2D Selection in Virtual Reality with Head Mounted Displays

by

Adrian Ramcharitar

A thesis submitted to the Faculty of Graduate and Postdoctoral Affairs in partial fulfillment of the requirements for the degree of

Master of Computer Science

in

Human-Computer Interaction

Carleton University
Ottawa, Ontario

Abstract

We present an evaluation of a new selection technique for virtual reality (VR) systems presented on head-mounted displays. The technique, dubbed EZCursorVR, presents a 2D cursor that moves in a head-fixed plane, simulating a 2D desktop-like cursor for VR. The cursor can be controlled by any 2 degree of freedom (DOF) and 3/6DOF input device. We conducted two experiments based on ISO 9241-9. In the first study, we compared the effectiveness of EZCursorVR using six different controllers. Results indicate that the mouse offered the best performance, while the position-control joystick performed the worst. In the second study we evaluate EZCursorVR using three different transfer functions using the mouse with different degrees of cursor acceleration. Results indicate that, despite previous research, constant acceleration performed better than the other two transfer functions. We believe that future evaluation needs to be conducted to evaluate different acceleration curve steepnesses using the same transfer function.

Acknowledgements

First and foremost, I would like to thank my thesis supervisor, Dr. Robert Teather for his continuous guidance and support throughout my time as a graduate student. I am grateful to him for not only teaching me how to be a better researcher, but also for pushing me to publish my work in academic conferences. His perseverance in ensuring that my research wasn't just confined to Carleton University, led to several publications and gave me the opportunity to present my work internationally in Denver, CO and Brighton, UK.

I would also like to thank all my colleagues and classmates for all the fun times and for helping make those stressful days before deadlines more bearable. Thank you to my friends for their interest and helpfulness in my research, which made me feel reassuring in the work I did. I would like to express my deepest gratitude to my friends Yasin Farmani and Elliott Martin for their extensive help and for going out of their way by staying late in our lab to help pilot test and tweak my user study.

Finally, I would like to thank my parents for all their help and support, not only throughout my Master's degree but also throughout my entire life. Without their help and support, I wouldn't have had the opportunity to pursue a graduate education. I am eternally grateful for their perseverance in ensuring that I have the best education possible, no matter the cost.

My work would not have been possible without all these people.

Table of Contents

List of Figures

List of Appendices

1 Chapter: Introduction

Selection is a fundamental task in virtual reality (VR) user interaction, and involves specifying a point or object for subsequent operations. Consider, for example, shooting an enemy in a VR first-person shooter game, or grasping a virtual object presented in a museum exhibit; both tasks involve selection of targets. Selection in VR has traditionally been divided into two (rough) classes of virtual hands (requiring depth precision to grasp an object) and ray-based techniques (requiring remote pointing at a target) [3]. There are numerous selection techniques that have been previously developed for use in VR (see e.g., [1,2,11,16]). Previous work has shown that two dimensional (2D) selection techniques outperform three dimensional (3D) selection techniques [4,29,41,42]. As of yet, no such technique exits for use with head mounted displays (HMDs). Therefore, we developed a 2D section technique for use with HMDs to address the question: Does 2D based input offer better performance in comparison to 3D based input in HMD VR environments?

One common selection technique used with devices such as Microsoft's Hololens and various "cardboard" VR[1] displays is to use a ray cast from the head (controlled by head rotation) in lieu of a 3D wand, presenting a cursor fixed in the centre of the screen. However, since the viewpoint is coupled to the selection ray, this can yield excessive head motion which in turn can cause neck fatigue, and disorient users. Worse, the excessive head movement of the user may induce nausea and cybersickness [12]. To avoid these problems, most modern head-mounted displays (e.g., the Oculus Rift, and HTC Vive) employ tracked wand input devices. While immersive, 6 degree of freedom (6DOF) devices employing

[1] Including devices that use a smartphone as the display such as Google Cardboard (https://vr.google.com/cardboard/) and Samsung's Gear VR (http://www.samsung.com/global/galaxy/gear-vr/)

typical virtual hand or ray-based selection techniques can be problematic. Depth perception is imprecise leading to inaccuracy with selection methods that require accuracy in depth [7,45], and latency (input lag) and jitter (input noise) remain problems, especially with ray-based techniques [41]. LaViola et al. recommend minimizing the number of DOFs when considering the design of a selection device or technique [20]. Furthermore, previous work has shown that 2DOF selection can offer superior performance, even in stereo 3D virtual environments [4,43].

Based on these observations, we designed and implemented a novel selection technique we call EZCursorVR. The intended advantages of EZCursorVR include low fatigue due to no in-air control, potentially greater precision via 2DOF input devices (both in depth and lateral axes) and a shallow learning curve to start using EZCursorVR. EZCursorVR is a 2D head-coupled cursor fixed in the screen plane of the head mounted display (HMD). Unlike stationary cursors in the center of the field of view (as used with Hololens, for example), EZCursorVR can move independently using 2DOF input from any peripheral input device, employing position or rate-control mappings. Several non-VR games such as ArmA[2] use this method of aiming. Unlike most first-person shooter (FPS) games, where the mouse simultaneously controls the cursor and rotates the viewpoint, ArmA decouples these: moving the mouse controls the cursor, and viewpoint rotation begins when the cursor reaches the screen edge. Some Nintendo Wii games (e.g., GoldenEye) use a similar technique, with the remote pointing controller, effectively allowing the player to decouple view direction and selection. This effective style of interaction was our inspiration for EZCursorVR. In addition to supporting any source of

[2] https://arma3.com/

2DOF input, EZCursorVR also allows users to use their head rotation to perform selections, or a combination of both head rotation and 2DOF input. EZCursorVR's selection ray operates like a standard head selection ray, with the exception of the ability to control the ray using an external 2D (or 3D) controller in addition to head movement.

We note that EZCursorVR supports combinations of head and controller movement for selection. We believe that using a 2D controller cursor with VR HMDs will offer overall superior selection performance when compared to traditional 3D selection techniques while vastly reducing fatigue. EZCursor VR also allows the option for users to use head-based movement to get the cursor in the general vicinity of a target, and then use the mouse (or other input device) to perform fine-grained positioning.

EZCursorVR's 2D cursor implementation also allows for the ability to apply a transfer function to modify the cursor's acceleration. This may be useful by reducing the cursor's CD gain with slow movements in order to better select small perspective-scaled targets, while at the same time increasing the CD gain with fast movements for long-range ballistic motions, which is turn will benefit selection performance in VR

1.1 **Contributions**

The first contribution of this thesis is design, implementation and evaluation of a 2D head-coupled cursor for selection in VR environments. We also developed and released a framework for evaluating selection performance in 3D VR HMDs that can be used in other future studies. Our second contribution features the implementation and evaluation of several transfer functions to determine their performance when compared to the cursor's constant velocity. This is, to our knowledge, the first study involving the study of transfer functions in VR. We also evaluate the cursor's performance with head tracking on and off

to reveal the effects of head movement on the cursors performance. Participant's opinions on the different cursor and acceleration function designs are gathered as qualitative data.

1.2 Thesis Outline

The thesis is divided into five Chapters:

Chapter 1 introduces the importance as well as the disadvantages with the current VR selection techniques.

Chapter 2 provides insight and discussion regarding previous related research that has been done on 3D Selection Techniques, 2D vs 3D Selection, Fitts' Law, VR Input Devices and Control-Display Gain and Transfer Functions.

Chapter 3 describes the design and implementation of our 2DCursor, called EZCursorVR.

Chapter 4 presents our initial user study investigating the effectiveness of our 2D cursor technique using multiple input devices, in comparison to a standard 6DOF ray-based and head-based selection techniques. The experiment conformed to a previously validated 3D extension [43] of ISO 9421-9 [38] which uses Fitts' law to compare pointing devices [13].

Chapter 5 extends upon the EZCursorVR framework by implementing and testing two transfer functions that dynamically changes the gain in relation to the acceleration of the device as well as test with a constant gain level as a baseline condition and then performed a follow up user study using the same ISO 9421-9 [38] standard methodology based on Fitts' law to compare pointing device performance [13]. We present what is, to our knowledge, the first experiment on the use of CD gain and transfer functions for 2D selection in 3D environments. We also provide insight into what transfer function

parameters provide the best selection performance by testing with two sets of parameters in our transfer function.

Finally, Chapter 6 draws conclusion from both user studies and generally discusses both qualitative and quantitively results, suggests design considerations, and provides insight into future studies and improvements that can be done.

1.3 **Associated Publications**

Excerpts from this thesis are featured in the following publications [35]

1. Ramcharitar, A., & Teather, R. J. (2018). EZCursorVR: 2D selection with virtual reality head-mounted displays. In Proceedings of Graphics Interface 2018. pp.123-130. https://doi.org/10.20380/GI2018.17
2. Ramcharitar, A., & Teather, R. J. (2017). A head coupled cursor for 2D selection in virtual reality. In Proceedings of the ACM Symposium on Spatial User Interaction (SUI ’17), pp. 162–162. ACM Press. https://doi.org/10.1145/3131277.3134358

2 Chapter: Related Work

The following chapter covers research previously done on 3D Selection Techniques, 2D vs 3D Selection, Fitts' Law, VR Input Devices and Control-Display Gain and Transfer Functions.

2.1 3D Selection Techniques

There is an extensive body of literature on 3D selection techniques, dating back to the 90s. For the sake of brevity, we discuss only key studies here, and refer the reader to Argelaguet and Andujar's comprehensive 3D selection survey [1] and/or LaViola, Kruijff, McMahan, Bowman, and Poupyrev [20 , Chapter 7] for a more thorough overview.

Past studies have compared variations of direct touch [22] with ray-based techniques. Traditional ray-based techniques, although the most commonly used technique in commercial VR systems, are susceptible to hand tremor which at far distances and when selecting smaller targets yield high error rates [40]. Several methods to addressed these issues have been proposed such as the bubble cursor [14,46] and go-go [32], which were designed to support easier selection of remote or small targets by changing the style of the selection cursor. Non-traditional 3D selection techniques such as starfish (which uses a cursor with four branches that lands on nearby targets) are useful for selection in dense environments [48]. However, non-standard techniques may necessitate additional learning. In contrast, EZCursorVR should be easy to understand due to its similarity to desktop interaction – users already have extensive experience with two-dimensional cursors and can leverage their familiarity.

Previous research has also looked at progressive refinement selection interfaces. Kopper, Bacim and Bowman proposed a two tier selection process where to user first

selects a group of objects, then in multiple steps, refines the selection using a quad divided menu for increased object selection accuracy [19]. They report that this was more accurate at selecting remote objects compared to ray-casting. Similarly, our proposed technique allows combinations of 2DOF input for cursor movement with refinement via head movement (or vice versa). Unlike progressive refinement techniques, this can be done simultaneously rather than dividing the selection process into multiple steps.

Young, Teather and Mackenzie developed an IMU-based input device mounted on the users' arm to enabled 6DOF target selection via virtual hand techniques [49]. Such a device is an attractive option for use with EZCursorVR since it does not require tethering as it is largely self-contained and does not require an external tracker. Although the results show a lower error rate than optical trackers, throughput was lower and arm fatigue was very high. Fatigue is a major on-going problem with VR controllers [6,15]. Our goal with EZCursorVR was to design a control scheme that supports the kind of "lazy" interactions envisioned by Mine, Brooks and Sequin [26], using an approach to minimize physical movements (and hence fatigue) while increasing target selection throughput.

2.2 **2D vs 3D Selection**

Generally, using a 2D selection technique in VR can offer less complexity when performing selections in comparison to a 3D selection technique [20]. This is mostly attributed to 2D selection techniques having fewer number of DOFs when compared to 3D selection techniques. 2D selections are most likely to be familiar with a wider pool of users that are already accustomed to performing 2D sections with, for example, a desktop mouse or Wii controller. This drastically reduces the learnability curve and therefore allows users

to experience VR without taking an extensive amount of time to learn new controls and allowing them to jump right into experiencing VR.

Image-plane interaction is an early example of leveraging the benefits of 2D interaction in 3D spaces [30]. Like our technique, it requires only 2DOF input to select objects, but does so by lining up the hand with objects rather than explicit use of a cursor. We provide a detailed comparison between our technique and image-plane interaction in Section 3.3.

Like previous work [22,41,43] our selection task presents targets in a plane. When viewed from the starting position, this essentially "collapses" the 3D selection task into a 2D task [21]. Our selection technique is similar to that of Qian and Teather, who used a 2D eye-controlled cursor that moved within the reference-frame established by head orientation [33], however, our technique can use any 2DOF device, including the eyes to perform selections. Eye-based selection was shown to offer worse error rates and throughput than head-based selection. This is likely due to the imprecise and jittery nature of eye saccades. We expect different results, as our implementation used lower jitter controller inputs such as a joystick and mouse. Hence, we expect our results to be more in line with previous comparisons of 2D and 3D selection [44,47] which revealed 2D techniques outperformed 3D techniques [28,30].

One issue with using 2D selection cursors in stereo 3D environments is having two cursor images – double vision – due to lining up the cursor at one depth with a remote feature at a different depth. This diplopia occurs since the eyes cannot converge to the depth of the cursor and target simultaneously. The result is a "doubling" of either the target or cursor, and has been shown to influence 3D selection, more so when the depth difference

between the cursor and target is large [43]. One possible solution is to render the cursor to one eye only, but this may cause eye fatigue [36]. We instead address this by dynamically scaling and resizing the cursor according the target depths such that it always remains the same size and is rendered close to the target to avoid diplopia while also being rendered to both eyes. This approach is recommended by Unity3D tutorials on interaction in VR[3].

