward vector provides intuitive ray direction mapping, creating natural pointing behavior that aligns with user expectations.

4.6 TECHNICAL ASPECTS OF RST

This section gives a detailed overview of how the proposed interaction techniques are applied to enable rotation, scaling, and translation (RST) of objects in both 2D and 3D, providing insight into the technical design choices behind each manipulation mode. It is also intended as a reference for future developers or researchers who wish to understand, replicate, or extend the RST implementation in similar XR systems.

4.6.1 Rotation

The rotation mechanism illustrated in Figure 4.1 details how user input is translated into smooth and intuitive object rotation in both 2D and 3D contexts. Rotation is computed using vector mathematics and coordinate space transformations, tailored to the dimensional context of the interaction. For 3D, the system employs quaternion-based or axis angle transformations to ensure smooth manipulation. In 2D context, rotation is applied around the object's local Z-axis within screen space.

There are two main approaches to 2D rotation: hand position mapping and ray vector projection. In hand position mapping, the object responds directly to the user's hand movement—when the user moves their hand left or right, the object rotates in that direction, almost as if they're physically grabbing and turning it. In contrast, ray vector projection works more like pointing with a laser. The user aims a ray at the canvas, and the object rotates to the point where the ray hits while holding the pinch. This method gives a more indirect way to control orientation by simply pointing to where the object should face.

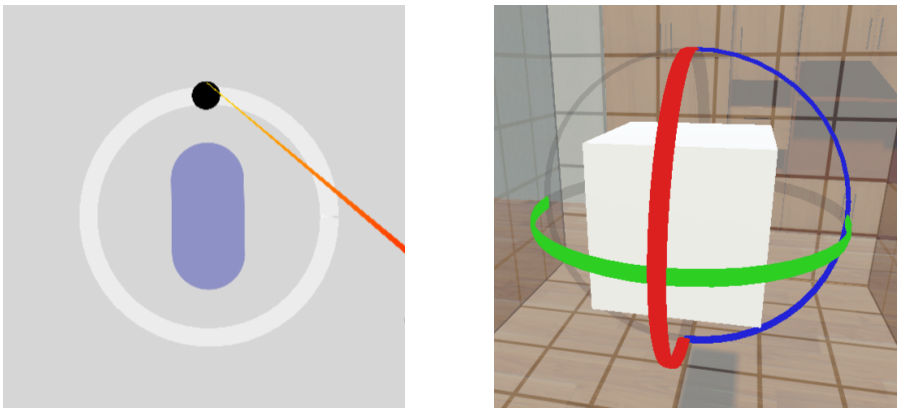


Figure 4.1: Sample images showing the rotation of 2D and 3D objects. The left image demonstrates 2D rotation with an object constrained to planar movement. The right image shows 3D rotation of an object using a tri-axial coordinate system, where red, green, and blue handles represent rotation around the X, Y, and Z axes, respectively.

For the *Gaze + Pinch*, as it relies on the eyes for selection and the position of the hand for manipulating the object, for which it uses hand position-based rotation, here the application measures changes in position by comparing the current palm placements with the first hand locations.

```
1 Vector2 currentHandPosition = new Vector2(palm.position.x, palm.position.
      y);
2 Vector2 initialHandPosition = new Vector2(initialPalmPosition.x,
      initialPalmPosition.y);
3 Vector2 handMovementDelta = currentHandPosition - initialHandPosition;
4 float rotationAmount = -handMovementDelta.x * rotMovementMultiplier;
```

Listing 4.5: Hand movement converted to 2D rotation delta

To rotate the object, the user just has to move their hand in a horizontal direction after pinching on the Rot Handle. If the user moves their hand sideways, the system measures the movement from the initial point where the user initiated the initial pinch and rotates the object according to the change in distance. The negative sign ensures that rightward hand movement produces clockwise rotation, creating intuitive control. The Rot handle's position is constrained to follow a

circular path, which will keep the Rot handle in a fixed place around the targeted object, by using parametric circle equations:

```
Vector2 newPosition = new Vector2(
    rotationCenter.x + Mathf.Cos(Mathf.Deg2Rad * (initialAngle +
        rotationAmount)) * rotationRadius,
    rotationCenter.y + Mathf.Sin(Mathf.Deg2Rad * (initialAngle +
        rotationAmount)) * rotationRadius
);
rotObject.transform.position = new Vector3(
            newPosition.x,
            newPosition.y,
            rotObject.transform.position.z
        );

\\ Updating the rotation of the selected object
Quaternion targetRotation = initialObjectRotation * Quaternion.Euler(0,
    0, rotationAmount);
parentTransform.rotation = targetRotation;
```

Listing 4.6: Rot handle circular orbit using parametric circle equations

This calculation maintains the Rot handle at a fixed distance from the rotation center while moving it along the circular path. The rotation of the selected object happens by updating the handle's angular position on the circular path and the parent transform's rotation together, creating a synchronized rotation of the selected object and the Rot handle in the orbit.

However, the interaction techniques that rely on raycasting(*Hand Ray* & *Look&Drop*) use the ray-based rotation method by implementing vector projection techniques by calculating the ray's endpoint and projecting it onto the plane:

```
Vector3 currentRayEndPoint = rayOrigin + (rayDirection * rayDistance);
Vector3 currentRayEndPointFlat = Vector3.ProjectOnPlane(
    currentRayEndPoint - rotationCenter, Vector3.forward).normalized;
Vector3 initialRayEndPointFlat = Vector3.ProjectOnPlane(
    initialRayEndPoint - rotationCenter, Vector3.forward).normalized;
float rotationAngleDelta = Vector3.SignedAngle(initialRayEndPointFlat,
    currentRayEndPointFlat, Vector3.forward);
```

Listing 4.7: Ray-based rotation using projected 2D vectors and signed angle delta

The rotation is done by comparing the angular difference between the initial ray direction and the current ray direction. The `ProjectOnPlane()` function flattens the 3D vectors onto the 2D plane, allowing for accurate angle calculation. The `SignedAngle()` function determines both the magnitude and direction of rotation needed, with the rotation being applied incrementally based on this angular delta.

The 3D rotation occurs by isolating movement to specific planes. For X-axis rotation, the system projects both start and end vectors onto the YZ plane by zeroing out the X component, effectively creating

2D vectors that represent the rotation around the X-axis. The angle between these projected vectors determines the rotation amount. The `Quaternion.AngleAxis()` creates a rotation quaternion around the specified axis, which is then combined with the current rotation to produce the final orientation.

The system addresses quaternion interpolation challenges through a custom smoothing function `SmoothDampQuaternion()`:

```
Vector3 currentEuler = current.eulerAngles;
Vector3 targetEuler = target.eulerAngles;

for (int i = 0; i < 3; i++)
{
    while (targetEuler[i] - currentEuler[i] > 180f) targetEuler[i] -= 360
        f;
    while (targetEuler[i] - currentEuler[i] < -180f) targetEuler[i] +=
        360f;
}

Vector3 result = new Vector3(
    Mathf.SmoothDamp(currentEuler.x, targetEuler.x, ref angVelocity.x,
        smoothTime),
    Mathf.SmoothDamp(currentEuler.y, targetEuler.y, ref angVelocity.y,
        smoothTime),
    Mathf.SmoothDamp(currentEuler.z, targetEuler.z, ref angVelocity.z,
        smoothTime)
);
```

Listing 4.8: Custom quaternion smooth damping using Euler angles and per-axis smoothing

The smooth rotation occurs by converting quaternions to Euler angles, normalizing the angular differences to prevent jumps when crossing 360 degrees, and applying smooth damping to each axis independently. This ensures that rotations appear fluid and natural rather than snapping abruptly between orientations. The implementation incorporates safety mechanisms, including movement threshold validation using `Vector3.Distance(pinchStartPoint, pinchEndPoint) > 0.01f` to prevent erratic rotation from minimal hand movements, ensuring stable and predictable rotation behavior throughout the interaction.

Scaling

The section details how objects are scaled, focusing on translating hand and ray movements into size adjustments. The system ensures stable and visually clear interactions. With this, users can use coordinate space transformations and movement detection for comprehensive 2D and 3D object resizing control as shown in Figure 4.2. We used two primary approaches for scaling: through raycasting and hand position

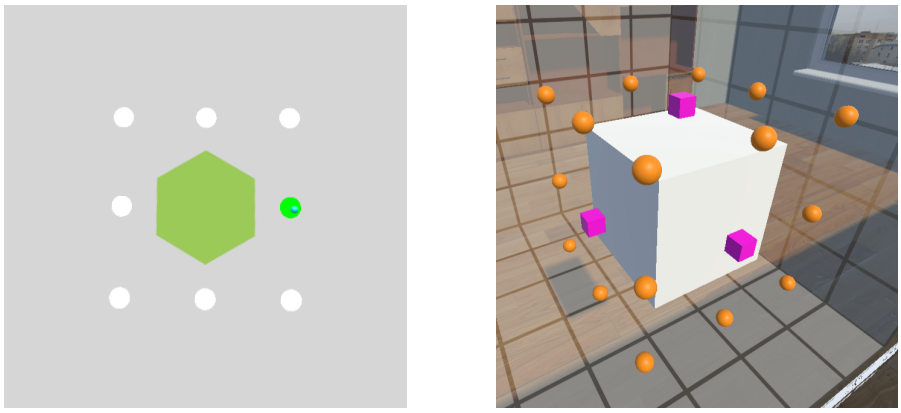


Figure 4.2: These images show the initial state of the Object before scaling the 2D and 3D Object:

mapping, with each solution using mathematical functions for scale adjustments.