2.3 **Fitts' Law**

Since both our studies employs Fitts' law, we briefly describe it here. Fitts' law is a predictive model that characterizes performance of selection techniques and pointing devices, revealing the highly linear relationship between task difficulty (*ID* – index of difficulty) and selection time (*MT*). The model is given as:

$$MT = a + b \times ID \qquad (1)$$

where

$$ID = log_2\left(\frac{D}{W} + 1\right) \qquad (2)$$

D is the distance to the target and *W* is the target's size (width), while *a* and *b* are derived via linear regression. This has been formalized as a tool for testing input devices [8, 15] via ISO 9241-9 [38]. Many studies have used the ISO 9241-9 standard for comparing 2D input devices [7,28]. The standard has also been adapted for use in 3D selection tasks[39,43]. The standard prescribes the use of throughput (*TP*) as a dependent variable. Throughput is calculated as

$$TP = \frac{ID_e}{MT} \qquad (3)$$

[3] https://unity3d.com/learn/tutorials/topics/virtual-reality/interaction-vr

As per the ISO 9241-9 standard, effective *ID* (ID_e) is used to calculate throughput as:

$$ID_e = log_2\left(\frac{D_e}{W_e} + 1\right) \quad (4)$$

where

$$W_e = 4.133 \times SD_x$$

D_e is the effective amplitude and W_e is the effective target width. Effective *ID* enables direct comparison between studies with varying error rates, as it adjusts experimental error rate to 4%. The accuracy adjustment is done by calculating SD_x – the standard deviation of over/under-shoot lengths relative to the target centre, projected onto the task axis (the line between subsequent targets). It is multiplied by 4.133, which corresponds to a z-score of ±2.066 in a normal distribution, or 96% of the selection coordinates hitting the target (i.e., a 96% hit rate, or 4% error rate). It also better accounts for the task participants performed, than that which they were presented with.

2.4 **VR Input Devices**

VR input devices are just as important as the selection techniques that are associated with them. When designing new interaction techniques, it is also important to consider the device that it is going to be used with. Designing a selection technique that can work with many different input devices, as well as appealing to a wider audience (i.e. users with disabilities), can be beneficial as it allows users to use the device that they are most comfortable with.

Most traditional VR controllers include an Xbox/PS4 controller or a tracked HTC Vive/Oculus Touch controller. While this may be a more conventional approach to

controller design, several studies have explored alternative controllers for input in VR environments. TickTockRay is a smartwatch controller mounted on the users arm that uses raycasting to perform selection in smart phone VR environments [18]. While effective, this method causes fatigue as the user has to constantly move their hand to select targets. Our approach is to utilize a controller that requires very little arm movement to ensure that fatigue is reduced significantly.

Hand worn glove devices have also been explored for interacting and performing direct selections in VR [5,10,50]. Selecting objects further than the user's reach become an issue with these methods of direct selection. Ideally a selection device should support selection of both near and far targets. A recent study using an arm-mounted IMU-based 6DOF input device was developed and tested in a Fitts' law test environment [49]. Despite results showing a lower error rate, selection performance was mediocre but more importantly, many users expressed arm fatigue due to the nature of the selection where the user had to hold their arm in front of them and move it to select targets. This seems to be an ongoing issue with input devices in general [15] but increases as the number of DOF increases. In light of this, we chose to use the mouse as the input device in our studies due to the fact that it does not require the user to perform physically demanding movements.

Eye based input in VR has been shown to perform poorly likely due to inaccurate and jittery nature of eye saccades [33]. As an alternative, we use a mouse as the 2D input source for all testing to eliminate any unintended co-variables such as the unpredictable jittery nature of the eyes. Users would already be familiar with using the mouse and wouldn't require any additional training to use the input device thus we expect better results in overall performance.

2.5 Control-Display Gain and Transfer Functions

Control Display (CD) gain can offer several benefits for selection in VR. Since the user's viewpoint can move interactively in VR, CD gain is one possible option to decelerate cursor movement in order to facilitate fine positioning when selecting small and close together targets while also offering accelerated cursor movement for selecting larger and further apart targets.

Our second study employs the usage of control-display (CD) gain as a possible option to improve performance. CD gain is defined as the relationship between the amount of display movement versus the amount of controller movement [24] and can often be represented as a linear ratio as:

$$\boldsymbol{CD_{gain} = \frac{V_{pointer}}{V_{device}}}$$

where $V_{pointer}$ is the velocity of the cursor and V_{device} is the velocity of the input device respectively. If $CD_{gain} < 1$ then the cursor moves slower than the input device, but precision is increased. On the other hand, if $CD_{gain} > 1$ then the cursor moves faster than the mouse, therefore less clutching is required but precision suffers. This creates a speed-accuracy tradeoff which was first pointed out by Jenkens and Connor [17] and shown in Figure 1.

Mackenzie and Riddersma tested the expectation that the optimal gain setting would be at the intersection of both lines [37]. Their results suggest that there was no optimal CD gain setting as error rates were highest when using the expected optimal setting.

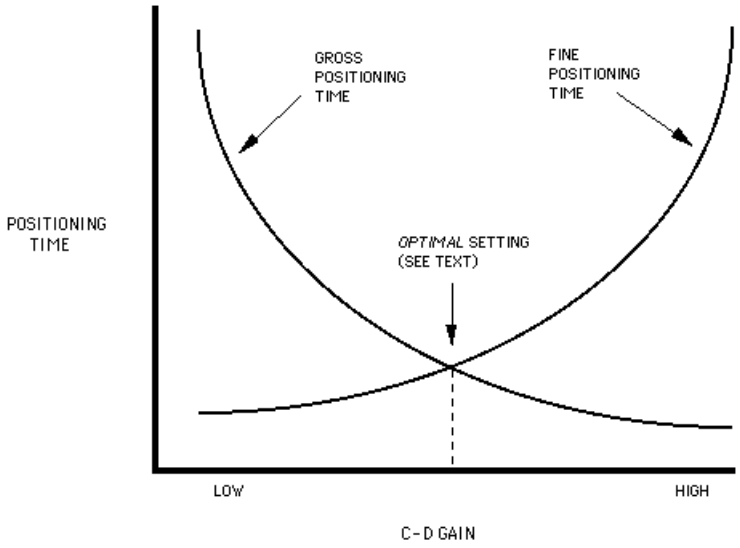


Figure 1: The relationship between CD gain and selection time. Fine positioning becomes more difficult with high gain (due to overshooting targets) while coarse positioning takes longer with low gain due to excessive clutching. Figure reproduced from Jenkens and Connor [17]

Based on the results of previous research on CD gain, [37] we only included one constant CD gain condition in our experiment as a baseline of comparison. Additionally, we tested the implementation of several pointer transfer (or acceleration) functions where the CD gain isn't constant but changes as a function of the speed of the device. The faster the device is moved, the faster the cursor moves while slow movements causes CD gain to decrease. Acceleration functions have been shown to be more effective when compared to constant CD gain [8,9] not only for standard mouse pointing tasks but also for large scale input using in air gestures for pointing [27]. These acceleration functions can be seen in Figure 2. Some arm-extension interaction techniques such as Go-Go [32] rely on a similar concept in VR, by employing a non-linear acceleration on the users virtual hand, once the real hand moves beyond a certain distance threshold. Similar to Go-Go, our study was aimed to measure selection performance using acceleration functions but using a 2D selection technique instead of a 3D selection technique.

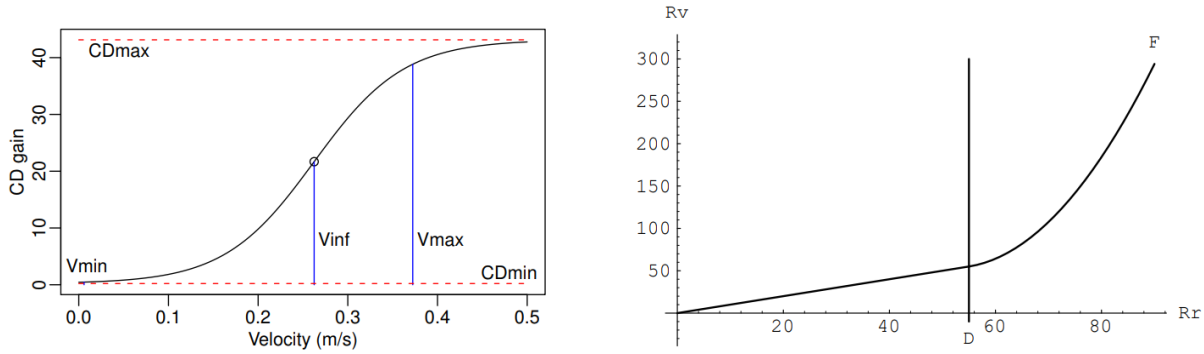


Figure 2: Curves showing two different acceleration functions. Left: Nancel 2015 [27]**, Right: Poupyrev 1996** [32]**.**

2.6 Summary

Previous research has shown that 2D selection as well as the use of transfer functions increases selection performance while reducing the effects of the speed accuracy tradeoff. However, previous research has only been conducted on 2D and 3D desktop environments and therefore worth exploring the question of whether the benefit of using 2D controllers and transfer functions carry over to HMD based VR scenarios. To our knowledge, our studies represent the first evaluation of 2D pointing selection performance in a VR head-mounted display, as well as the first application of control-display gain in such scenarios.

3 Chapter: The 2D Cursor – EZCursorVR

Selection in VR typically involves two aspects: the interaction technique itself (i.e., the software part), and the input device (i.e., the hardware part). Example *interaction techniques* include ray-casting, Poupyrev's go-go technique [32], and direct touch with the hand. Common VR *input devices* include wands, such as those provided with the HTC Vive and Oculus Rift, joysticks (e.g., on game controllers) and even the mouse can be used. LaViola et al. [20] point out that interaction technique and input device are separable – an input device can support multiple different interaction techniques, and vice versa. Consider, for example, that ray-casting (an interaction technique) is supported by both 3D trackers and the mouse (input devices). Likewise, 3D trackers support both ray-casting and direct touch metaphor interaction techniques. Both components are important considerations when performing selections in VR, and it is desirable when designing new interaction techniques that they can work with multiple different input devices. After all, not all users have access to the same equipment.

Based on previous research, we formulated several design considerations and goals when creating EZCursorVR:

1. Supporting 2DOF input sources.
2. Being able to operate the cursor in 2D (eliminating depth).
3. If desired, allowing selection with the head.
4. Keeping the cursor size constant despite object depth.
5. Minimizing the learning curve for new users.

The following is a detailed explanation on the design and implementation of our cursor. Like screen-based techniques [44] EZCursorVR uses ray-casting and relies on the

concept of image plane selection [30]. From the user's perspective, they appear to select targets using a 2D cursor to overlap the 2D "screen-space" projection of targets. The plane the cursor resides in appears to be fixed to the head. In other words, rotating or moving the head also results in movement of the cursor, although the cursor position itself appears fixed in this plane. This is depicted in Figure 3. Unlike classical image-plane interaction [30], which allows the user to line up their hand with virtual objects for selection, our technique instead does this indirectly, via the use of an external controller to control the position of a selection cursor, similar to desktop environments.

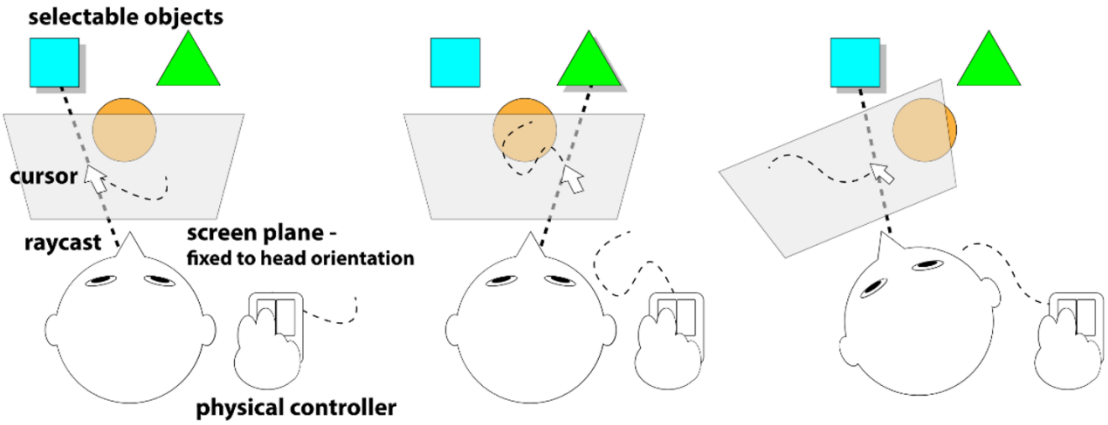


Figure 3: Movement of EZCursorVR. Head-movement and rotation influences the position of the plane-fixed cursor. The cursor can be independently controlled by an external input device (e.g., a mouse, in this example, although other sources of 2DOF or even 6DOF are supported.)

In actuality, the *rendered* cursor is displayed in world-space at the intersection point of a ray originating at the head (the camera in Figure 4) and directed towards an invisible *control* cursor that moves in a head-coupled plane (#1 in Figure 4). The control cursor is constrained to move from one extent of the user's field of view to the other. The ray from the head to the control cursor is used to determine which object is selected, and where to position the rendered cursor (#2 in Figure 4).

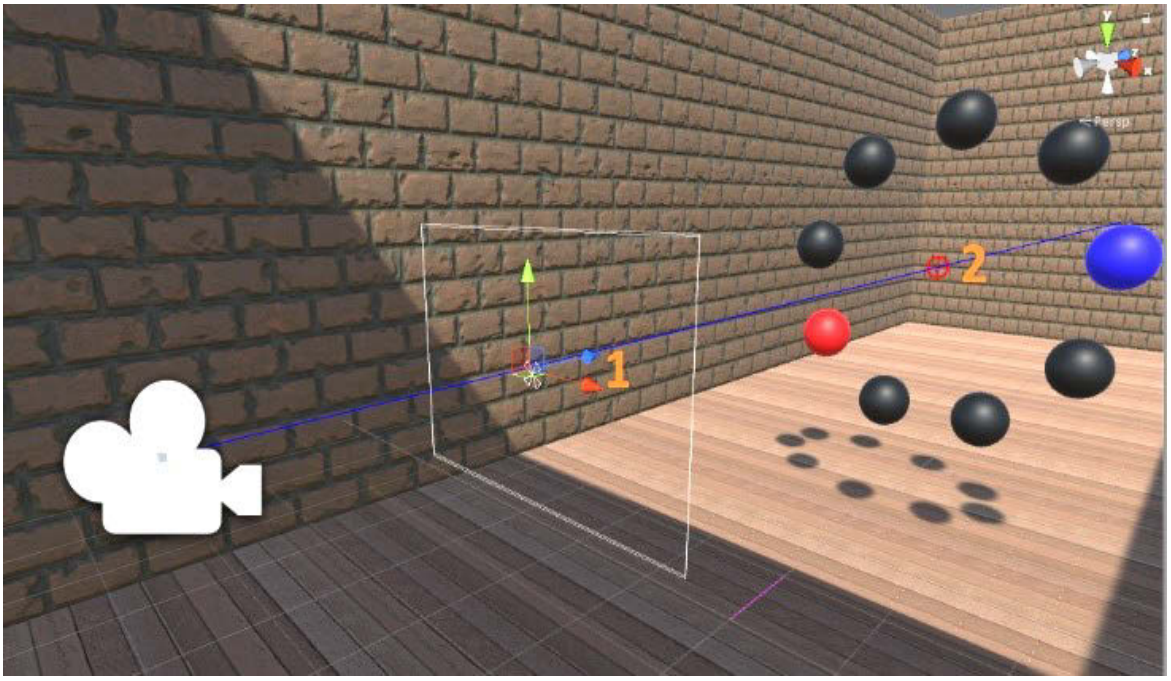


Figure 4: The invisible control cursor (#1) that moves in the head-coupled plane, and the visible rendered cursor (#2).

3.1 Cursor Rendering

Although our intent is to support 2D selection in 3D spaces, simply rendering the control cursor fixed in a head-coupled plane would introduce the double-vision problem detailed earlier [43]. We address this problem by instead displaying the *rendered* cursor (#2 in Figure 4) as an object in the scene. The control cursor is not displayed at all. The rendered cursor is displayed at the correct depth, as determined by ray-casting, using the ray depicted in Figure 4, originating at the eye/head position, and directed through the control cursor. The rendered cursor is drawn at the intersection point with the scene. We then scale the rendered cursor to cancel out the scaling effect of perspective. As a result, the rendered cursor appears consistent in size regardless of its depth. We also render it as a billboard, so it is always oriented towards the viewer. The end result is that the rendered

cursor appears to operate in 2D, but its stereo depth is correct for any point in the scene, eliminating double-vision effects [43].

3.2 Input Sources

Since the *control* cursor resides in a plane, 2DOF input sources can readily control its movement through simple mappings. For example, from the default screen-centre position, mouse displacement can map to control cursor displacement (subject to a gain function). Similarly, joysticks can be used in both velocity- and position-control mappings. Changes in the position of the control cursor are reflected in changes to that of the rendered cursor, via ray-casting as described above. Due to cancelling out perspective, the rendered cursor appears to move in 2D, but with correct stereo depth.

For our study, we have also implemented a technique that uses a 6DOF input source to control the cursor. In our case, the user points a tracked wand at the head-coupled plane. The wand-ray/plane intersection point is used for the position of the control cursor. This is similar to the ray-screen technique demonstrated in previous work [44], which in turn, is similar to how remote pointing works with the Nintendo Wii remote.

3.3 Comparison with Image-Plane Selection

Our technique is similar to image-plane selection introduced by Pierce and Forsberg [30]. The 2D plane for our technique is a head-coupled plane that moves along with the user's head rotation to remain parallel to the user's FOV. Our technique most closely resembles the 'Sticky Finger' technique where a user can select objects by aligning an outstretched finger with the target. In contrast, we replace direct interaction with a 2D controlled cursor.

Our technique is different from image-plane selection in two key ways. First, with image-plane selection, the user must outstretch their arms to point at or frame targets. This in-air interaction causes extreme fatigue after extended use, leading to the well-known "gorilla-arm syndrome" [15]. Our technique avoids this by using 2D selection devices, which necessitate less effort and thus reduce fatigue. Second, with image-plane selection, movement is mapped 1:1. In contrast, EZCursorVR offers the ability to apply control-display (CD) gain to cursor movement. While this tends not to be available with 1:1 VRselection techniques (e.g., ray-casting), we argue that gain could help with 2DOF control. Consider, for example, that remote targets perspective scale to be smaller and in accordance with Fitts' law, harder to select. Remote targets are difficult to select with rays[31], but with EZCursorVR, slow 2D movement (e.g., with a mouse) could be further decelerated by lowering CD gain, enabling precise selection of small targets. Similarly, gain could be increased for long-range ballistic movements, enabling fast crossing of the screen for far away targets.

4 Chapter: User Study 1 - 2D Cursor for VR

In this chapter, we present our first study evaluating EZCursorVR with several input techniques including the mouse, thumbstick, and motion controller.

4.1 Hypotheses

The main hypotheses of our first user study were:

H1: Performance with 2D techniques will be higher than 3D techniques, as found in prior research [47].

H2: The mouse will perform best, followed by Ray2D, Velocity-Joystick, Head-only, and finally Position-Joystick. This ranking is based on our own pilot testing and intuition, as well as previous studies that used similar input methods [25,28,33].

***H3*:** Throughput will be consistent across target depth using EZCursorVR , but will vary with depth using the standard ray, as found in previous research [44].

4.2 Participants

Our study included 18 participants (15 male, 3 female, aged 18-44 years) recruited from the local community. We gave participants a pre-test questionnaire asking about their familiarity with VR. 12 participants had previous exposure to VR.

4.3 Apparatus

4.3.1 Hardware

The experiment was conducted on a VR-ready laptop with an Intel core i7-7700HQ quad core processor, a Nvidia Geforce 1070 GPU, and 16GB of RAM, running the latest build of Microsoft Windows 10. We used an Oculus Rift CV1 head-mounted display, connected to the computer via HDMI. The CV1 features a resolution of 1080 x 1200 per eye, a 90Hz refresh rate and a 110° field of view.

Participants were seated in our lab, positioned far enough away from obstacles to ensure there was no chance of hitting anything. Depending on the experimental condition, participants either used a mouse, an Oculus Touch controller, or the HMD itself (via head tracking) as an input device. The Oculus Touch controller features real-time motion tracking, a thumb joystick, two trigger buttons, and vibrotactile feedback and was used for several different input methods in our experiment. See Figure 5.

Figure 5: Participant wearing the Oculus Rift using the touch controllers. Inset: close-up of Oculus Touch controllers.

4.3.2 Software

Our test environment was created in Unity with the purpose to test device selection performance in VR HMDs and using external libraries for the Oculus Rift hardware. The test environment was based on ISO 9241-9 standard reciprocal selection task (Figure 6).

Each round consisted of 9 spherical targets, presented in one of three different sizes, at three different distances apart from each other. Each ring of targets was presented at one of three different depths from the user. Within a round, target size, distance, and depth were held constant.

Targets were shown in four different colours:

Green: Indicates targets that were already hit.

Red: Indicates targets that were previously missed.

Blue: Indicate the "active" target that the user is supposed to select.

Black: Indicate targets that are not yet active.

The software automatically logged performance data, such as selection times, error rates, and calculated throughput as described in Equation (4).

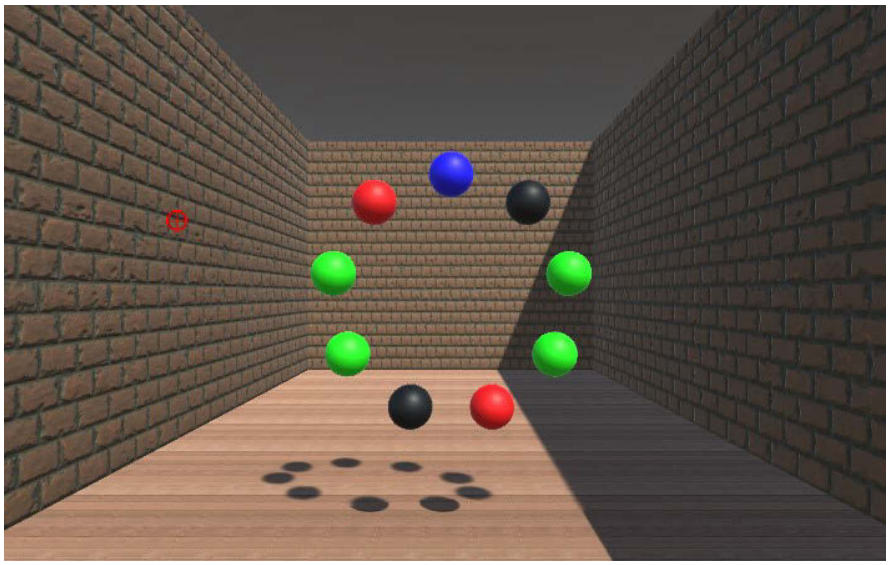


Figure 6: Fitts' law test environment in Unity. The red cursor depicts the position of the head-coupled cursor, as described in Section 3.1.1

4.3.3 Controllers

Our study included 6 input-device/interaction-technique combinations, which we refer to as "controllers". All controllers, except for Ray3D use EZCursorVR. We describe

these, and their effect on the control cursor (noting that the effect on the rendered cursor is implied, as described in Chapter 3) as follows:

Mouse: The control cursor is controlled by the mouse using a direct mapping of the mouse's x and y movement.

Head: The control cursor was fixed in the center of the field of view, and thus can only controlled by the user's head gaze. This was intended as a commonly used baseline condition (i.e., EZCursorVR was disabled) to assess the added value of independent cursor control.

Velocity-Joystick: The control cursor is controlled by the joystick on the Oculus Touch controller and moves at a constant velocity in the direction the user pushes on the joystick.

Position-Joystick: The control cursor is controlled by the joystick on the Oculus Touch controller but uses a position-control mapping. It thus moves depending on the location the joystick is pushed – i.e., pushing the joystick moves the cursor to the corresponding position across the field of view. When the user is not pushing the joystick, the control cursor returns to the center position.

Ray2D: The control cursor position is determined by the intersection of the head-coupled plane and the 6DOF ray from the Oculus Touch controller. In other words, the user points the controller at the plane to control the cursor position, rather than at objects themselves.

Ray3D: The user controls a standard 6DOF ray using the Oculus Touch controller, necessitating selection by pointing at the target volumes (rather than their projection). This

was intended as another baseline condition, as the most typical interaction technique used with 6DOF-tracked wands in modern VR games.

4.4 **Procedure**

Upon arrival, we asked participants to answer a pre-experimental questionnaire about their familiarity with various VR input devices and any previous experiences in VR. They were then shown how to use each of the controllers and how the target selection task worked. They were given a practice round to familiarize themselves with the hardware and software. Data gathered from these practice trials were excluded from our analysis. After participants performed several practice trials and indicated that they were comfortable using the hardware and software, they were then asked to perform the actual experiment. Their instructions were to select the highlighted target as quickly as possible and as close as possible to the center. Upon pressing the selection button, the trial advanced to the next target (which turned blue, indicating it was the "active" target) regardless if the selection hit or missed. Upon finishing a round (9 targets) a new combination of target width, distance, and depth was randomly picked (without replacement). The experiment ended after the participant completed all combinations of distance, width, and depth, with each controller.

After completing the experiment, we gave participants another questionnaire that asked them to evaluate their preference toward each controller. We also asked them to rank their preferred controller from best to worst. Finally, they were debriefed and were given $10 compensation for their time. The entire experiment took roughly 1 hour.

4.5 **Design**

Our experiment employed a within-subjects design with a single independent variable, controller, with 6 levels: Mouse, Head, Position-Joystick, Velocity-Joystick, Ray2D and Ray3D. Controller ordering was counterbalanced according to a balanced Latin square to offset learning effects.

Each participant completed a total of 9 trials per round × 6 controllers × 3 distances × 3 widths × 3 depths = 1458 trials, or 26244 trials over all 18 participants. The combinations of distance and width produced 9 indices of difficulty, ranging from 1.2 bits to 3.7 bits. These were not analyzed, but rather used to produce a realistic range of task difficulties.

Our experiment included 3 dependent variables: Throughput (bits/sec, calculated as described earlier), error rate (percentage of missed targets), and movement time (in milliseconds). Movement time was calculated as the difference in time from selection of target n to target n+1.

4.6 **Results**

4.6.1 Throughput

Results for throughput are shown in Figure 7. A Shapiro-Wilk test using right tailed distribution indicated that the data is normally distributed ($p = 0.05$). Repeated measures ANOVA revealed that the main effect of controller on throughput was statistically significant ($F_{5,85} = 68.74$, $p < 0.0001$), as was the main effect for depth ($F_{2,34} = 48.09$, $p < 0.0001$). The controller × depth interaction effect was also statistically significant ($F_{10,170} = 6.87$, $p < 0.0001$). The Scheffe posthoc test indicated that most pairs of controllers were significantly different ($p < .05$). These pairwise differences are also seen in Figure 7. Average throughput with the mouse was lower (around 2.66 bps) than those of the other

3D studies that have reported mouse throughput of around 3.7 bits/sec [28]. This may be because the cursor was controlled by both the head and the mouse, and head movements may have adversely affected the throughput. Previous studies did not use head-coupled cursor planes.