For 2D scaling operations, the system utilizes raycasting-based movement detection combined with local coordinate space transformations. The core scaling logic begins with movement threshold validation to prevent unwanted scaling from minimal hand movements:

```
Vector3 pinchDelta = raycastOrigin - lastRayPosition;
// Threshold check to prevent jittery scaling
if (pinchDelta.magnitude < pinchThreshold)
    return;
pinchDelta *= scaleFactor;
```

Listing 4.9: Threshold filtering to avoid jitter in scaling

The movement delta is converted from world space to the parent object's local coordinate system, enabling scale calculations that remain consistent regardless of the object's orientation or position in the scene:

```
Vector3 localPinchDelta = parentTransform.InverseTransformDirection(
    pinchDelta);
Vector3 currentScale = parentTransform.localScale;
float scaleChangeX = 0, scaleChangeY = 0;
Vector3 localPositionOffset = Vector3.zero;
```

Listing 4.10: Transforming world-space delta to local space for consistent scaling

Scale modifications are applied based on the specific handle being manipulated, with each handle type corresponding to different scaling behaviors (uniform, X-axis only, Y-axis only, or corner scaling). The system enforces minimum scale limits to prevent objects from becoming invisible or inverted:

```
ApplyScalingBasedOnTag(hitObject, localPinchDelta, ref scaleChangeX, ref
    scaleChangeY, ref localPositionOffset);
currentScale.x = Mathf.Max(currentScale.x + scaleChangeX, 0.1f);
```

```
currentScale.y = Mathf.Max(currentScale.y + scaleChangeY, 0.1f);
```

Listing 4.11: Applying local scale changes with minimum size limits

Position compensation is applied to maintain the scaling origin at the appropriate handle location, converting local space offsets back to world space coordinates:

```
Vector3 worldPositionOffset = parentTransform.TransformDirection(
    localPositionOffset);
currentPosition += worldPositionOffset;
```

Listing 4.12: Converting local offset back to world space to maintain handle origin

The 3D scaling implementation extends the 2D approach by incorporating depth control and maintaining proper scale relationships across all three axes. For hand-based scaling, the system combines ray endpoint movement with palm position changes for comprehensive 3D control:

```
Vector3 rayDelta = rayEndPoint - initialRayEndPoint;
rayDelta *= 0.05f; // Reduce ray sensitivity for the control
float zDelta = (palm.position.z - initialPalmPosition.z) *
    handMovementMultiplier * 0.05f;
Vector3 movementDelta = new Vector3(rayDelta.x, rayDelta.y, zDelta);
```

Listing 4.13: 3D scaling based on palm transformed to local space

The movement data are transformed into the parent object's local coordinate system, enabling consistent scaling behavior regardless of object orientation:

```
Vector3 localDelta = parentTransform.InverseTransformDirection(
    movementDelta);
float scaleChangeX = 0, scaleChangeY = 0, scaleChangeZ = 0;
Vector3 localPositionOffset = Vector3.zero;
Apply3DScalingBasedOnTag(hitObject, localDelta, ref scaleChangeX, ref
    scaleChangeY, ref scaleChangeZ, ref localPositionOffset);
```

Listing 4.14: Smooth application of 3D scale with per-axis clamping

Scale changes are applied with enhanced smoothing for more controlled transitions, while maintaining minimum scale thresholds across all axes:

```
Vector3 currentScale = parentTransform.localScale;
currentScale.x = Mathf.Max(currentScale.x + scaleChangeX,
    minScaleThreshold);
currentScale.y = Mathf.Max(currentScale.y + scaleChangeY,
    minScaleThreshold);
currentScale.z = Mathf.Max(currentScale.z + scaleChangeZ,
    minScaleThreshold);

float enhancedSmoothFactor = smoothFactor * 3.0f;
```

```
parentTransform.localScale = Vector3.SmoothDamp(
    parentTransform.localScale,
    currentScale,
    ref scaleVelocity,
    enhancedSmoothFactor);
```

Listing 4.15: Enhance scale smoothing such that it will follow the Ray

For gaze-based scaling interactions, the system maintains similar mathematical principles while adapting the input source to use palm position for movement detection:

```
...
Vector3 handMovementDelta = palm.position - initialPalmPosition;
if (handMovementDelta.magnitude < handMovementThreshold)
    return;
handMovementDelta *= AdjusthandMovementMultiplier;
Vector3 localPinchDelta = parentTransform.InverseTransformDirection(
    handMovementDelta);
```

Listing 4.16: Gaze-based palm movement converted to local space for scaling

The scaling system incorporates child object protection by storing and restoring world-scale relationships, ensuring that nested objects maintain their relative proportions during parent scaling operations. This is achieved through the "StoreChildrenWorldScales()" and "RestoreChildrenWorldScales()" functions, which preserve the visual appearance of complex hierarchical objects during manipulation.

Visual feedback is provided through color interpolation and line rendering, giving users a clear indication of active scaling operations and the relationship between hand position and object state throughout the interaction.

4.6.2 Translation

Translation allows for precise positioning control using various interaction techniques. The 2D translation system relies on raycast-based object detection and multi-modal interaction handling. The core detection mechanism uses "Physics2D.CircleCast()" to identify target objects within the interaction radius:

```
Vector3 endPoint = rayOrigin + (rayDirection * rayDistance);
float hoverOffset = 0.01f;
RaycastHit2D hit2D = Physics2D.CircleCast(endPoint, hoverOffset, Vector2.
    zero, 0, targetLayer);
```

Listing 4.17: 2D object detection using CircleCast at ray endpoint

Translation occurs through different modal approaches. In the Eye-Gaze and HandRay modes, translation occurs by directly positioning the ray endpoint. When a target is selected, the system calculates the

movement deltas by comparing the current ray endpoint with the initial hit point.

```
Vector3 movementDelta = endPoint - initialHitPoint1;
targetPosition = initialTargetPosition + movementDelta;
selectedTarget.transform.position = Vector3.Lerp(
    selectedTarget.transform.position,
    targetPosition,
    Time.deltaTime * smoothSpeed
);
```

Listing 4.18: Translation based on Techniques based on RayCasting

The translation is achieved through direct coordinate mapping, where the ray endpoint movement directly corresponds to object displacement. The "Vector3.Lerp()" function provides smooth interpolation between the current position and the target position, preventing abrupt movements and creating natural motion trajectories.

The *Gaze + Pinch* technique implements a more sophisticated translation approach using hand position tracking with amplification control:

```
Vector3 handMovementDelta = palm.position - initialPalmPosition;
handMovementDelta *= handMovementMultiplier;
Vector3 adjustedMovement = new Vector3(
    handMovementDelta.x,
    handMovementDelta.y,
    0  // No movement on Z axis
);
targetPosition = initialTargetPosition + adjustedMovement;
```

Listing 4.19: GazePinch hand movement converted to constrained 2D object translation

This technique translates hand movement into object displacement with configurable amplification through the "handMovementMultiplier". The Z-axis constraint ensures 2D planar movement while the multiplier allows for control based on user preferences. The system maintains visual feedback through line renderers connecting the hand to the manipulated object.

The 3D scaling system employs a handle-based approach where specific tagged objects serve as control points for parent object manipulation. The system uses "Physics.SphereCast()" for more generous hit detection in 3D space. Scaling operations begin with handle identification and parent relationship establishment. The system stores initial states, including object positions, scales, and child object configurations. Then the scaling calculation varies by interaction mode, with *Gaze + Pinch* mode using direct translation of hand movement:

```
Vector3 handMovementDelta = palm.position - initialPalmPosition;
handMovementDelta *= AdjusthandMovementMultiplier;
Vector3 localPinchDelta = parentTransform.InverseTransformDirection(
    handMovementDelta);
```

Listing 4.20: Scaling with Gaze+Pinch

The system converts world-space hand movements to local object space using "InverseTransformDirection()", allowing for consistent scaling regardless of object orientation. The "Apply3DScalingBasedOnTag()" function determines which axes to scale based on the handle's tag, enabling axis-specific or uniform scaling operations.

For *Look&Drop* mode, the system combines ray endpoint movement with finger position tracking for multi-dimensional control:

```
Vector3 rayDelta = rayEndPoint - initialRayEndPoint;
float zDelta = (indexTip.position.z - initialIndexTipZ) *
    LNDMovementMultiplier;
Vector3 movementDelta = new Vector3(rayDelta.x, rayDelta.y, zDelta);
```

Listing 4.21: Index Poisition Shift for Scaling with Look & Drop in Z axis

This approach separates XY control through gaze tracking from Z-axis control through finger positioning, providing intuitive 3D manipulation. The scaling application uses "Vector3.SmoothDamp()" for gradual transitions:

```
parentTransform.localScale = Vector3.SmoothDamp(
    parentTransform.localScale,
    currentScale,
    ref scaleVelocity,
    smoothFactor);
```

Listing 4.22: Parent scaling based on the techniques

The system maintains child object consistency through inverse scaling operations via "RestoreChildrenWorldScales()", ensuring that child objects maintain their world-space dimensions while the parent scales. Safety mechanisms include minimum scale thresholds using "Mathf.Max(currentScale.x + scaleChangeX, minScaleThreshold)" to prevent objects from becoming invisible or inverted. Visual feedback is provided through color interpolation and line renderers, giving users a clear indication of active manipulation states and connection visualization between interaction points and target objects.

These implementation details ensured that all three interaction techniques could be evaluated fairly, with system performance and visual feedback remaining consistent across techniques.

5 EVALUATION

This study examined how users interact with, utilize, and experience multimodal manipulation techniques. We evaluated task completion across users with varying VR experience levels, collecting primarily qualitative feedback to understand users' thoughts, feelings, and experiences while using the application and performing manipulations. The purpose of the experiment was to introduce users to intuitive object handling through different interaction techniques while demonstrating the method's suitability for simple and satisfying creative work.

5.1 RATIONALE

To understand how users truly experience interaction techniques, it is necessary to capture the nuanced human responses that emerge during actual use. We chose a qualitative hands-on user study because surveys and controlled experiments each have fundamental limitations for this type of research, as established by prior work on the HCI evaluation methodology. Surveys can tell us what people think they prefer in theory, but preferences stated in isolation often do not match real-world usage patterns. Controlled studies excel at measuring precise performance metrics—completion times, error rates, accuracy—but these numbers miss the nuanced human experience that determines whether someone will want to use a system. For instance, while prior work on *Look&Drop* and *Gaze + Pinch* demonstrated effectiveness through controlled studies measuring task completion times and accuracy, our preliminary observations suggested that this technique might not extend well to complex real-world RST (rotation, scaling, translation) tasks that require simultaneous multi-modal interactions. This gap between controlled performance and realistic usability aligns with broader concerns in HCI about evaluation methods that move "beyond usability" to capture the full user experience[12]. Moreover, Bergström et al. [3]'s comprehensive review of 20 years of VR evaluation studies highlights the ongoing challenges in establishing appropriate evaluation methods for VR interaction techniques, noting that "standards developed for evaluating 2D interaction often do not apply" to VR contexts.

We needed to understand not just whether users could complete tasks, but how the techniques felt to use, what frustrated them, what delighted them, and what made them confident or uncertain. This

experiential focus follows established HCI evaluation practices that recognize the importance of qualitative insights in understanding user interaction. Both quantitative and qualitative research traditions are essential to understand people and interactional contexts, and toolkit research in HCI benefits from evaluation strategies that capture both performance and user experience. This qualitative approach allowed us to witness real-time struggles, breakthroughs, and emotional responses that neither surveys nor controlled studies could capture.

This experiential focus also shaped our choice of tasks. Rather than simple pointing exercises or game interactions, we designed creative scenarios that mirror real-world applications where these techniques might be used. Design tasks involving alignment, scaling, and rotation reflect the spatial problem-solving that occurs in genuine VR applications, from arranging virtual furniture to positioning 3D model components. These tasks naturally escalate in complexity, revealing how each technique handles both straightforward movements and intricate multi-modal interactions that demand simultaneous consideration of position, size, and orientation.

To analyze the data collected from the user study, we selected non-parametric statistical methods appropriate for repeated measures and ordinal data characteristics. The Friedman test was chosen because it robustly evaluates differences across three related interaction techniques within the same participants, without assuming normal distribution of the subjective scores. Given the nature of NASA-TLX ratings—ordinal and potentially non-normally distributed—this test provides a valid omnibus comparison across techniques for each workload dimension. When the Friedman test indicated significant differences, we applied paired Wilcoxon signed rank tests as post hoc analyzes to identify specific technique pairs that differed significantly. This approach allowed us to rigorously assess not only whether differences exist, but also which interaction techniques perform better or worse on dimensions such as mental demand, effort, and frustration, thus providing detailed insights into user experience beyond raw performance metrics.

5.2 TASKS AND PROCEDURE

Participants were comfortably seated in a chair throughout the experiment to ensure consistent positioning and reduce fatigue during extended interaction sessions. Initially, participants completed a GDPR consent form and demographic questionnaire. Following these formalities, they received an introduction to the overall structure of the study, which consisted of three distinct phases:

5.2.1 Tutorial Phase

The first phase, where we introduce users to the application's overall concept and functionality. After a brief overview of the application and its interaction modalities, participants navigated through the application following step-by-step visual instructions that covered all aspects of the tools and interaction techniques in both 2D and 3D environments.

5.2.2 Testing Phase

After the user completed the tutorial, they went through Meta's eye calibration process - basically tracking a pink dot that moves around the screen to make sure the system could accurately detect where the user was looking. Once that was done, they moved on to the main test, which had three tasks, which we will see in Section 5.2.4. Each task tested a different technique (***Hand Ray***, ***Gaze + Pinch*** and ***Look&Drop***) for manipulating virtual objects.

To keep things fair and prevent people from getting better just because they'd done similar tasks before, we mixed up the order of these interaction methods for different participants. Before jumping into the actual tests, users got some practice time with the technique. They could play with three different virtual objects, as shown in Figure 5.1d - one to move things around, one to rotate, and one to scale - so they could get comfortable with how everything worked without any pressure.

Once they felt ready with a particular interaction method, they'd move on to the real evaluation tasks using that same method. In this way, we made sure everyone was comfortable with the controls before we started measuring their performance.

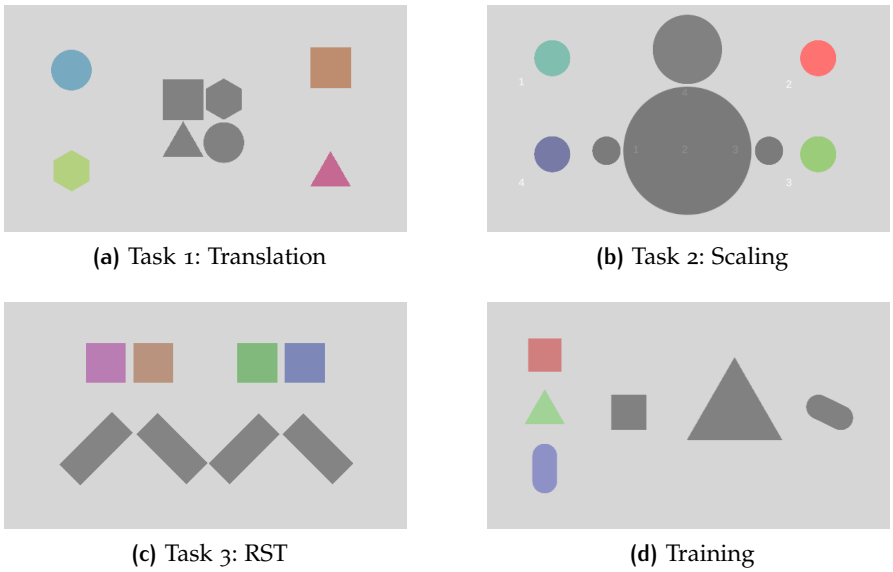
5.2.3 Feedback Phase

After each task (tutorial followed by testing rounds) is completed, the participant is asked to complete a feedback form about their experience. The user completes an initial feedback on the application after the tutorial, followed by three technique-specific evaluations after testing each interaction technique. For each technique, users complete a NASA-TLX[1] questionnaire using 7-point Likert scales to assess their task experience. The study concludes with two additional forms: technique ranking based on individual tasks and overall technique preference ranking.

1 NASA TLX https://humansystems.arc.nasa.gov/groups/tlx/

5.2.4 Task Design

The main test had **three different challenges**, each task testing different ways of manipulating objects. Each task included four Gray target objects and matching shapes that participants needed to align with. The tasks got progressively harder and more complex, as shown in Figure 5.1.



(a) Task 1: Translation

(b) Task 2: Scaling

(c) Task 3: RST

(d) Training

Figure 5.1: Overview of different tasks during the Testing Phase

Task 1: Translation

This was the simplest task - users just had to move colored shapes to line up with their matching shapes by moving them left, right, up, and down. No resizing or rotating allowed. This test evaluated how well people could select and position objects using the assigned interaction method Figure 5.1a.

Task 2: Translation and Scaling

Here, users had to move and resize four objects. Each color circle had to be lined up with a masked target of a different size, so people had to move the objects to the correct spot and adjust their size to match. This tested whether people could handle doing two things at once and showed us how easy (or frustrating) it was to make fine size adjustments with each interaction method Figure 5.1b.