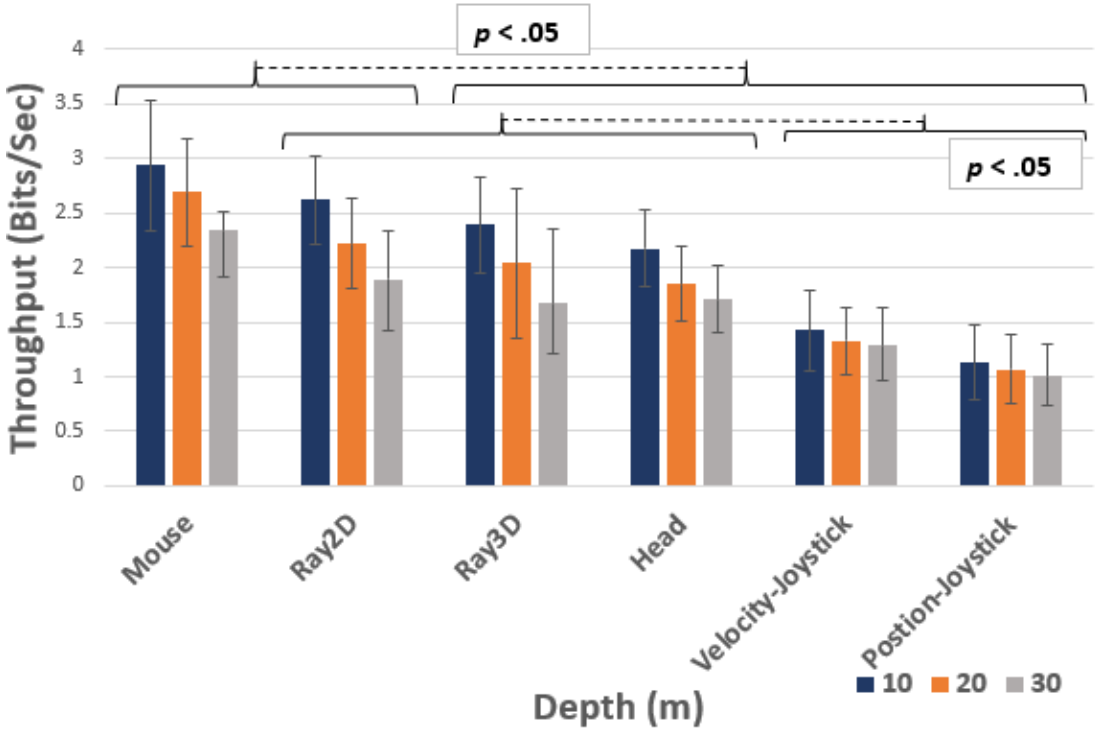


Figure 7: Throughput by Depth. Error bars show ±1 *SD*. Statistical groups (i.e., controllers that are not significantly different) are indicated with curly braces, with dashed lines showing significant differences to other groups via the Scheffe test.

4.6.2 Movement Time

Results for movement time are shown in Figure 8. A Shapiro-Wilk test using right tailed distribution indicated that the data is normally distributed ($p = 0.94$). Repeated-measures ANOVA revealed that the main effect of controller on movement time was statistically significant ($F_{5,85} = 36.63$, $p < 0.0001$) as was the main effect on depth ($F_{2,34} = 8.48$, $p < 0.005$). The controller × depth interaction effect was not statistically significant

($F_{10,170}$ = 1.21, p>0.5). The Scheffe posthoc test revealed many pairwise differences between the controller types (p < .05) – all of the Mouse, Ray2D, Ray3D, and Head controllers had significantly faster movement times than the two joystick-based controllers. These are seen in Figure 8.

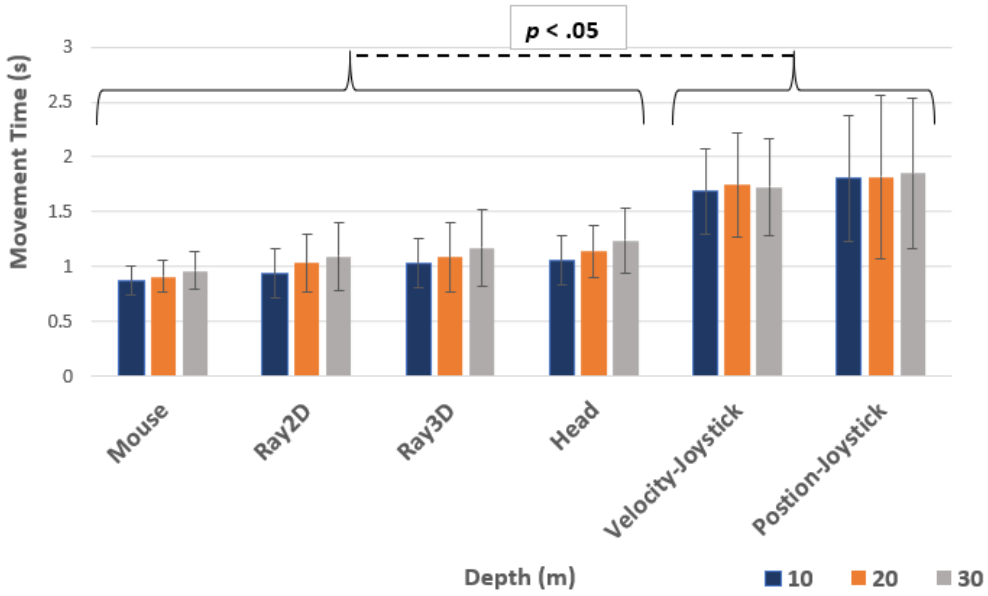


Figure 8: Movement time by controller and depth. Error bars show ±1 SD.

4.6.3 Error Rate

Results for error rate are seen in Figure 9. A Shapiro-Wilk test using right tailed distribution indicated that the data is normally distributed (p = 0.51). Repeated-measures ANOVA revealed that the main effect of controller on error rate was statistically significant ($F_{5,85}$ = 20.43, p < 0.0001) as was the main effect on depth ($F_{2,34}$ = 224.62, p < 0.001). The controller × depth interaction effect was statistically significant ($F_{10,170}$ = 7.43, p < 0.001). The Scheffe post hoc test revealed four pair-wise significant differences (p < .05), seen in Figure 9.

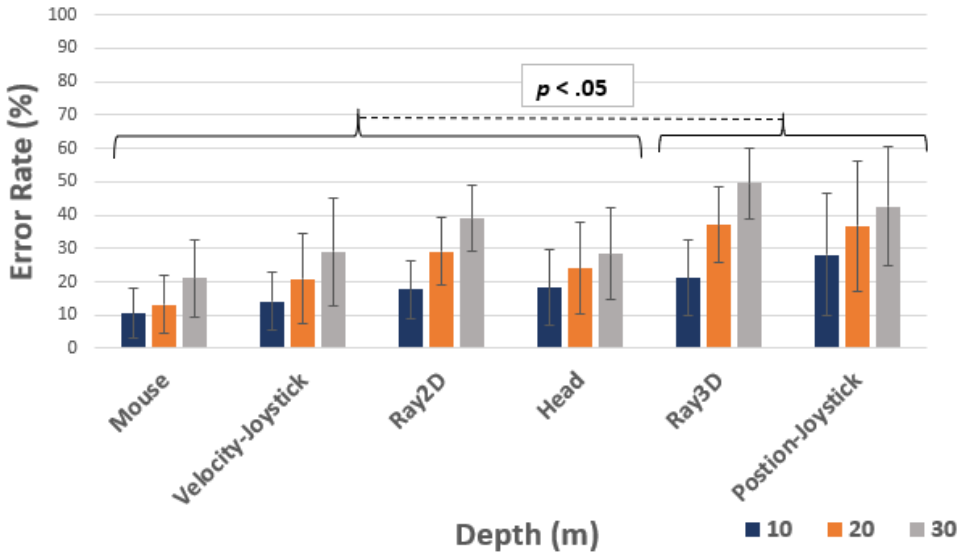


Figure 9: Error rate by controller and depth. Error bars show ±1 *SD*.

4.6.4 Qualitative

Participants rated the various control schemes out of 5 on a Likert scale on accuracy, fatigue and speed. Results are shown in Figure 10. Friedman non-parametric test shows that there was a difference in quality of results across accuracy, fatigue and speed reporting (χ^2= 59.620, $p < 0.0005$, $df = 5$), ($\chi^2 = 16.929$, $p < 0.005$, df = 5) and ($\chi^2 = 42.074$, $p < 0.0005$, $df = 5$) respectively. Vertical bars (●—●) show pairwise significance.

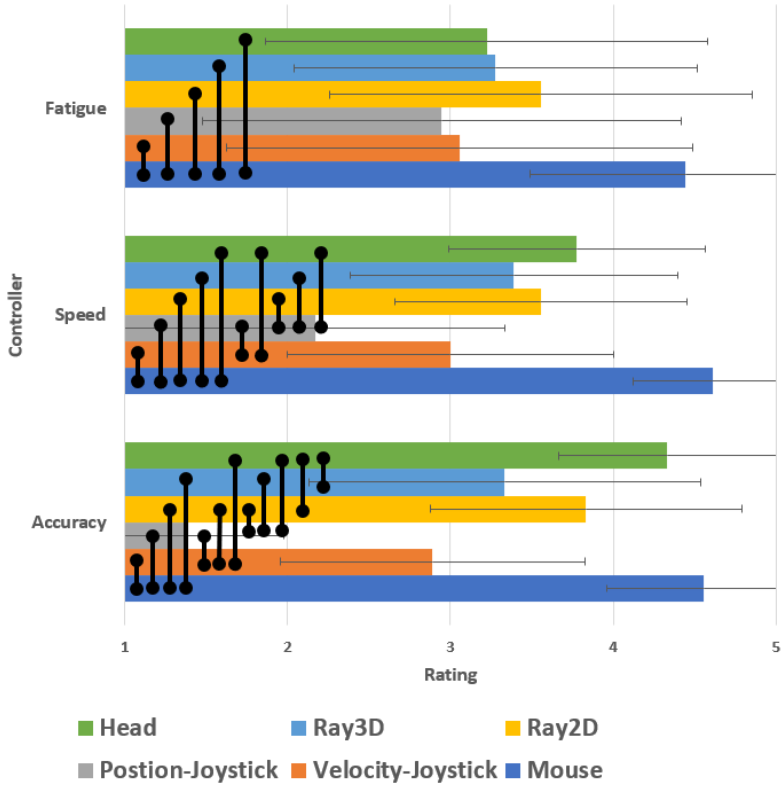


Figure 10: Qualitative Results for controller Fatigue, Speed and Accuracy. Error bars show ±1 SD. Higher values are better.

4.7 Discussion

Overall, the mouse outperformed the other controllers. This was expected based on previous work, and as the mouse was most familiar controller. However, using the mouse with EZCursorVR yielded worse performance than in previous work in non-head-tracked stereo 3D environments. Although we anticipated a larger difference, this result still validates the basic concept of EZCursorVR – the technique offered better performance than other common VR selection techniques, notably rays controlled by either a wand or the head.

Hypothesis H1, that 2DOF devices would perform better than 3/6DOF devices, was partly confirmed. The mouse Ray2D were the two top performers. EZCursorVR worked

well with some of the controller input devices. On the other hand, both joystick-based controllers performed very poorly. This suggests that the performance of EZCursorVR is highly dependent on the actual input device it is used with. Future work will investigate this further.

Similarly, hypothesis H2 was partially confirmed as well. While the mouse and Ray2D did outperform the other controller schemes, velocity-joystick did not perform as well as expected. The poor performance of the velocity-joystick may be attributable to the constant cursor speed. This restricted participant control over cursor acceleration, resulting in frequent overshooting of targets. This may highlight an opportunity to use CD gain, or a more complex transfer function to potentially improve joystick performance. Position-joystick also offered very low performance. This can likely be attributed to the high sensitivity of the cursor, and the fact that participants were unfamiliar with position-controlled cursors in general.

As expected the mouse had the lowest error rate. Both ray controllers as well as the head only had lower error rates compared to both joystick controller schemes. We attribute this to the abstract and unnatural pointing nature of the joysticks as opposed to a more natural feeling, ‘look to select’ or ‘point to select’ methods of the ray and head only controllers, especially for far or small targets where a controller that can fine tune the movement of the cursor would result in lower error rates.

Participants experienced some difficulty selecting remote targets with the ray-based techniques, as we had anticipated. With Ray3D, selecting remote targets was difficult, due to their smaller angular size. As a result, most participants preferred Ray2D over Ray3D, especially when selecting far targets. Additionally, participant hand tremor, while small,

propagated up the visible ray in Ray3D causing it to sway substantially, also making it difficult for selecting far targets. Although Ray2D also used a ray, this swaying was reduced due to the comparatively short distance to the head-coupled plane. This likely explains the easier time participants had with Ray2D.

Surprisingly, H3 was not found to be true, despite previous evidence [44] that suggests throughput calculated in the plane should be constant over depth. There are two possible reasons for this. First, we used more extreme depth differences than in previous work, which was constrained to a depth range of about 28 cm. In contrast, our depth range was 30 m. Another factor is that head motion influenced our techniques, unlike in previous work. In our study, the head was a constant source of potential input noise, as it was the origin of all rays. These two factors, taken together, may have yielded this result, and may speak to a limitation of our technique and/or a need to reinvestigate projected throughput.

The coupling of head with the cursor movement proved valuable for both joystick controllers as participants. Participants were observed using a combination of head and joystick movement and confirmed this in post-experiment debriefing. Several participants noted they used the joystick for coarse motions, and then "fine tuned" their selection via head movement. This made it easier to select small or high distance targets (although opposite to how we initially expected). Similarly, some participants expressed interest in being able to switch between the head only and EZCursorVR while performing selections (i.e., toggling independent cursor movement on and off). This is a topic for a future study and will allow us to definitively determine if participants actually use the two control styles (head + controller) together or independently.

4.8 **Summary**

In the first experiment, both joystick controllers moved the control cursor linearly, adjusted by a scale factor. Adding a dynamic and potentially non-linear gain function to provide cursor acceleration could improve joystick performance, and perhaps even the mouse condition. Consider, for instance, the difficulty participants had in selecting remote targets. Remote targets perspective scale to be smaller and hence harder to select targets. A gain function that reduced gain with slow movements might make these easier to select. We explore non-linear gain functions for our follow-up study in Chapter 5, as well as measure the difference in performance with head tracking on vs headtracking off by isolating the head movement from the controller.

5 Chapter: User Study 2 - 2D Cursor Acceleration

In this chapter, we present our second study evaluating EZCursorVR with several different distinct transfer functions including an inverted, sigmoid and constant function as a baseline.