Task 3: Rotating, Scaling, and Translation

This was a challenging task that required users to rotate, resize, and move objects to match target shapes. Each target was in a different location, shape and rotation, so people had to rotate objects around, make them the right size, and put them in the right place. This task was designed to be similar to real-world design work and gave us the best overall picture of how well each interaction method performed Figure 5.1c.

5.3 RESULTS

In this section, we will go through the findings from each of the feedback that we collected from the users after completing each of the tasks.

5.3.1 Participants

We recruited 18 participants aged 21-34 years ($M = 24.22$ years, $SD = 3.19$ years). The sample included 12 males, 5 females, and 1 non-binary participant, where 15 participants were right-handed and 3 left-handed, and 7 of them had Vision correction. The participants came from diverse academic backgrounds, including computer science, chemistry, astrobiology, political science, intercultural studies, HCI, Molecular medicine, IT product development, and medicine. Regarding VR experience, 13 participants had some prior VR experience, while 5 had no previous VR experience. Most of the participants had limited VR/MR experience overall ($M = 2.39$, $SD = 1.29$) and prototyping experience ($M = 1.5$, $SD = 1.04$).

5.3.2 Application feedback

From the application's feedback, users were quite positive about the VR application overall, finding it intuitive and surprisingly accessible even for those new to VR (P10, P15). Many praised the object manipulation features, describing the "good, effortless scaling, rotation and movement" (P1) and noting how smoothly the program worked. The interface design was well-received too, with users appreciating the "really nice design" and "clear tutorial" (P5) that made actions feel natural and easy to follow. What stood out was how quickly people adapted to it: first-time VR users mentioned that it was "surprisingly easy to use" (P10, P15) and became intuitive with just a little practice (P11). The 2D functionality worked particularly well and was easy to understand. However, users did point out some areas that could be improved, especially around tool selection indicators, where it was not always clear which tool was active (P7, P16). However, the application appears to have hit its mark in creating an engaging and user-friendly interface for multimodal object manipulation, particularly for people who are new to VR.

5.3.3 Tests feedback

Looking at the NASA-TLX workload assessment results from Figure 5.2, several interesting patterns emerge across the three interaction techniques. *Hand Ray* demonstrates the highest physical demand scores, which aligns with expectations given that users must main-

tain hand positioning and perform precise gestures throughout the interaction. This increased physical effort is reflected in participants' feedback, with several noting wrist fatigue and the challenge of maintaining hand visibility. P13 mentioned: "I got a bit annoyed for the precision because my wrist started to hurt," while P7 noted that "rotation, scaling, and translation with *Hand Ray* were really smooth." Interestingly, despite the higher physical demands, *Hand Ray* shows relatively low frustration levels and moderate effort scores, suggesting that users find the technique predictable and controllable even when physically taxing. In contrast, *Look&Drop* shows the lowest physical demand but the highest mental demand, indicating that while the technique reduces physical strain, it requires greater cognitive effort to coordinate eye movements with intended actions. P3 expressed this challenge, stating "Using the gaze to scale and translation task is really hard and I have to concentrate on the position I want to move/scale, however I feel like gaze can not work as I want, because of the jitter of gaze." The temporal demand remains relatively consistent across all techniques, although *Look&Drop* shows slightly higher scores, probably due to the learning curve associated with eye-tracking interactions. The *Gaze + Pinch* technique appears to strike a middle ground across most dimensions, showing moderate scores in both physical and mental demand while maintaining relatively good performance ratings and the lowest overall effort scores, with P10 commenting that "This was the easiest one to use. It felt most intuitive and easy."

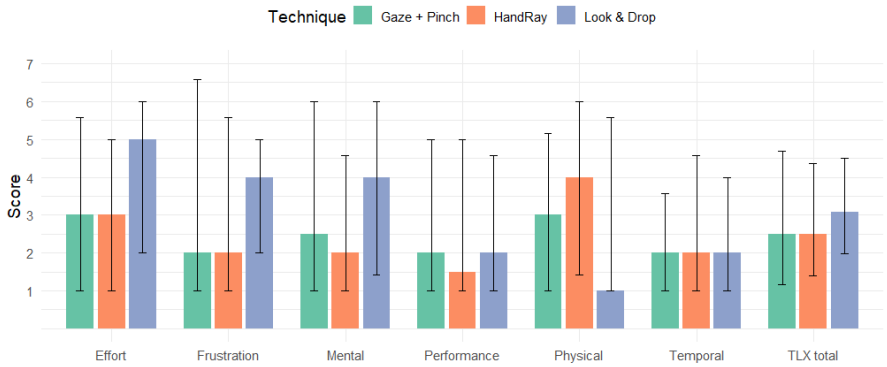


Figure 5.2: Median results from NASA-TLX

Our statistical analysis on NASA-TLX data revealed significant differences between interaction techniques. Mental demand differed significantly ($\chi^2(2) = 14.56$, $p < .001$, $W = 0.404$), where *Hand Ray* was perceived as less mentally demanding than *Look&Drop* ($p < .001$) and *Gaze + Pinch* ($p = .049$), and *Gaze + Pinch* was also rated as less demanding than *Look&Drop* ($p = .013$). Effort showed a significant effect ($\chi^2(2) = 10.34$, $p = .006$, $W = 0.287$), with both *Hand Ray* and *Gaze + Pinch* being rated as less effortful than *Look&Drop* ($p = .004$ and $p = .002$, respectively). Similarly, frustration differed significantly

across techniques ($\chi^2(2) = 10.83$, $p = .004$, $W = 0.301$), with *Hand Ray* perceived as less frustrating than *Look&Drop* ($p = .007$) and *Gaze + Pinch* also rated lower than *Look&Drop* ($p = .041$). However, no significant differences were found for physical demand ($\chi^2(2) = 5.38$, $p = .068$), temporal demand ($\chi^2(2) = 2.97$, $p = .227$), or perceived performance ($\chi^2(2) = 2.57$, $p = .276$). These patterns were confirmed by Wilcoxon signed rank post hoc comparisons conducted across all 18 participants.

5.3.4 Ranking

Starting with the overall ranking as shown on the right side of Figure 5.3, the analysis based on 18 participants reveals a clear hierarchy in user preferences among the three interaction techniques. *Gaze + Pinch* emerges as the dominant technique, securing approximately 70% of the first-place rankings, demonstrating its overwhelming acceptance among users as the preferred interaction method. *Hand Ray* shows moderate performance with roughly 25% of first-place votes, but struggles to maintain consistency across ranking positions, indicating polarized user opinions about its effectiveness. *Look&Drop* faces significant challenges, and the majority of participants ranked it third (approximately 60%), suggesting fundamental usability issues that hinder user adoption and satisfaction.

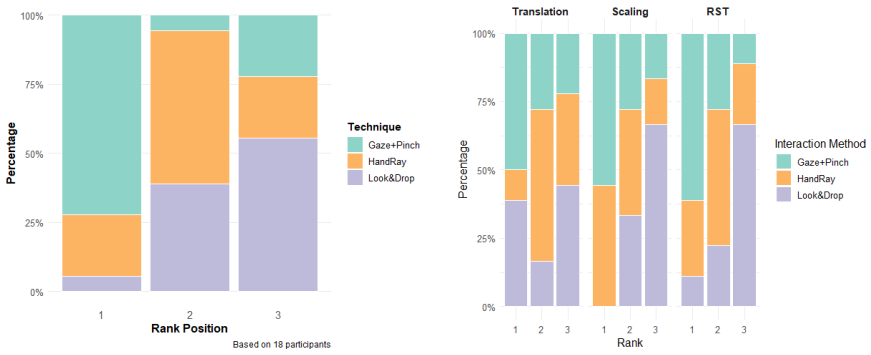


Figure 5.3: Task-wise and overall ranking of the techniques

When examining task-specific performance across Translation, Scaling, and RST (Rotation, Scaling, Translation) operations, distinct patterns emerge that highlight each technique's strengths and weaknesses, as shown in the right side of Figure 5.3. For translation tasks, *Gaze + Pinch* maintains its superiority with approximately 50% first-place rankings, while *Hand Ray* and *Look&Drop* compete more closely for the second and third positions. Scaling operations reveal the most dramatic performance differences, where *Hand Ray* shows improved competitiveness, achieving nearly 40% of the top rankings, while *Look&Drop* consistently underperformed in all ranking positions. The RST combined task analysis demonstrates *Look&Drop*'s particular struggles with complex multi-modal interactions, consistently rank-

ing third in over 70% of cases, while *Gaze + Pinch* maintains steady performance and *Hand Ray* shows variable but competitive results.

5.4 PROTOTYPE AND EXPERIMENTAL SETUP

All participants used a Unity-based VR application that runs on the Meta Quest Pro headset. The prototype had three different techniques to manipulate objects: Hand Ray, *Gaze + Pinch*, and *Look&Drop*. Users could easily switch between working with any of the interaction techniques to move, rotate, and resize objects.

We designed tasks that felt like real creative work, similar to arranging elements in kindergarten. Participants had to align shapes with targets, resize objects to approximately match given sizes, and rotate items to fit specific orientations.

Each participant attempted all three techniques on the same set of tasks mentioned in 5.2.2, allowing us to compare their effectiveness and identify which was preferred. We collected user feedback to understand how easy, intuitive, and physically demanding each technique was.

5.4.1 Learnability

Few participants mentioned an initial learning curve and need for assistance at the beginning, but once they understood the system and how they should use different techniques to perform manipulation, they felt comfortable: "While there's an initial learning curve and some assistance needed at the beginning, it is worth it once you understand the setup." In the test questionnaire, all users rated the tool to be fluent, easy to use, and quick to learn for most people.

5.4.2 Design Pipeline

We asked participants how they liked the design pipeline and process. The common feedback was positive. P5 said: "The application has a really nice design, the tutorial is clear, easy to follow, actions needed are intuitive. I also liked the choice of colors and the menu placement. ", with similar responses from other participants. Several participants found the concept of using manipulation techniques easy and useful and that it provides an 'I would like to have this type of interaction in VR with the application I work with in my study'. The participants also appreciated that all the techniques work well in performing the manipulation tasks.

5.4.3 Comparison with Professional Tools

Professional VR technologies, such as Gravity Sketch, Tilt Brush, and ShapesXR, are equipped with useful tools for 3D modeling and creative projects. These applications give artists and designers great flexibility, which at the same time means they are not easy to use. Because hand gestures or controllers are important, using them for a long time can get tiring. These tools are designed for individuals who already have a basic understanding of VR, and they are not intended to help you experiment with interacting with virtual objects.

We use a different strategy in our application. That way, it focuses on helping users understand and test how gaze and hand movements affect object interactions. It has a lightweight UI and feels more like a creative design tool, a little like PowerPoint for virtual reality. By focusing on this simple idea, we can concentrate on the most important aspect of this study: how easily and comfortably gaze-based methods work when people attempt to move, scale, or rotate objects. Therefore, since it does not have all the features of Pro Tools, we get a clean workspace that helps us to see how these interactions can work in creative jobs.

6 DISCUSSION

This study evaluated three VR interaction techniques for object manipulation tasks: *Hand Ray*, *Gaze + Pinch*, and *Look&Drop*. The results reveal distinct patterns in user preferences, performance characteristics, and subjective workload across different manipulation tasks.

6.1 STUDY PARTICIPANTS

We recruited 18 paid participants for this study, as with three techniques being evaluated. The participants came from diverse academic backgrounds, including computer science, chemistry, political science, HCI, and medicine, providing cross-disciplinary perspectives on VR interaction design. Most participants had limited prior VR experience, making our findings particularly relevant for novice users as VR technology becomes more accessible, though the relatively young participant age range may limit generalizability to older adult populations.

The varied AR/VR experience levels among participants likely influenced the results, particularly for *Look&Drop*, which showed the steepest learning curve. Participants with prior VR experience may have adapted more quickly to novel interaction paradigms, while VR novices might have been more conventional in their technique preferences, potentially favoring *Gaze + Pinch* for its familiar hand-based component. The diverse academic backgrounds also contributed to result variability, as participants from technical fields (computer science, HCI) might have been more tolerant of interaction complexity compared to those from non-technical backgrounds.

6.2 USER PERFORMANCE

Gaze + Pinch was found to be the best method in all the evaluation measures. The trends shown in Section 5.3 indicate that approximately 70% of people considered the technique as their favorite in overall preference, which shows people preferred this technique.

According to NASA-TLX ratings, *Gaze + Pinch* performs well because it uses less effort than other techniques and requires only a moderate amount of mental and physical work. Statistical analysis revealed that *Gaze + Pinch* was rated as significantly less mentally demanding than *Look&Drop* and required significantly less effort com-

pared to the pure gaze-based approach. User feedback consistently praised its balanced approach, with participants noting that it felt "most intuitive and easy to use"(P10). The technique's success stems from its dual-modality approach, combining eye-tracking for fast object identification with hand movements for accurate manipulation, effectively addressing the limitations encountered by devices that rely on just one type of interaction. However, as the technique relies on pinch detection can introduce inconsistencies, such as when the user moves their arm, potentially occluding the pinch gesture and preventing successful manipulation, and prolonged use can lead to fatigue from maintaining coordinated eye-hand movements, although this was less pronounced than with purely hand-based approaches.

Looking at task-specific performance, *Gaze + Pinch* proved effective across diverse manipulation operations, consistently placing first in translation tasks and performing well in complex RST procedures. This consistency demonstrates the dual-modality approach's versatility across varying task complexity levels.

Hand Ray showed strong performance in precision-oriented tasks, particularly in the scaling task, where it achieved nearly 40% of top rankings. The technique's strength lies in its direct hand-to-object mapping, which gives users intuitive control for manipulating objects as they move where the user is pointing. Statistical analysis confirmed that *Hand Ray* was perceived as the least mentally demanding technique across all three methods.

However, *Hand Ray*'s limitations become apparent during extended use. The NASA-TLX results showed the highest physical demand scores among all techniques, indicating significant ergonomic challenges. Users reported wrist strain during prolonged interactions, and one participant noting: "I got a bit annoyed with the precision because my wrist started to hurt."(P13). The physical demands of the technique manifest themselves in reported fatigue and the challenge of maintaining hand visibility throughout interactions, although it maintains relatively low levels of frustration, suggesting that users find it predictable and controllable despite physical costs.

Look&Drop showed the most unfavorable user responses, with approximately 60% of participants ranking it third overall. The technique shows promise for fast object positioning and simple translation tasks, where the direct eye-to-object mapping demonstrates intuitiveness for basic movements: looking at the object, performing a pinch gesture, and then looking at the desired location results in the object moving to that place.

The NASA-TLX analysis reveals *Look&Drop*'s contradictory nature: while showing the lowest physical demand, it demonstrates the highest mental demand scores and significantly higher effort and frustration levels compared to both other techniques. This pattern indicates that the technique reduces physical strain but requires significant mental

effort to coordinate eye movements with intended manipulations. User feedback confirmed these usability challenges, with participants expressing frustration about gaze jitter and precision difficulties. One participant noted: "Using the gaze to scale and translation task is really hard and I have to concentrate on the position I want to move/scale, however I feel like gaze cannot work as I want, because of the jitter of gaze."(P3).

Complex manipulation tasks reveal *Look&Drop*'s primary limitations. The technique struggles mainly with tasks such as scaling and rotating, because users must simultaneously look at the object being manipulated and monitor the transformation, which leads to discomfort and decreased precision. However, participants mentioned that while it represents a novel way to interact with objects, it requires significant adaptation time to become proficient at manipulating objects solely through eye movements. Some users, after gaining familiarity with this technique, performed tasks more effectively.

6.3 TASK COMPLEXITY AND INTERACTION PATTERNS

Translation tasks, which are the simplest of all manipulation operations, showed better performance in all techniques, with more evenly distributed user preferences. Both the gaze-based techniques showed competitive performance in basic positioning, supporting the viability of eye-based input for straightforward object movement scenarios.

Complex manipulation tasks like scaling showed clear hierarchical preferences favoring techniques that provide precise control. *Gaze + Pinch* consistently outperformed other techniques in scaling operations, while *Hand Ray* maintained strong performance despite higher physical demands. *Look&Drop*'s performance degraded significantly in scaling tasks, highlighting the limitations of pure gaze-based manipulation for complex operations.

The most complex combined RST tasks demonstrated the importance of technique versatility, with *Gaze + Pinch* maintaining consistent performance across all operation types while *Look&Drop* showed particular struggles with multi-modal interactions, ranking third in over 70% of cases.

6.4 DESIGN IMPLICATIONS

The success of *Gaze + Pinch* demonstrates that multimodal interaction is key to effective object manipulation in VR environments. By incorporating both gaze and hand input, the technique benefits from the

strengths of each input modality while gaining additional synergies that enhance the user experience.

The performance differences across interaction techniques suggest that optimal VR interaction may require adaptive approaches based on task requirements. Gaze-based selection can be valuable for basic positioning, while hand-based manipulation tends to be superior for precise control operations.

Hand Ray makes it evident that ergonomics plays a fundamental role in VR interfaces. For this technique to be viable for extended VR sessions, methods are needed to reduce physical strain while maintaining interaction precision. *Gaze + Pinch* demonstrates how combining different input modalities can address ergonomic challenges while preserving manipulation accuracy.

Meanwhile, *Look&Drop* opens up new possibilities for VR interaction design by demonstrating that pure gaze-based manipulation is feasible, even if the current implementation faces precision challenges. The technique's ability to eliminate physical strain makes it particularly valuable for accessibility applications and extended use scenarios. Its approach of connecting visual attention directly to object manipulation represents a different model of human-computer interaction that could become more effective as eye-tracking technology improves.