5.1 Hypotheses

The use of constant CD gain levels has previously been shown to have no difference in 2D selection performance [37]; non-constant transfer functions have shown greater promise [8,27]. However, VR selection may benefit more, due to the scaling effect of perspective in 3D; selecting remote (i.e., small) targets may benefit from decreased gain. We therefore hypothesize that the use of transfer functions will increase performance when compare to constant gain. Finally, we also provide insight into what transfer function parameters provide the best selection performance by testing with two sets of parameters in our transfer function.

5.2 Participants

Our study included 15 participants (10 male, 5 female, aged 19-31 years) recruited from the local community. We gave participants a pre-test questionnaire asking about their familiarity with VR. All 15 participants had previous exposure to VR.

5.3 Apparatus

5.3.1 Hardware

The experiment was conducted on a VR-ready laptop with an Intel core i7-7700HQ quad core processor, a Nvidia Geforce 1070 GPU, and 16GB of RAM, running Microsoft Windows 10. We used an Oculus Rift CV1 head-mounted display, connected to the laptop

via HDMI. The CV1 features a resolution of 1080 x 1200 per eye, a 90Hz refresh rate and a 110° field of view.

5.3.2 Software

We used the same 3D Fitts' law framework developed for the previous study[4]. The test environment is based on the ISO 9241-9 reciprocal selection task. Each round had 9 spherically shaped targets varying in one of three different sizes, distances from each other and depths from the user. Target size, distance and depth were held constant within a specific round. Target colours were the same as described in section 4.2.2. The software automatically logged data on performance such as throughput, selection times and error rates.

5.3.3 Transfer Functions

Three transfer functions were designed and implemented to resemble functions used in previous research [8,27]

Constant (Green) is a constant 1:1 gain function

$$f(x) = x$$

Where x is the fixed slope of the line.

Shallow (Blue) is modeled after the absolute value of an inverted function.

$$f(x) = \frac{1}{l} \times x \times (x + t)$$

Where l represents the linearity (the curvature of the line) and t represents the intersection (the point on the x axis where the curvature of the line crosses the constant line. A higher

[4] https://github.com/adrianramcharitar/UnityFittsLawVR

value yields a more linear curve and thus lower levels of cursor acceleration at high device movement speeds.

Steep (Red) is modeled after a Sigmoid function.

$$f(x) = \left(\frac{k}{1 + e^{(a+bx)}}\right)$$

Where a represents the phase shift, b represents the steepness and k represents the amplitude of the function. We define each of these terms below.

Phase shift: The horizontal shift of the function.

Steepness: The magnitude of the cursor acceleration, higher values equal a faster acceleration.

Amplitude: The maximum acceleration threshold of the cursor until the speed is constant.

All three control functions were implemented in Unity by first getting the delta movement for the mouse, then applying the Pythagorean theorem to the change in movement of the x and y axis to get diagonal change distance. This value was then substituted into each of the transfer functions and the cursor was then displace by the resulting value. Functions were implemented using Unity's Math.f[5] class. These functions are all integrated as part of the Fitts' law framework and are show in Figure 11.

[5] https://docs.unity3d.com/ScriptReference/Mathf.html

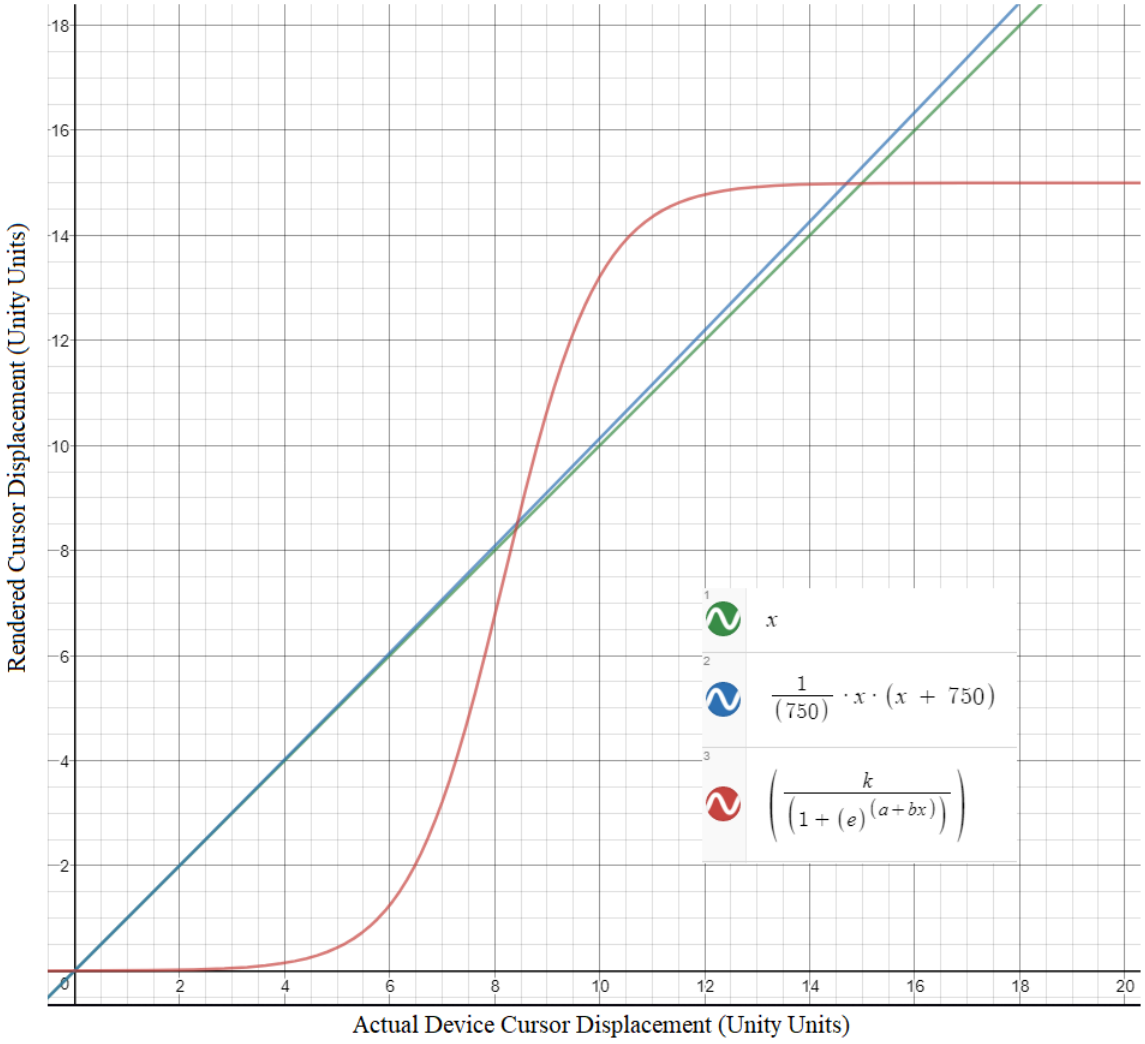


Figure 11: Graph of both Transfer acceleration functions used along with constant acceleration as a baseline (Green: Constant, Blue: Shallow, Red: Steep).

5.4 Procedure

Upon arrival, participants completed a questionnaire on their familiarity with VR input devices. Participants were then shown how to perform the selection task and were given a practice round to familiarize themselves with the hardware and software. Data gathered from the practice trials were excluded from analysis. They were then asked to perform the actual experiment and were instructed to select the highlighted target as quickly and as close to the center as possible. As participants pressed the select button, the

trial advanced to the next target, turning blue to indicate the next target to select. After each round was finished (9 targets), a new random (without replacement) combination of target width, distance and depth was presented to the participant. The experiment ended after the participant completed all combinations of distance, width and depth with each gain level/transfer function. At the end of the experiment, participants were asked to fill out another questionnaire as well as a post experimental structured interview, so we could gather quantitative data about features such as gain level and head tracking preference.

Upon completion of the experiment, participants were debriefed and given $10 compensation for their time. The experiment took approximately 1 hour to complete.

5.5 **Design**

Our experiment used a 2 × 3 (Head tracking × Control function) within-subjects design with the following variables and levels:

Control function: Constant, Steep Transfer Function, Shallow Transfer Function

Head tracking: On, Off

Width: 0.75, 0.5, 0.25 m

Distance: 1, 2, 3 m

Depth: 10, 20, 30 m

Each participant completed 9 trials per round × 3 distances × 3 widths × 3 depths × 3 control functions × 2 head tracking = 1458 trials, or 21,870 trials over all 15 participants. The combinations of distance and width produced 9 indices of difficulties, ranging from 1.2 bits to 3.7 bits. Width, distance and the resulting *ID*s were not analyzed, but used to produce a realistic range of task difficulty.

Our experiment included 3 dependent variables:

Throughput: Measured as Bits/Sec (Calculated as described in Chapter 2)

Error Rate: Measured as percentage of missed targets in a round

Movement Time: Measured as average time to select targets, in milliseconds

Movement time was calculated as the delta time between target n and n+1.

5.6 **Results**

5.6.1 Throughput

Results for throughput are shown in Figure 12. A Shapiro-Wilk test using right tailed distribution indicated that the data is normally distributed (p = 0.99). Repeated measures ANOVA revealed that the main effects for head tracking and control function on throughput were both statistically significant ($F_{1,11} = 14.168, p < 0.005$) and ($F_{2,22} = 92.596, p < 0.0001$) respectively, as was the main effect for depth ($F_{2,22} = 20.030, p < 0.0001$). The head tracking × depth interaction effect was also statistically significant ($F_{2,22} = 17.063, p < 0.0001$). The 3-way interaction effects were not significant ($F_{4,44} = 0.870, p > 0.05$). The Scheffe posthoc test indicated that most pairs of control functions were significantly different ($p < .05$). These pairwise differences between transfer functions are also depicted in Figure 12 as horizontal bars (●—●).

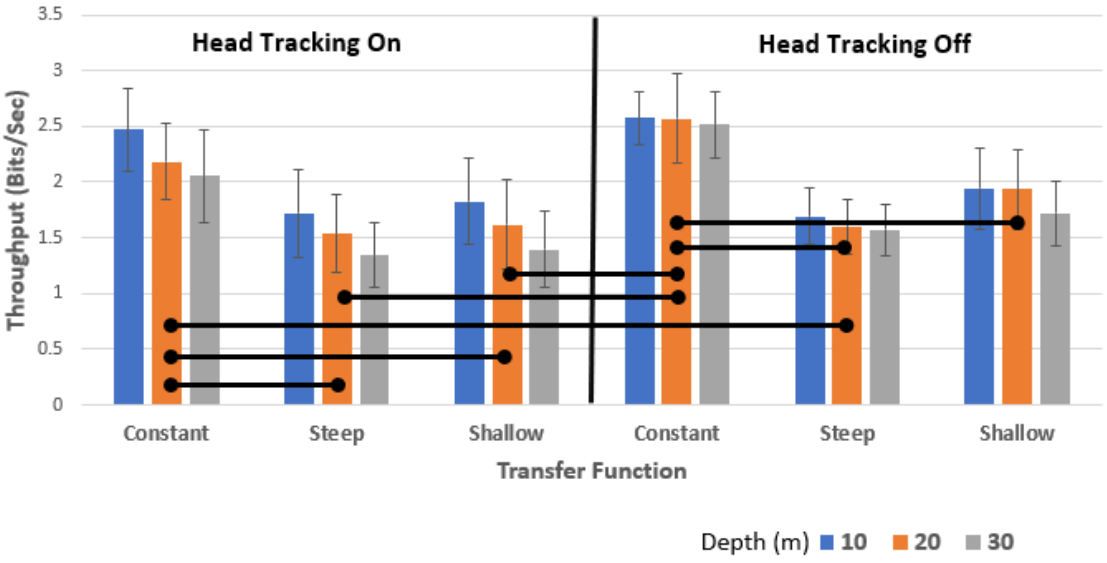


Figure 12: Throughput on Depth. Error bars show ±1 SD.

5.6.2 Movement Time

Results for movement time are shown in Figure 13. A Shapiro-Wilk test using right tailed distribution indicated that the data is normally distributed (p = 0.93). Repeated measures ANOVA revealed that the main effect of control function on movement time was statistically significant ($F_{2,22}$ = 89.911, p < 0.0001), as was the main effect for depth ($F_{2,22}$ = 4.304, p < 0.05). Both the head tracking × depth and control function × depth interaction effects were also statistically significant ($F_{2,22}$ = 13.146, p < 0.0005) and ($F_{4,44}$ = 0.193, p < 0.0001) respectively. 3-way interaction effects were not significant ($F_{4,44}$ = 0.870, p > 0.05). The Scheffe posthoc test indicated that several pairs of control functions were significantly different (p < .05). These pairwise differences are depicted in Figure 13 as horizontal bars (●—●).

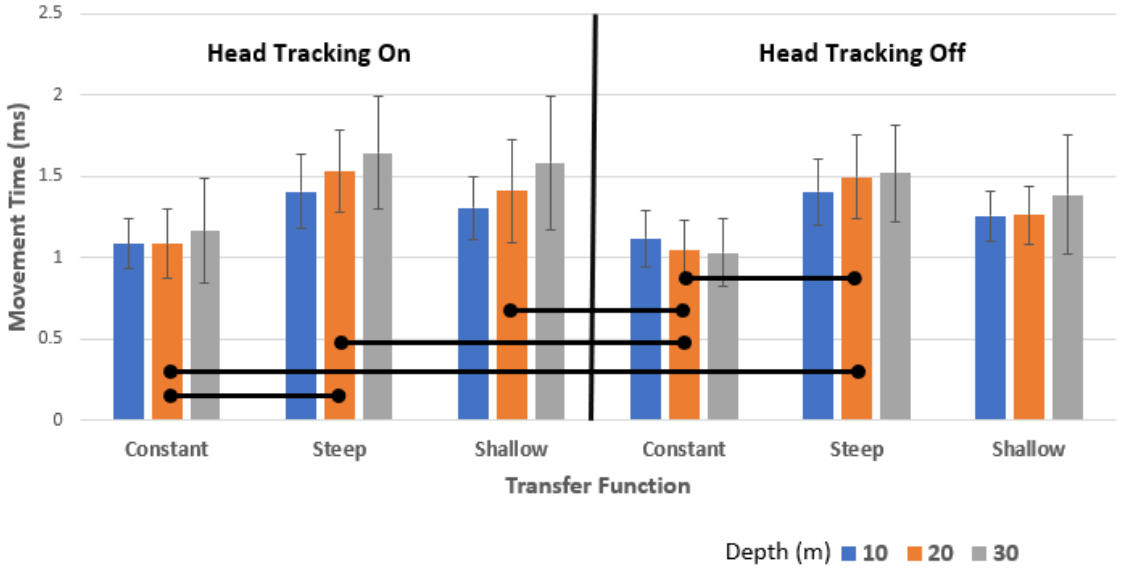


Figure 13: Movement Time on Depth. Error bars show ±1 *SD*. Note: bars show significant pairwise differences between transfer functions, but not depths.