6.5 USER EXPERIENCE AND ADOPTION PATTERNS

The evaluation revealed a significant variation in the patterns of adoption of the technique. *Gaze + Pinch* was accepted by users quickly and required minimal learning time, while *Look&Drop* needed users to invest considerable time in adaptation. However, *Hand Ray* was also readily accepted by users as it represents a simple technique for manipulating objects across all RST tasks.

The comparison between *Gaze + Pinch* and *Look&Drop* is particularly interesting: while *Gaze + Pinch* provides immediate usability through its combination of familiar interaction methods, *Look&Drop* requires users to develop new skills but offers unique advantages in terms of physical comfort and accessibility. Some participants noted that *Look&Drop* became more manageable with practice, suggesting that the technique's current limitations may decrease with user adaptation and technological improvements.

User feedback on the application was generally positive, with participants expressing satisfaction with the tutorials and interface design. Many users praised the system's success in teaching VR interaction concepts to newcomers. The accessibility and ease of use of the application demonstrate that focusing on core interaction techniques was an appropriate design choice.

6.6 LIMITATIONS

Based on user feedback, several key limitations emerge across the three interaction techniques while manipulating objects in rotation, scaling, and translation tasks in VR environments. The *Look&Drop* technique, while offering intuitive object translation, suffers from notable precision challenges, particularly during scaling and rotation operations. Users consistently reported that small eye movements produce disproportionately large changes in object manipulation, making fine adjustments extremely difficult. The gaze jitter inherent in eye tracking compounds this issue, with participants noting frustration when trying to achieve precise control. Additionally, the technique requires users to divide their attention between looking at the object being manipulated and the target location, creating a mental load that many found counterintuitive and mentally demanding.

The *Hand Ray* and *Gaze + Pinch* techniques present distinct limitations despite generally superior controllability. *Hand Ray* users frequently experienced physical fatigue and wrist strain during extended use, require to maintained precise hand positioning that becomes uncomfortable over time. The method struggles with very fine adjustments, particularly for scaling operations requiring minute changes. *Gaze + Pinch*, despite its overall efficiency, creates a disconnection between visual selection and manual manipulation that some users found disorienting. The technique's dependence on precise interaction controls makes targeting exact manipulation points challenging, and many noted that eye tracking and hand gesture coordination did not always flow smoothly, particularly during rapid or complex manipulation sequences.

7 CONCLUSION

This comprehensive evaluation delivers clear answers to the research questions on integrating multimodal interaction for RST tasks in virtual environments, revealing two distinct and complementary directions for VR interface development. *Gaze + Pinch* demonstrated superior performance on standard evaluation measures, with approximately 70% of the participants ranking it as their preferred technique, and NASA-TLX assessments confirming its balanced approach of significantly lower mental demand than *Look&Drop* and reduced effort compared to pure gaze-based methods. This multimodal technique successfully combines the natural speed of eye tracking for rapid object identification with hand gestures for accurate manipulation control, creating a seamless interaction paradigm that addresses the fundamental limitations of single-input methods while maintaining cognitive efficiency and user satisfaction. However, there are some limitations to *Gaze + Pinch*, including occlusion of the hand during the interaction that leads to disrupted manipulation and inconsistent input recognition. In contrast, *Look&Drop* represents an equally significant contribution as a pure gaze-based manipulation paradigm that introduces a radical departure from conventional interaction models by eliminating hand gestures. Despite current implementation challenges, including precision difficulties and gaze jitter, this approach transforms the conceptualization of virtual object manipulation by converging intention, attention, and action into a singular direct relationship between user gaze and object behavior.

The findings support two meaningful research trajectories that advance VR interaction design. *Gaze + Pinch* provides a practical solution for current VR applications, demonstrating that optimal multimodal integration through gaze-based selection and hand-based manipulation creates sustainable and satisfying user experiences for extended creative work in virtual environments. Its success validates the importance of leveraging the strengths of multiple input modalities while maintaining ergonomic sustainability and intuitive control. Conversely, *Look&Drop*'s unique position as the technique that eliminates physical strain opens new possibilities for accessibility and extended VR sessions, while its high mental demand scores indicate the emergence of new forms of spatial cognition that users can develop over time. According to participants, *Look&Drop* was considered the fastest for object translation, which makes it particularly well suited for rapid repositioning tasks in spatial layouts. Together, these gaze-based techniques contribute vital insights to the field: *Gaze + Pinch* establishes the

current standard for practical VR manipulation through proven multimodal efficiency, while *Look&Drop* provides a foundational proof of concept for direct intention-to-action manipulation that could revolutionize how we understand the relationship between human cognition and digital environments. Collectively, they define both the present capabilities and the future potential of gaze-based interaction in virtual reality systems.

7.1 FUTURE WORK

While this study demonstrates the effectiveness of multimodal gaze-based interaction techniques, several areas need improvement from user feedback that need further investigation. The most critical enhancement involves perfecting rotation and scaling operations with 3D objects, particularly addressing the current limitation of single-axis rotation by implementing multi-axis rotation capabilities that would allow users to manipulate objects more naturally in three-dimensional space. Users consistently requested more responsive scaling interactions, suggesting the need for easier scaling mechanisms that provide immediate visual feedback and reduce the lag between user input and object transformation. Furthermore, improving visual feedback systems based on participant suggestions could significantly enhance the overall user experience by providing clearer indicators of active tools, manipulation boundaries, and real-time transformation previews. Future research should implement adaptive gaze-target thresholds that dynamically adjust according to individual user behavior patterns and fatigue levels, potentially using machine learning algorithms to optimize interaction accuracy over time.

Looking beyond the refinement of existing techniques, future research should explore expanding the creative capabilities of gaze-enhanced VR environments through the integration of free-hand drawing tools for 3D object creation. This would allow users to sketch and create their desired objects directly in 3D space using natural hand movements combined with gaze direction, potentially revolutionizing how designers conceptualize and prototype ideas in virtual environments. Such developments could bridge the gap between traditional sketching workflows and advanced 3D modeling, making VR design tools more accessible to artists and designers who prefer drawing-based creative processes. Further investigation of improved eye tracking accuracy and cross-platform compatibility would also strengthen the practical applicability of these multimodal techniques in professional design workflows.

A AN APPENDIX

A.1 MULTI-SELECT IMPLEMENTATION

The multiselect functionality operates differently between 2D and 3D environments, adapting to the specific interaction paradigms of each spatial context. In both cases, the system detects when users activate multiselect mode by interacting with a designated UI element, then allows them to draw selection regions through continuous input gestures.

For the 2D implementation, the selection mechanism creates a rectangular selection area using a LineRenderer to draw the outline and a MeshRenderer to provide a semi-transparent fill. The system captures the starting position when users begin dragging and dynamically updates the rectangle boundaries as they move their input. The core selection detection uses 2D physics to find all objects within the drawn rectangle.

```
// Create bounds from the two corners
Vector3 min = new Vector3(
    Mathf.Min(selectionStartPosition.x, currentPosition.x),
    Mathf.Min(selectionStartPosition.y, currentPosition.y),
    selectionStartPosition.z
);

Vector3 max = new Vector3(
    Mathf.Max(selectionStartPosition.x, currentPosition.x),
    Mathf.Max(selectionStartPosition.y, currentPosition.y),
    selectionStartPosition.z
);

// Find all objects under the mask
Collider2D[] colliders = Physics2D.OverlapAreaAll(min, max);
```

Listing A.1: 2D Selection Detection with creating a bounded box

The 3D multiselect approach creates volumetric selection regions using a transparent cube primitive that users can resize by dragging. Instead of flat rectangular areas, this method accommodates depth perception and spatial relationships in three-dimensional space. The selection detection uses 3D physics overlap to capture objects within the defined volume.

```
// Get the world space bounds of selection volume
Vector3 center = selectionMask.transform.position;
Vector3 extents = visualCube.localScale * 0.5f;

```

```
// Find all colliders within the bounds
Collider[] colliders = Physics.OverlapBox(center, extents, Quaternion.
    identity, targetLayer);

foreach (Collider collider in colliders)
{
    if (collider.CompareTag("Target"))
    {
        selectedObjects.Add(collider.gameObject);
        collider.transform.SetParent(selectionMask.transform);
    }
}
```

Listing A.2: 3D Selection with a rectangular Volume Detection

Both implementations share the fundamental approach of creating visual feedback during selection and parenting selected objects to a container for unified manipulation. The key distinction lies in their spatial detection methods - 2D uses "Physics2D.OverlapAreaAll()" for rectangular bounds while 3D employs "Physics.OverlapBox()" for volumetric space. After finalizing selections, the systems enable users to manipulate multiple objects simultaneously as a single group, maintaining the visual selection boundaries to provide clear feedback about the grouped objects.

A.2 COLOR PICKER IMPLEMENTATION

The color picker functionality operates differently between 2D and 3D environments, adapting to the specific interaction paradigms of each spatial context. In both cases, the system detects when users activate color sampling mode by positioning their cursor over color sources, then allows them to extract and apply colors through direct interaction with visual elements.

In the 2D implementation, the system uses "Physics2D.CircleCast" to detect color sources and applies sampled colors to various target types. The core workflow begins by casting a small circular ray at the cursor position to identify objects tagged as "Color", then extracts the exact pixel color from sprites or UI elements at that specific world position.

```
RaycastHit2D hit2D = Physics2D.CircleCast(endPoint, hoverOffset, Vector2.
    zero, 0);

if (hit2D.collider != null && hit2D.collider.CompareTag("Color"))
{
    isHoveringColor = true;
    copiedColor = SampleExactSpriteColor(hit2D.point, hitObject);

    // Apply to selected target
    if (selectcolObject != null && selectcolObject.tag == "Target")
    {
```

```
        Renderer renderer = selectcolObject.GetComponent<Renderer>();
        for (int i = 0; i < renderer.materials.Length; i++)
        {
            renderer.materials[i].color = copiedColor;
        }
    }
}
```

Listing A.3: 2D Color Detection and Sampling

The 3D version replaces the 2D physics with "Physics.SphereCast" to accommodate three-dimensional space, using volumetric detection instead of flat circular casting. This allows the system to work naturally with 3D objects and maintains depth-aware interaction while preserving the same color sampling logic.

```
RaycastHit hit;
if (Physics.SphereCast(rayOrigin, 0.01f, rayDirection, out hit,
    rayDistance))
{
    if (hit.collider.CompareTag("Target") || hit.collider.CompareTag("
        DrawnLine"))
    {
        GameObject targetObject = hit.collider.gameObject.transform.
            parent != null
            ? hit.collider.gameObject.transform.parent.gameObject
            : hit.collider.gameObject;

        if (isHoldingNow)
        {
            selectcolObject = targetObject;
            InitialColor = targetObject.GetComponent<Renderer>()?.
                material.color ?? Color.white;
            selectCol = true;
        }
    }
}
```

Listing A.4: 3D Spatial Color Detection

Both implementations handle complex color sampling scenarios, including sprite textures, UI images, and specialized color wheels. The "SampleExactSpriteColor" function converts world coordinates to texture pixel coordinates, accounting for sprite boundaries, pivot points, and scaling factors. When direct texture access fails due to read/write restrictions, the system creates temporary render textures to extract pixel data, ensuring reliable color sampling across different texture configurations.

The key difference between 2D and 3D approaches lies in their spatial detection methods - 2D uses flat circular casting while 3D employs spherical volumetric detection. However, both maintain identical color application logic, supporting materials, line renderers, and shape fills through unified interfaces. This design allows artists and developers

to work seamlessly across dimensional contexts while the underlying system handles the technical complexities of accurate color sampling and application.

A.3 FREEHAND DRAWING

The free hand drawing functionality operates as a 2D interaction system that enables users to create custom shapes through pinch gestures on a canvas surface. When users point at the canvas and perform pinch actions, the system creates circular drawing points at those locations, building up a connected shape through sequential point placement.

The drawing process begins when the system detects canvas interaction through raycast detection. Users point at specific locations on the canvas and execute pinch gestures to place drawing points. Each pinch action creates a new circular marker at the targeted position, provided it meets the minimum distance threshold from the previous point.

```
if (Physics.Raycast(rayOrigin, rayDirection, out hit, rayDistance))
{
    if (hit.collider.CompareTag("Canvas"))
    {
        if (isHoldingNow)
        {
            Vector3 drawPoint = hit.point;

            if (drawingPoints.Count == 0 ||
                Vector3.Distance(drawPoint, lastPointPosition) >=
                    minDistanceBetweenPoints)
            {
                CreateDrawingPoint(drawPoint);
                UpdateDrawingLine();
                lastPointPosition = drawPoint;
            }
        }
    }
}
```

Listing A.5: Canvas Interaction and Point Placement

Shape closure occurs when users perform a pinch gesture while pointing at an existing drawing point, provided at least three points have been placed. The system detects this interaction through collision detection and automatically completes the shape by connecting the current drawing state to the selected point. Once closure is detected, the system begins the shape finalization process by creating a polygon collider that matches the drawn geometry and generating a visual sprite representation.

```
private void FinalizeShape(int closeToPointIndex = 0)
```

```
{
    freeHandDrawingMode = false;

    // Add final point to close the shape
    drawingLine.positionCount = drawingPoints.Count + 1;
    drawingLine.SetPosition(drawingPoints.Count, drawingPoints[
        closeToPointIndex].transform.position);

    // Create polygon collider matching the shape
    PolygonCollider2D polygonCollider = currentDrawing.AddComponent<
        PolygonCollider2D>();
    Vector2[] colliderPoints = new Vector2[drawingPoints.Count];

    for (int i = 0; i < drawingPoints.Count; i++)
    {
        colliderPoints[i] = new Vector2(
            drawingPoints[i].transform.position.x,
            drawingPoints[i].transform.position.y
        );
    }
    polygonCollider.points = colliderPoints;

    // Generate sprite for visual representation
    CreateSpriteForShape(colliderPoints);
}
```

Listing A.6: Shape Finalization and Polygon Creation

The sprite generation process calculates the bounding rectangle of the drawn shape and creates a high-resolution texture that fills the polygon area. The system uses advanced anti-aliasing techniques to ensure smooth edges and converts the world-space coordinates to texture-space coordinates for accurate rendering.

```
private void CreateSpriteForShape(Vector2[] points)
{
    // Calculate shape bounds
    Vector2 min = points[0], max = points[0];
    foreach (Vector2 point in points)
    {
        min.x = Mathf.Min(min.x, point.x);
        min.y = Mathf.Min(min.y, point.y);
        max.x = Mathf.Max(max.x, point.x);
        max.y = Mathf.Max(max.y, point.y);
    }

    Vector2 size = max - min;
    float pixelsPerUnit = Mathf.Min(500f / size.x, 500f / size.y);

    int textureWidth = Mathf.CeilToInt(size.x * pixelsPerUnit);
    int textureHeight = Mathf.CeilToInt(size.y * pixelsPerUnit);

    Texture2D texture = new Texture2D(textureWidth, textureHeight,
        TextureFormat.RGBA32, false);
    Color[] colors = new Color[textureWidth * textureHeight];

    // Convert to texture space coordinates
```

```
    Vector2[] texturePoints = new Vector2[points.Length];
    for (int i = 0; i < points.Length; i++)
    {
        texturePoints[i] = new Vector2(
            ((points[i].x - min.x) / size.x) * textureWidth,
            ((points[i].y - min.y) / size.y) * textureHeight
        );
    }

    FillPolygonAntiAliased(colors, textureWidth, textureHeight,
        texturePoints, Color.white);
    texture.SetPixels(colors);
    texture.Apply(true);

    Sprite sprite = Sprite.Create(texture, new Rect(0, 0, textureWidth,
        textureHeight),
                                  new Vector2(0.5f, 0.5f), pixelsPerUnit)
                                      ;
}
```

Listing A.7: Dynamic Sprite Generation with Anti-Aliasing

Upon completion, the system positions the final object at the calculated center of the shape, ensures proper alignment between the sprite and collision boundaries, and automatically places manipulation handles around the created object for immediate interaction capabilities. The original drawing points become invisible while maintaining their reference positions, creating a clean final result ready for further manipulation.

A.4 SOURCE CODE

All source code, implementation files, and project materials related to this thesis are available at:

https://github.com/pratikmohanty1425/LookNRST

BIBLIOGRAPHY

[1] Christoph Anthes, Rubén Jesús García-Hernández, Markus Wiedemann, and Dieter Kranzlmüller. "State of the art of virtual reality technology." In: *2016 IEEE aerospace conference*. IEEE. 2016, pp. 1–19.

[2] Ferran Argelaguet and Carlos Andujar. "A survey of 3D object selection techniques for virtual environments." In: *Computers & Graphics* 37.3 (2013), pp. 121–136.

[3] Joanna Bergström, Tor-Salve Dalsgaard, Jason Alexander, and Kasper Hornbæk. "How to Evaluate Object Selection and Manipulation in VR? Guidelines from 20 Years of Studies." In: *Proceedings of the 2021 CHI Conference on Human Factors in Computing Systems*. CHI '21. Yokohama, Japan: Association for Computing Machinery, 2021. ISBN: 9781450380966. DOI: 10.1145/3411764.3445193. URL: https://doi.org/10.1145/3411764.3445193.

[4] Doug A. Bowman, Donald B. Johnson, and Larry F. Hodges. "Testbed evaluation of virtual environment interaction techniques." In: *Proceedings of the ACM Symposium on Virtual Reality Software and Technology*. VRST '99. London, United Kingdom: Association for Computing Machinery, 1999, 26–33. ISBN: 1581131410. DOI: 10.1145/323663.323667. URL: https://doi.org/10.1145/323663.323667.

[5] Géry Casiez, Nicolas Roussel, and Daniel Vogel. "1 € filter: a simple speed-based low-pass filter for noisy input in interactive systems." In: *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. CHI '12. Austin, Texas, USA: Association for Computing Machinery, 2012, 2527–2530. ISBN: 9781450310154. DOI: 10.1145/2207676.2208639. URL: https://doi.org/10.1145/2207676.2208639.