5.6.3 Error Rate

Results for error rate are shown in Figure 14. A Shapiro-Wilk test using right tailed distribution indicated that the data is not normally distributed ($p = 0.001$). Friedman non-parametric revealed a significant difference in the quality-of-result assessments between the six conditions ($\chi^2 = 34.356$, $p < 0.001$, $df = 5$). . However, the Scheffe posthoc test failed to detect any significant pairwise differences. The 3-way interaction effect was not significant ($F_{4,44} = 1.423$, $p > 0.05$).

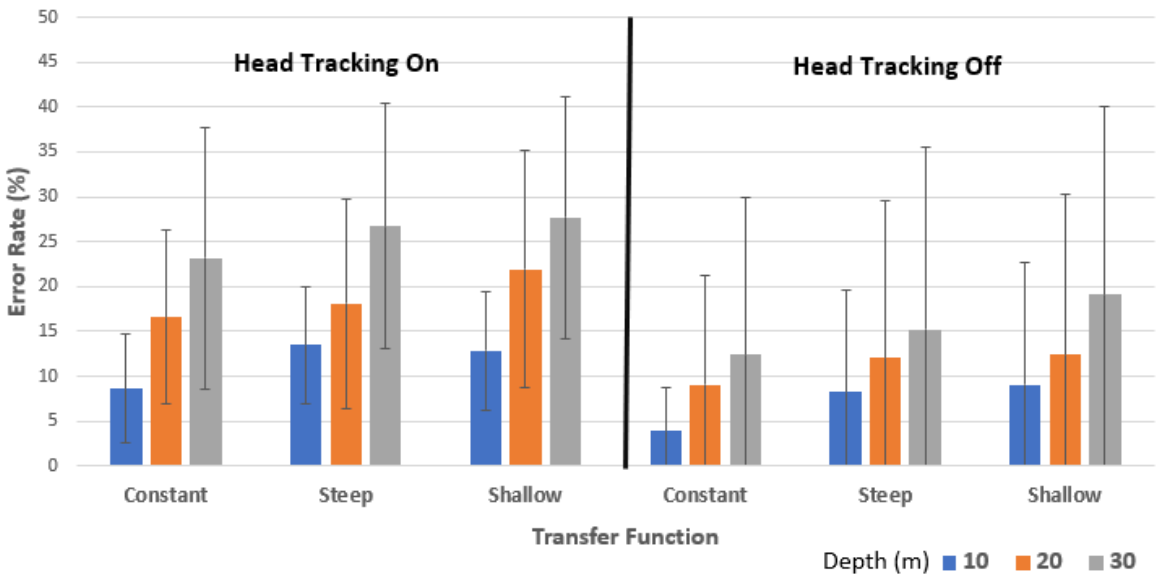


Figure 14: Error Rate on Depth. Error bars show ±1 SD.

5.7 Discussion

Contrary to previous research [27,37] our results indicate that constant velocity performs better than both transfer functions for throughput, movement time and error rate. We believe that this is attributed to our experimental design where we tested each transfer function with one level of steepness instead testing each transfer function with several steepness levels to address participant gain preference and familiarity. This is further supported by our post experimental structed interview where several participants stated that they could not tell the difference between constant and accelerated curves while others found there to be extreme differences in cursor acceleration. Previous research evaluated several different degrees of steepness with the same function shape to better normalize results over participant preference of gain [8]. For example, gamers might prefer high gain/acceleration while more casual participants might prefer lower gain/acceleration.

In conditions where head tracking was disabled, throughput was not significantly affected by depth. Note how throughput scores are relatively flat in the head-tracking off

conditions in Figure 12. This is consistent with previous studies using non-HMD 3D environments [44]. However, when head-tracking was enabled, depth affected throughput. This suggests that the inclusion of an additional 6DOF input source causes inconsistent performance over target depth.

With constant gain and head tracking off, error rate was consistent with previous mouse-based Fitts' law studies, around 4% [34]. This reinforces the results for other depths. This condition was most comparable to desktop like setups as opposed to VR setups and therefore reinforces our results while also giving us a calibration point to other 2D selection-based research.

From data collected through a post experimental questionnaire, most participants preferred using a combination of head and cursor to aim at targets. This could explain the increased throughput for the steep transfer function due to most participants feeling the steep transfer function was too sensitive and ended up aiming using their head for small and further targets. However, as noted in the previous study, head tracking introduces additional unintentional movement – input noise, or jitter – which explains the higher error rate for all conditions with head tracking on.

Most participants indicated that they felt it was faster and more precise to select targets when head tracking was enabled. Several indicated that they would aim in the general direction of the target using their head, and then fine tune with mouse but other participants found that a secondary method of selection was distracting and made it harder to select targets. This suggests that introducing a secondary input method that moves a cursor via an inconsistent acceleration function while concurrently moving that same cursor via head movement may introduce an extra layer of complexity into the participant's

task of aiming and selection targets. It was observed that participants would often overshoot targets and therefore a cursor that moves with constant acceleration or an option to toggle input selection between the head and mouse may resolve this issue.

Overall, while transfer functions may help the user in selecting a target, introducing a secondary input method that moves the cursor concurrently as well as forcing the user the use a transfer function with a pre-defined acceleration curve steepness may be detrimental when performing a selection task.

5.8 **Summary**

In the second experiment, we added a non-linear gain function to provide cursor acceleration to see if it could improve cursor performance using the mouse. Despite what research indicates, our transfer function did not improve performance of the cursor. We believe that this may have been due to our experimental design where we tested one steepness per transfer function instead of testing for several steepness across a particular transfer function to accommodate participant familiarity and comfortability when using a particular transfer function.

6 Chapter: Conclusion

6.1 Summary

In this thesis, we proposed, implemented and tested EZCursorVR, a 2D head coupled cursor that can be controlled by any 2DOF input device. The idea of EZCursorVR was to leverage benefits of 2DOF input techniques, which had previously been perform better than 6DOF input techniques [30]. The technique was developed for use in VR environments display on head-mounted displays, where 2D input has not previously been studied.

In our first study, we looked at how EZCursorVR's selection performance compared to traditional 3D selection techniques as well as different input techniques for 2D cursor movement. EZCursorVR was tested in a Fitts' law environment built in the Unity game engine and conforming to ISO 9241-9. Our results showed that a 2D cursor controller by the Oculus Touch motion controller performed the best (other than the mouse which we used as a baseline condition). We also discovered that most users preferred to use a combination of the head and 2D input device to move the cursor while some found it to unintentionally move the cursor. We proposed a follow up study that looks at further increasing cursor selection performance by implementing a transfer function which has also shown to increase selection performance [8,9,27].

Our second study explored selection performance of EZCursorVR after the application of several transfer functions to the cursors acceleration. We implemented three transfer functions: Constant, Shallow and Steep. These transfer functions were tested with EZCursorVR with the same Fitts' law selection task framework. Our results indicated that the Constant transfer function performed the best, contrary to previous studies on transfer

functions [8,9]. We believe that this is due to participant unfamiliarity with the cursor acceleration curve and that different curve steepnesses need to also be tested to get an overall idea of how that specific curve performs.

Throughput, movement time and error rate received similar values for the mouse with the head tracking on condition in both our user studies (2.26 bps vs 2.65 bps, 0.92 s vs 1.1 s and 14.9 % vs 16.1 % respectively). This suggests that the data collected was fairly consistent between both studies, which is important because a consistent throughput score provides a baseline of comparison across both studies. Additionally, throughput across both studies with the head on condition decreases as target depth increases, however when head tracking is turned off, throughput remains more or less consistent across target depth which is more in line with what previous studies have noted [44]. This suggests that the addition of head movement consequently introduces diminished performance for further targets. When a participant naturally moves their head towards a target this also moves the cursor. While not an issue for closer and larger targets, for smaller and further targets, even small head movements may cause the cursor to completely pass over and miss the intended target.

It is also worth noting that when comparing throughput results from study 1 for the Ray2D condition and the mouse + head on condition from study 2, the results are almost identical (2.23bps vs 2.24bps) respectively. This demonstrates that Ray2D's Wii like targeting method may be a suitable contender for a controller if used in conjunction with head aiming, since it performs roughly as well as the mouse when head tracking is enabled. This also suggests that having a 2D cursor in VR accelerating at a constant speed may not

necessarily hinder selection performance as the cursor in both methods of selection accelerates by a constant value.

In conclusion, to address our original research question, we showed that 2D selection offers superior performance when compared to 3D selection in HMD VR environments, which validates the basic idea of the cursor. In both studies, participants enjoyed using EZCursorVR and preferred using a combination of head and controller to perform selections. However, results show that for far distance targets, head movement noise introduces unintentional cursor movement, decreasing performance. An option to toggle head on and off could resolve this issue. Applying a head dampening function where head movements under a certain threshold prevent the in game camera from moving, therefore stabilizing small head movements and increasing selection performance can also be explored.

6.2 **Limitations and Future Work**

In both our studies we tested with non-occluded targets which is not realistic scenario but is necessary for Fitts' law evaluations. A follow up study could explore how users can select objects behind other objects. This might be accomplished, for example, by changing the roll of the controller (the unused DOF) for depth selection. The visual design of EZCursorVR itself could be explored, for instance, using different crosshair styles, sizes, and transparency levels. Participants noted that, especially for small targets, the cursor could sometimes occlude targets, making it more difficult to select them. An improved visualization might eliminate such problems

We also note that a number of control variables were chosen based on pilot testing and could be further explored for "fine-tuning" such as alternative transfer function shapes

and constant gain levels. Other factors such as non-gamers/experienced gamers would further add to and reinforce our initial study.

Another limitation we encountered was learnability. A 60-minute user study was not enough time for participants to fully get accustomed a new cursor input device or acceleration that what they are already used to. A longitudinal study would be beneficial for learning the effects of performance of the control schemes over longer periods of time.

A Fitts' law test conforming to ISO 9241-9 does not allow for realistic scenarios such as player movement, target movement and seeking tasks. Performing a gameplay evaluation with EZCursorVR using a commercial game for example, would provide additional insight into player experience and in terms of qualitative data.

Finally, in both studies, participants performed the experiment while in a seated environment while in most VR setups users are moving around and standing up. An input hardware device built specifically for EZCursorVR could be prototyped and built that allows portability as well as control EZCursorVR in VR HMDs, while accommodating first-time users.

6.3 **Design Consideration**

Based on our studies, the following guidelines for designing a 2D cursor for VR are recommended.

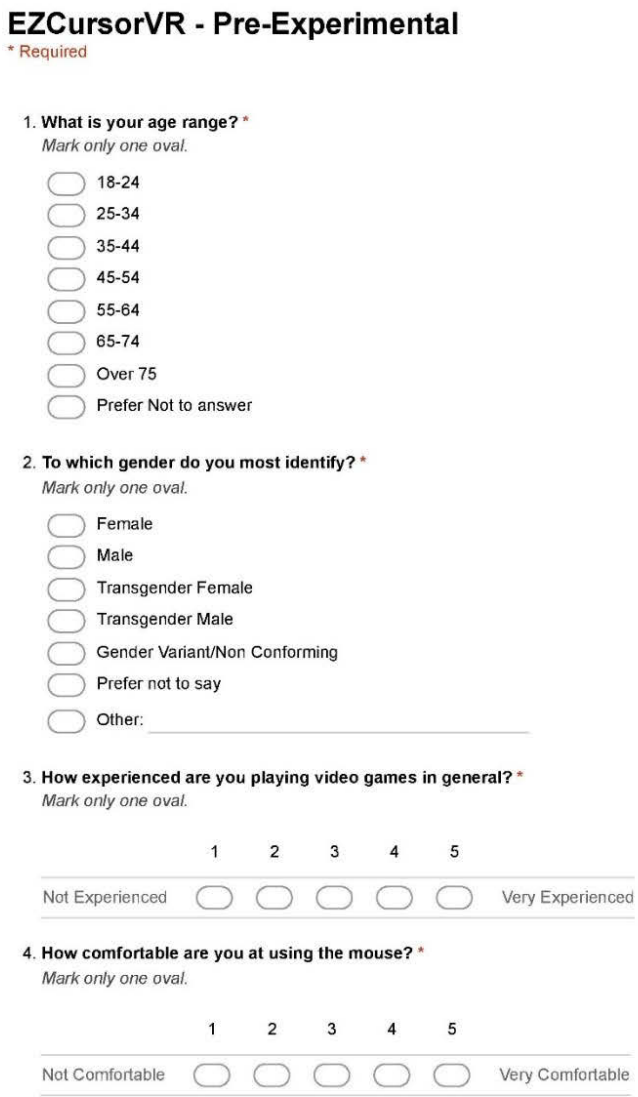
1. Due to the unintentional movement to EZCursorVR when a secondary input method is introduced, we recommend providing a clutch mechanism, to toggle on and off the additional input method.

2. When designing transfer functions for EZCursorVR, giving the user the option to chooses their curve steepness is recommend as a user may be more comfortable with a certain acceleration curve steepness.

Appendices

Appendix A Questionnaires

A.1 User Study 1



EZCursorVR - Pre-Experimental

* Required

1. **What is your age range?** *
   Mark only one oval.
   - ◯ 18-24
   - ◯ 25-34
   - ◯ 35-44
   - ◯ 45-54
   - ◯ 55-64
   - ◯ 65-74
   - ◯ Over 75
   - ◯ Prefer Not to answer

2. **To which gender do you most identify?** *
   Mark only one oval.
   - ◯ Female
   - ◯ Male
   - ◯ Transgender Female
   - ◯ Transgender Male
   - ◯ Gender Variant/Non Conforming
   - ◯ Prefer not to say
   - ◯ Other: _____

3. **How experienced are you playing video games in general?** *
   Mark only one oval.

	1	2	3	4	5	
Not Experienced	◯	◯	◯	◯	◯	Very Experienced

4. **How comfortable are you at using the mouse?** *
   Mark only one oval.

	1	2	3	4	5	
Not Comfortable	◯	◯	◯	◯	◯	Very Comfortable

EZCursorVR: Post-Experimental

* Required

Accuracy

1. **How hard did you find it to aim at targets using the mouse?** *
   Mark only one oval.

	1	2	3	4	5	
Very Easy	◯	◯	◯	◯	◯	Very Hard

2. **How hard did you find it to aim at targets using the rate controlled thumbstick?** *
   Mark only one oval.

	1	2	3	4	5	
Very Easy	◯	◯	◯	◯	◯	Very Hard

3. **How hard did you find it to aim at targets using the position controlled thumbstick?** *
   Mark only one oval.

	1	2	3	4	5	
Very Easy	◯	◯	◯	◯	◯	Very Hard

4. **How hard did you find it to aim at targets using head only?** *
   Mark only one oval.

	1	2	3	4	5	
Very Easy	◯	◯	◯	◯	◯	Very Hard

5. **How hard did you find it to aim at targets using motion cursor?** *
   Mark only one oval.

	1	2	3	4	5	
Very Easy	◯	◯	◯	◯	◯	Very Hard

6. **How hard did you find it to aim at targets using the raycast?** *
   Mark only one oval.

	1	2	3	4	5	
Very Easy	◯	◯	◯	◯	◯	Very Hard

Fatigue

7. **How tiring did you find it to select targets using the mouse?** *
Mark only one oval.

	1	2	3	4	5	
Not Tiring	◯	◯	◯	◯	◯	Very Tiring

8. **How tiring did you find it to select targets using rate controlled thumbstick?** *
Mark only one oval.

	1	2	3	4	5	
Not Tiring	◯	◯	◯	◯	◯	Very Tiring

9. **How tiring did you find it to select targets using position controlled thumbstick?** *
Mark only one oval.

	1	2	3	4	5	
Not Tiring	◯	◯	◯	◯	◯	Very Tiring

10. **How tiring did you find it to select targets using head only?** *
Mark only one oval.

	1	2	3	4	5	
Not Tiring	◯	◯	◯	◯	◯	Very Tiring

11. **How tiring did you find it to select targets using motion cursor?** *
Mark only one oval.

	1	2	3	4	5	
Not Tiring	◯	◯	◯	◯	◯	Very Tiring

12. **How tiring did you find it to select targets using the raycast?** *
Mark only one oval.

	1	2	3	4	5	
Not Tiring	◯	◯	◯	◯	◯	Very Tiring

Speed

13. **How fast do you think the mouse performed when selecting targets?** *
Mark only one oval.

	1	2	3	4	5	
Very Slow	◯	◯	◯	◯	◯	Very Fast

14. **How fast do you think the rate controlled thumbstick performed when selecting targets?** *
Mark only one oval.

	1	2	3	4	5	
Very Slow	◯	◯	◯	◯	◯	Very Fast

15. **How fast do you think the postion controlled thumbstick performed when selecting targets?** *
Mark only one oval.

	1	2	3	4	5	
Very Slow	◯	◯	◯	◯	◯	Very Fast

16. **How fast do you think the head only performed when selecting targets?** *
Mark only one oval.

	1	2	3	4	5	
Very Slow	◯	◯	◯	◯	◯	Very Fast

17. **How fast do you think the motion cursor performed when selecting targets?** *
Mark only one oval.

	1	2	3	4	5	
Very Slow	◯	◯	◯	◯	◯	Very Fast

18. **How fast do you think the raycast performed when selecting targets?** *
Mark only one oval.

	1	2	3	4	5	
Very Slow	◯	◯	◯	◯	◯	Very Fast

Overall

19. **Rate the mouse overall as an input method** *
Mark only one oval.

	1	2	3	4	5	
Poor	◯	◯	◯	◯	◯	Excellent

20. **Rate the rate controlled thumbstick overall as an input method** *
Mark only one oval.

	1	2	3	4	5	
Poor	◯	◯	◯	◯	◯	Excellent

21. **Rate the position controlled thumbstick overall as an input method** *
Mark only one oval.

	1	2	3	4	5	
Poor	◯	◯	◯	◯	◯	Excellent

22. **Rate the head only overall as an input method** *
Mark only one oval.

	1	2	3	4	5	
Poor	◯	◯	◯	◯	◯	Excellent

23. **Rate the motion cursor overall as an input method** *
Mark only one oval.

	1	2	3	4	5	
Poor	◯	◯	◯	◯	◯	Excellent

24. **Rate the raycast overall as an input method** *
Mark only one oval.

	1	2	3	4	5	
Poor	◯	◯	◯	◯	◯	Excellent

25. **Feel free to express any other comments about your experience:**

20. **Rate the rate controlled thumbstick overall as an input method** *
Mark only one oval.

	1	2	3	4	5	
Poor	◯	◯	◯	◯	◯	Excellent

21. **Rate the position controlled thumbstick overall as an input method** *
Mark only one oval.

	1	2	3	4	5	
Poor	◯	◯	◯	◯	◯	Excellent

22. **Rate the head only overall as an input method** *
Mark only one oval.

	1	2	3	4	5	
Poor	◯	◯	◯	◯	◯	Excellent

23. **Rate the motion cursor overall as an input method** *
Mark only one oval.

	1	2	3	4	5	
Poor	◯	◯	◯	◯	◯	Excellent

24. **Rate the raycast overall as an input method** *
Mark only one oval.

	1	2	3	4	5	
Poor	◯	◯	◯	◯	◯	Excellent

25. **Feel free to express any other comments about your experience:**

A.2 User Study 2

VR Mouse Acceleration: Pre-Test Questionaire

* Required

1. **What is your age? (Leave Blank if Prefer not to say)**

2. **To which gender do you most identify?** *
   Mark only one oval.
   - ◯ Male
   - ◯ Female
   - ◯ Gender Variant
   - ◯ Prefer not to say

3. **How experienced are you playing video games in general?** *
   Mark only one oval.

	1	2	3	4	5	
Not Comfortable	◯	◯	◯	◯	◯	Very Confortable

4. **How many times have you tried VR** *
   Mark only one oval.
   - ◯ Never
   - ◯ 1-5
   - ◯ 5-10
   - ◯ 10+

5. **If so, what devices have you used?**
   Check all that apply.
   - ☐ Oculus Rift
   - ☐ HTC Vive
   - ☐ Google Cardboard/Daydream

6. **How experienced are you playing First Person Shooter Games?** *
   Mark only one oval.

	1	2	3	4	5	
Not Experienced	◯	◯	◯	◯	◯	Very Experienced

VR Mouse Acceleration: Post-Test Questionnaire

* Required

1. **Did you find it more precise to select targets with:** *
   Mark only one oval.
   - ◯ Head Tracking On
   - ◯ Head Tracking Off

2. **Did you find it faster to select targets with:** *
   Mark only one oval.
   - ◯ Head Tracking On
   - ◯ Head Tracking Off

3. **Did you find it more tiring to select targets with:** *
   Mark only one oval.
   - ◯ Head Tracking On
   - ◯ Head Tracking Off

4. **Do you prefer using a combination of Head + Mouse to select Targets** *
   Mark only one oval.
   - ◯ Yes
   - ◯ No

5. **Please feel free to express any additional comments:**

Appendix B Consent Forms

B.1 User Study 1

Carleton UNIVERSITY
Canada's Capital University

CUREB clearance #: 106142

Consent Form

Title: EZCursorVR: 2D Selection with Virtual Reality Head-Mounted Displays

Funding Source: NSERC

Date of ethics clearance: To be determined by CUREB (as indicated on the clearance form)

Ethics Clearance for the Collection of Data Expires: To be determined by CUREB (as indicated on the clearance form)

Researcher: Adrian Ramcharitar, Carleton School of Computer Science

---

This study involves one 60 minute experiment. With your consent, the session will be video recorded for analysis by usability software. The experiment involves wearing a head-mounted display (e.g., Oculus Rift) while performing tasks in a virtual environment, such selecting objects in the environment using the tracked 3D controller. Software will automatically and anonymously log your task performance data during participation. We may also (with your consent) take audio/video recordings for further analysis, for example, to study facial expressions to assess your user experience with our software.

Like most virtual reality systems, ours uses a stereo 3D display. Hence you must have normal or corrected-to-normal stereo viewing capabilities to participate – if you can see the 3D effect in 3D movies, you should be fine. There is thus also a minor risk of eye fatigue, headache, or nausea, consistent with the use of stereo displays (e.g., 3D movies) for about 10% of the population. Should you experience such symptoms, you are encouraged to take a break to alleviate the discomfort, or alternatively to withdraw from the study.

You have the right to end your participation in the study at any time, for any reason, up until completion of the experiment. If you withdraw from the study, all information you have provided will be immediately destroyed.

As a token of appreciation, you will receive $10 upon completion of the experiment. This is yours to keep, even if you withdraw from the study.

All research data, including audio/video recordings and any notes will be encrypted and stored on the researcher's password-protected hard drive. Any hard copies of

Page 1 of 2

data (including any handwritten notes or USB keys) will be kept in a locked cabinet at Carleton University. Research data will only be accessible by the researcher.

Once the project is completed, all research data will be kept in anonymous form indefinitely, and potentially used for other research projects on this same topic. If you wish to withdraw your data or any audio/video recordings, please contact us and we will arrange to meet with you to do so. If you would like a copy of the finished research project, you are invited to contact the researcher to request an electronic copy which will be provided to you.

The ethics protocol for this project was reviewed by the Carleton University Research Ethics Board, which provided clearance to carry out the research. Should you have questions or concerns related to your involvement in this research, please contact:

CUREB contact information:
Professor Shelley Brown, Chair (CUREB-B)
Professor Andy Adler, Vice-Chair
Carleton University Research Ethics Board
Carleton University
511 Tory
1125 Colonel By Drive
Ottawa, ON K1S 5B6
Tel: 613-520-2517
ethics@carleton.ca

Researcher contact information:
Adrian Ramcharitar
School of Computer Science
Carleton University
Tel: [redacted]
Email: Adrian.Ramcharitar@carleton.ca

Do you agree to be video-recorded: ___Yes ___No

I _____, choose to participate in a study on 3D user interfaces.

_____ _____
Signature of participant Date

_____ _____
Signature of researcher Date

B.2 User Study 2

Carleton UNIVERSITY
Canada's Capital University

CUREB clearance #: 106142

Consent Form

Title: Making Gains: Leveraging Gain and 2D Input Devices for Selection in Virtual Reality

Funding Source: NSERC

Date of ethics clearance: To be determined by CUREB (as indicated on the clearance form)

Ethics Clearance for the Collection of Data Expires: To be determined by CUREB (as indicated on the clearance form)

Researcher: Adrian Ramcharitar, Carleton School of Computer Science

---

This study involves one 60 minute experiment. With your consent, the session will be video recorded for analysis by usability software. The experiment involves wearing a head-mounted display (e.g., Oculus Rift) while performing tasks in a virtual environment, such selecting objects in the environment using the tracked 3D controller. Software will automatically and anonymously log your task performance data during participation.

Like most virtual reality systems, ours uses a stereo 3D display. Hence you must have normal or corrected-to-normal stereo viewing capabilities to participate – if you can see the 3D effect in 3D movies, you should be fine. There is thus also a minor risk of eye fatigue, headache, or nausea, consistent with the use of stereo displays (e.g., 3D movies) for about 10% of the population. Should you experience such symptoms, you are encouraged to take a break to alleviate the discomfort, or alternatively to withdraw from the study.

You have the right to end your participation in the study at any time, for any reason, up until completion of the experiment. If you withdraw from the study, all information you have provided will be immediately destroyed.

As a token of appreciation, you will receive $10 upon completion of the experiment. This is yours to keep, even if you withdraw from the study.

All research data, including audio/video recordings and any notes will be encrypted and stored on the researcher's password-protected hard drive. Any hard copies of data (including any handwritten notes or USB keys) will be kept in a locked cabinet

at Carleton University. Research data will only be accessible by the researcher.

Once the project is completed, all research data will be kept in anonymous form indefinitely, and potentially used for other research projects on this same topic. If you wish to withdraw your data, please contact us and we will arrange to meet with you to do so. If you would like a copy of the finished research project, you are invited to contact the researcher to request an electronic copy which will be provided to you.

The ethics protocol for this project was reviewed by the Carleton University Research Ethics Board, which provided clearance to carry out the research. Should you have questions or concerns related to your involvement in this research, please contact:

CUREB contact information:
Professor Shelley Brown, Chair (CUREB-B)
Professor Andy Adler, Vice-Chair
Carleton University Research Ethics Board
Carleton University
511 Tory
1125 Colonel By Drive
Ottawa, ON K1S 5B6
Tel: 613-520-2517
ethics@carleton.ca

Researcher contact information:
Adrian Ramcharitar
School of Computer Science
Carleton University
Tel: [redacted]
Email: Adrian.Ramcharitar@carleton.ca

I _____, choose to participate in a study on 3D user interfaces.