[6] Ishan Chatterjee, Robert Xiao, and Chris Harrison. "Gaze+ gesture: Expressive, precise and targeted free-space interactions." In: *Proceedings of the 2015 ACM on international conference on multimodal interaction*. 2015, pp. 131–138.

[7] Juan David Hincapié-Ramos, Xiang Guo, Paymahn Moghadasian, and Pourang Irani. "Consumed endurance: a metric to quantify arm fatigue of mid-air interactions." In: *Proceedings of the SIGCHI conference on human factors in computing systems*. 2014, pp. 1063–1072.

[8] Juan David Hincapié-Ramos, Xiang Guo, Paymahn Moghadasian, and Pourang Irani. "Consumed endurance: a metric to quantify arm fatigue of mid-air interactions." In: *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. CHI '14. Toronto, Ontario, Canada: Association for Computing Machinery, 2014, 1063–1072. ISBN: 9781450324731. DOI: 10.1145/2556288.2557130. URL: https://doi.org/10.1145/2556288.2557130.

[9] Sujin Jang, Wolfgang Stuerzlinger, Satyajit Ambike, and Karthik Ramani. "Modeling Cumulative Arm Fatigue in Mid-Air Interaction based on Perceived Exertion and Kinetics of Arm Motion." In: *Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems*. CHI '17. Denver, Colorado, USA: Association for Computing Machinery, 2017, 3328–3339. ISBN: 9781450346559. DOI: 10.1145/3025453.3025523. URL: https://doi.org/10.1145/3025453.3025523.

[10] Jaejoon Jeong, Soo-Hyung Kim, Hyung-Jeong Yang, Gun A Lee, and Seungwon Kim. "GazeHand: A Gaze-Driven Virtual Hand Interface." In: *IEEE Access* 11 (2023), pp. 133703–133716.

[11] Donald E. Knuth. "Computer programming as an art." In: *Commun. ACM* 17.12 (Dec. 1974), 667–673. ISSN: 0001-0782. DOI: 10.1145/361604.361612. URL: https://doi.org/10.1145/361604.361612.

[12] David Ledo, Steven Houben, Jo Vermeulen, Nicolai Marquardt, Lora Oehlberg, and Saul Greenberg. "Evaluation Strategies for HCI Toolkit Research." In: *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems*. CHI '18. Montreal QC, Canada: Association for Computing Machinery, 2018, 1–17. ISBN: 9781450356206. DOI: 10.1145/3173574.3173610. URL: https://doi.org/10.1145/3173574.3173610.

[13] Chang Liu, Jason Orlosky, and Alexander Plopski. "Eye Gaze-based Object Rotation for Head-mounted Displays." In: *Proceedings of the 2020 ACM Symposium on Spatial User Interaction*. SUI '20. Virtual Event, Canada: Association for Computing Machinery, 2020. ISBN: 9781450379434. DOI: 10.1145/3385959.3418444. URL: https://doi.org/10.1145/3385959.3418444.

[14] Chang Liu, Alexander Plopski, and Jason Orlosky. "OrthoGaze: Gaze-based three-dimensional object manipulation using orthogonal planes." In: *Computers & Graphics* 89 (2020), pp. 1–10. ISSN: 0097-8493. DOI: https://doi.org/10.1016/j.cag.2020.04.005. URL: https://www.sciencedirect.com/science/article/pii/S0097849320300431.

[15] Eduardo G. Q. Palmeira, Alexandre Campos, Ígor A. Moraes, Alexandre G. de Siqueira, and Marcelo G. G. Ferreira. "Quantifying the 'Gorilla Arm' Effect in a Virtual Reality Text Entry Task via Ray-Casting: A Preliminary Single-Subject Study." In: *Proceedings of the 25th Symposium on Virtual and Augmented Reality*. SVR '23. Rio Grande, Brazil: Association for Computing Machinery, 2024, 274–278. ISBN: 9798400709432. DOI: 10.1145/3625008.3625046. URL: https://doi.org/10.1145/3625008.3625046.

[16] Nelusa Pathmanathan, Michael Becher, Nils Rodrigues, Guido Reina, Thomas Ertl, Daniel Weiskopf, and Michael Sedlmair. "Eye vs.Head: Comparing Gaze Methods for Interaction in Augmented Reality." In: *ACM Symposium on Eye Tracking Research and Applications*. ETRA '20 Short Papers. Stuttgart, Germany: Association for Computing Machinery, 2020. ISBN: 9781450371346. DOI: 10.1145/3379156.3391829. URL: https://doi.org/10.1145/3379156.3391829.

[17] Ken Pfeuffer, Benedikt Mayer, Diako Mardanbegi, and Hans Gellersen. "Gaze + Pinch Interaction in Virtual Reality." In: *Proceedings of the 5th Symposium on Spatial User Interaction*. SUI '17. Brighton, United Kingdom: Association for Computing Machinery, 2017, 99–108. ISBN: 9781450354868. DOI: 10.1145/3131277.3132180. URL: https://doi.org/10.1145/3131277.3132180.

[18] Ken Pfeuffer, Benedikt Mayer, Diako Mardanbegi, and Hans Gellersen. "Gaze + pinch interaction in virtual reality." In: *Proceedings of the 5th Symposium on Spatial User Interaction*. SUI '17. Brighton, United Kingdom: Association for Computing Machinery, 2017, 99–108. ISBN: 9781450354868. DOI: 10.1145/3131277.3132180. URL: https://doi.org/10.1145/3131277.3132180.

[19] Mel Slater and Maria V Sanchez-Vives. "Enhancing our lives with immersive virtual reality." In: *Frontiers in Robotics and AI* 3 (2016), p. 74.

[20] Jayson Turner, Jason Alexander, Andreas Bulling, and Hans Gellersen. "Gaze+RST: Integrating Gaze and Multitouch for Remote Rotate-Scale-Translate Tasks." In: *Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems*. CHI '15. Seoul, Republic of Korea: Association for Computing Machinery, 2015, 4179–4188. ISBN: 9781450331456. DOI: 10.1145/2702123.2702355. URL: https://doi.org/10.1145/2702123.2702355.

[21] Uta Wagner, Andreas Asferg Jacobsen, Tiare Feuchtner, Hans Gellersen, and Ken Pfeuffer. "Eye-Hand Movement of Objects in Near Space Extended Reality." In: *Proceedings of the 37th Annual ACM Symposium on User Interface Software and Technology*. UIST '24. Pittsburgh, PA, USA: Association for Computing Machinery,

2024. ISBN: 9798400706288. DOI: 10.1145/3654777.3676446. URL: https://doi.org/10.1145/3654777.3676446.

[22] Shu Wei, Desmond Bloemers, and Aitor Rovira. "A Preliminary Study of the Eye Tracker in the Meta Quest Pro." In: *Proceedings of the 2023 ACM International Conference on Interactive Media Experiences*. IMX '23. Nantes, France: Association for Computing Machinery, 2023, 216–221. ISBN: 9798400700286. DOI: 10.1145/3573381.3596467. URL: https://doi.org/10.1145/3573381.3596467.

[23] Dennis Wolf, Jan Gugenheimer, Marco Combosch, and Enrico Rukzio. "Understanding the Heisenberg Effect of Spatial Interaction: A Selection Induced Error for Spatially Tracked Input Devices." In: *Proceedings of the 2020 CHI Conference on Human Factors in Computing Systems*. CHI '20. Honolulu, HI, USA: Association for Computing Machinery, 2020, 1–10. ISBN: 9781450367080. DOI: 10.1145/3313831.3376876. URL: https://doi.org/10.1145/3313831.3376876.

[24] Siju Wu, Amine Chellali, and Samir Otmane. "FingerOscillation: clutch-free techniques for 3D object translation, rotation and scale." In: *Proceedings of the 20th ACM Symposium on Virtual Reality Software and Technology*. VRST '14. Edinburgh, Scotland: Association for Computing Machinery, 2014, 225–226. ISBN: 9781450332538. DOI: 10.1145/2671015.2671117. URL: https://doi.org/10.1145/2671015.2671117.

[25] Difeng Yu, Xueshi Lu, Rongkai Shi, Hai-Ning Liang, Tilman Dingler, Eduardo Velloso, and Jorge Goncalves. "Gaze-Supported 3D Object Manipulation in Virtual Reality." In: *Proceedings of the 2021 CHI Conference on Human Factors in Computing Systems*. CHI '21. Yokohama, Japan: Association for Computing Machinery, 2021. ISBN: 9781450380966. DOI: 10.1145/3411764.3445343. URL: https://doi.org/10.1145/3411764.3445343.

[26] Shumin Zhai, Carlos Morimoto, and Steven Ihde. "Manual and Gaze Input Cascaded (MAGIC) Pointing." In: *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. CHI '99. Pittsburgh, Pennsylvania, USA: Association for Computing Machinery, 1999, 246–253. ISBN: 0201485591. DOI: 10.1145/302979.303053. URL: https://doi.org/10.1145/302979.303053.