_____ _____
Signature of participant Date

_____ _____
Signature of researcher Date

Appendix C Software Implementation

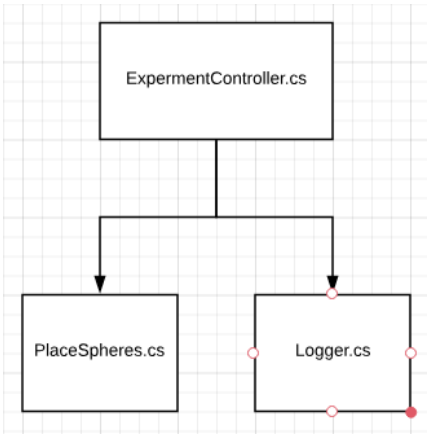
In order to test a device, the user must:

1. Specify input parameters (Width, Amplitude, Number of Spheres, etc...)
2. Specify controller input (Mouse, Thumbstick, Head, etc…)
3. Run testing
4. View logged results

Several key classes are used to control, render and log resulting data and are described below:

Experiment Control

Three classes play a key role in the control of running the experiment and are described in detail below.



UML diagram for experiment control classes

ExperimentController.cs: Allows the user to set parameters for targets, inputs and permutation combinations.

```
public static int numberSpheres = 9;

  //The individual user input array to be put into masArr
 static float[] tAmp = new float[] { 1, 2, 3 };
 static float[] tWid = new float[] {0.75f,0.5f, 0.25f};
 static float[] tDis = new float[] {10,20,30};
```

Code snippet where the user specifies the number of spheres as well as the Amplitude, Width and Depth.

Also generates permutations for the sphere parameters to randomize rounds and stores them into a new array containing the shuffled combination of parameters.

```
public T[][] Permutations<T>(T[][] vals) {
    int numCols = vals.Length;
    int numRows = vals.Aggregate(1, (a, b) => a * b.Length);

    var results = Enumerable.Range(0, numRows)
                            .Select(c => new T[numCols])
                            .ToArray();

    int repeatFactor = 1;
    for (int c = 0; c < numCols; c++) {
        for (int r = 0; r < numRows; r++)
            results[r][c] = vals[c][r / repeatFactor % vals[c].Length];
        repeatFactor *= vals[c].Length;
    }

    return results;
}
```

Code snippet showing the permutation function implementation.

PlaceSpheres.cs: Renders the round of spheres in the scene.

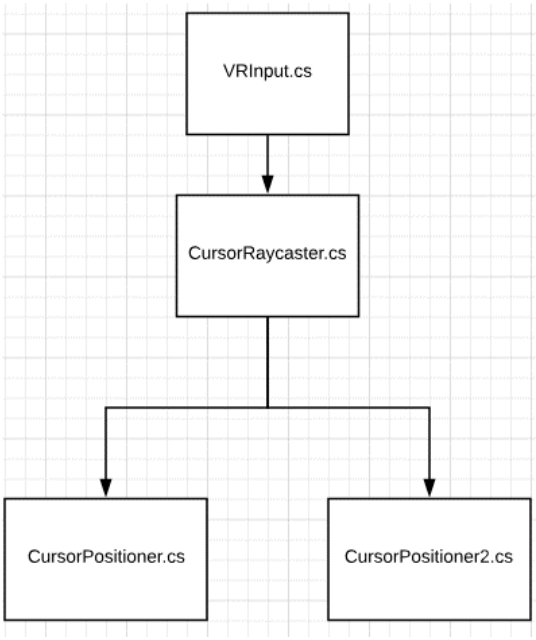
Logger.cs: Logs various experimental data and outputs to .txt file.

```
Participant,Condition,Block,Trial,A,W,Depth,dx,MT(s),Error,Head Tracking,CursorMag,CameraMag
P01,C07,1,1,2,0.5,20,-0.2024747,6.218996,0,1,3.2364,3.233696
P01,C07,1,2,2,0.5,20,-0.1265782,0.9495296,0,1,0.3808129,0.03711144
P01,C07,1,3,2,0.5,20,0.1353782,0.8692122,1,1,0.3956823,0.02131598

Participant,Condition,Block,Trials,A,W,Depth,id,We,IDe,MT(s),Error Rate,TP,Head Tracking,Intersection,Linearity
P01,C07,1,27,2,0.5,20,2.321928,0.539681,2.234468,0.9623235,11.11111,2.321951
P01,C07,1,27,3,0.25,30,3.70044,0.3518683,3.251858,1.243145,33.33333,2.615831
P01,C07,1,27,3,0.5,10,2.807355,0.7216635,2.366549,1.117681,22.22222,2.117374
```

Text file that Loggers.cs generates. (Top: Log for per sphere selection. Bottom: Log for per round selections.)

Cursor positioning and rendering

There are four main classes that are used in the process of rendering both cursors as well as positioning them in the scene. They are described in detail below.



UML diagram for cursor positioning and rendering classes

VRInput.cs: Handles all input parameters such as click, double click, on mouse up, on mouse down.

CursorRaycaster.cs: Casts a Raycast from the reticle onto the canvas. Firstly, a ray is created that points forward from the camera (or center of the HMD viewport). Then the ray is casted into the scene and performs an intersection test with objects in the scene.

If the object happens to be a sphere, call the RaycastIntersect() function to get the intersection coordinates.

```
if (trackedHand.GetComponent<MotionController2>().enabled == true) {


    Vector3 fromPosition = Camera.main.transform.position;
    Vector3 toPosition = pointer.transform.position;
    // Vector3 direction = toPosition - fromPosition;


    rayStart2 = trackedHand.transform.position;
    rayDir2 = trackedHand.transform.TransformDirection(Vector3.forward);

    ray2 = new Ray(rayStart2, rayDir2);

    //Main ray
    rayStart = Camera.main.gameObject.transform.position;
    rayDir = toPosition - fromPosition;

    ray = new Ray(rayStart, rayDir);
```

Code snippet showing raycast generation from the tracked controller

```
// Do the raycast forweards to see if we hit an interactive item
if (Physics.Raycast(ray, out hit,Mathf.Infinity, lMask)) { //Test t values here


    //Returns Vector3 positon of the raycast
    rayHitPos = hit.point;

    VRInteractiveItem interactible = hit.collider.GetComponent<VRInteractiveItem>(); //attempt to get the VRInteractiveItem
    m_CurrentInteractible = interactible;

    // If we hit an interactive item and it's not the same as the last interactive item, then call Over
    if (interactible && interactible != m_LastInteractible)
        interactible.Over();

    // Deactive the last interactive item
    if (interactible != m_LastInteractible)
        DeactiveLastInteractible();

       m_LastInteractible = interactible;

    // Something was hit, set at the hit position

    if (m_Reticle2) {
        // m_Reticle.SetPosition(hit);

        m_Reticle2.SetPosition(hit);
    }


    if (OnRaycasthit != null) {
        OnRaycasthit(hit);
    }
```

Code snippet showing the raycast hit detection on targets

CursorPositioner.cs: Positions and moves first invisible cursor against the plane. The SetPostion() method positions the first invisible cursor onto the invisible plane. This cursor's position is constrained to the area of the invisible plane (or the HMD's FOV).

```
public void SetPosition() {
    // Set the position of the reticle to the default distance in front of the camera.
    m_ReticleTransform.position = m_Camera.position + m_Camera.forward * m_DefaultDistance;

    // Set the scale based on the original and the distance from the camera.
    m_ReticleTransform.localScale = m_OriginalScale * m_DefaultDistance;

    // The rotation should just be the default.
    m_ReticleTransform.localRotation = m_OriginalRotation;
}


// This overload of SetPosition is used when the VREyeRaycaster has hit something.
public void SetPosition(RaycastHit hit) {

   // Debug.Log("HIT" + hit.point);
   // Debug.Log("POS" + m_ReticleTransform.position);

    //Set z to raycast postion
    Vector3 temp = m_ReticleTransform.position;
    temp.z = hit.point.z;
    m_ReticleTransform.position = temp;


    m_ReticleTransform.localScale = m_OriginalScale * hit.distance;

    // If the reticle should use the normal of what has been hit...
    if (m_UseNormal)
        // ... set it's rotation based on it's forward vector facing along the normal.
        m_ReticleTransform.rotation = Quaternion.FromToRotation(Vector3.forward, hit.normal);
    else
        // However if it isn't using the normal then it's local rotation should be as it was originally.
        m_ReticleTransform.localRotation = m_OriginalRotation;
}
```

Code snippet showing the first cursor position

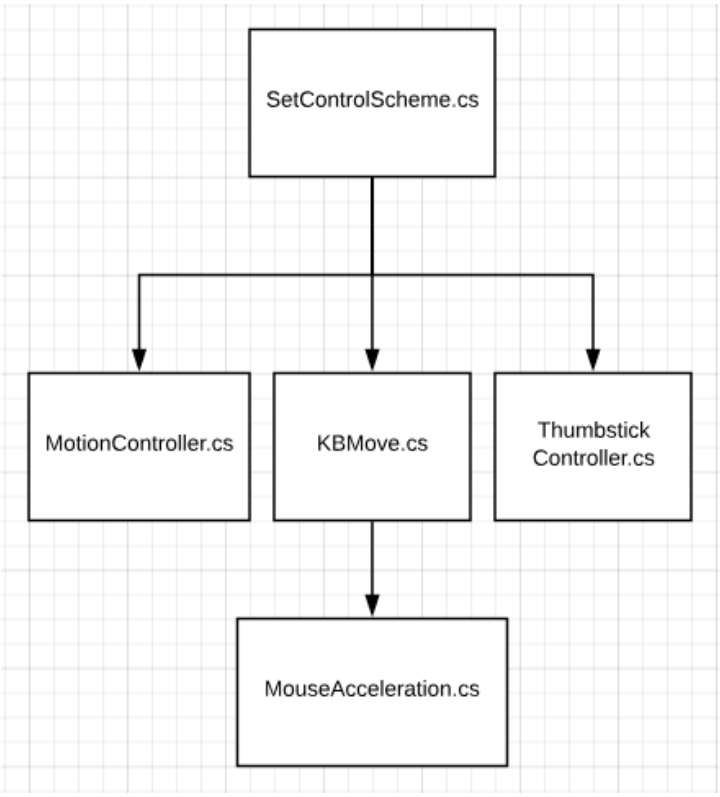
CursorPositioner2.cs: Positions, moves and resizes the 2nd visible cursor against objects in the scene. Also ensures that the cursor is always rotated to face the user.

```
// This overload of SetPosition is used when the the VREyeRaycaster hasn't hit anything.
public void SetPosition() {
    // Set the position of the reticle to the default distance in front of the camera.
    m_ReticleTransform.position = m_Camera.position + m_Camera.forward * m_DefaultDistance;

    // Set the scale based on the original and the distance from the camera.
    m_ReticleTransform.localScale = m_OriginalScale * m_DefaultDistance;

    // The rotation should just be the default.
    m_ReticleTransform.localRotation = m_OriginalRotation;
}


// This overload of SetPosition is used when the VREyeRaycaster has hit something.
public void SetPosition(RaycastHit hit) {

        m_ReticleTransform.position = hit.point;


        m_ReticleTransform.localScale = m_OriginalScale * hit.distance;


    // If the reticle should use the normal of what has been hit...
    if (m_UseNormal)
        // ... set it's rotation based on it's forward vector facing along the normal.
        m_ReticleTransform.rotation = Quaternion.FromToRotation(Vector3.forward, hit.normal);
    else
        // However if it isn't using the normal then it's local rotation should be as it was originally.
        m_ReticleTransform.localRotation = m_OriginalRotation;
}
```

Code snippet showing the second rendered cursor position

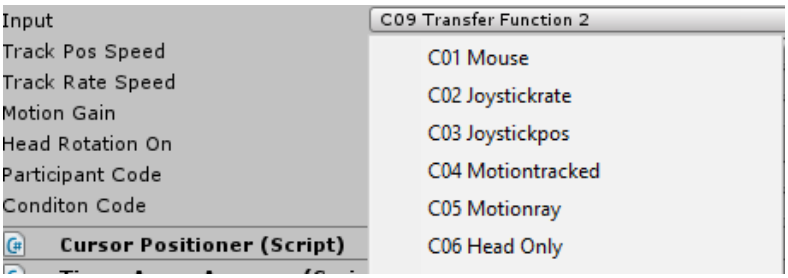
Cursor movement via Input devices

There are five main classes that are used receive movement from several different types on input devices and translate their movement to cursor movement. They are described below:



UML diagram for cursor movement via input devices

SetControlScheme.cs: Allows user the set their preferred control scheme (Mouse, Joystick, Motion…). This class also allows the user to easily select their input device to test with using a drop down list in the main Unity editor.



SetControlScheme.cs resulting list in the Unity editor

MotionController.cs: Controls the cursor via the Oculus Rift's GetLocalControllerPosition() method to return the touch controller's rotational position in Vector3 coordinates.

```
//Tracked postion of Cube hand object
transform.localPosition = OVRInput.GetLocalControllerPosition(Raycontroller);
transform.localRotation = OVRInput.GetLocalControllerRotation(Raycontroller);
```

Code snippet showing how the framework receives the position from touch controller's rotation.

KBMove.cs: Controls the cursor via the mouse using GetAxisRaw() function and return the position as a Vector2 coordinate.

```
Vector2 mousePos = pointer.GetComponent<RectTransform>().localPosition;

//Debug.Log( "Axis" + Input.GetAxis("Mouse X") + " " + Input.GetAxis("Mouse Y"));
if (gain == true) {
    mousePos += new Vector2(Input.GetAxis("Mouse X"), Input.GetAxis("Mouse Y")) * mouseMoveSpeed;
    Debug.Log("CONSTANT");
```

Code snippet showing how the framework receives the position from the mouse.

ThumbstickController.cs: Controls the cursor via the Touch controller's Axis2D.SecondaryThumbstick and returns the position as a Vector2 coordinates.

```
Vector2 ContPos = ThumbStickPointer.GetComponent<RectTransform>().localPosition;

if (PosControl) {
    ContPos = new Vector2(OVRInput.Get(OVRInput.Axis2D.SecondaryThumbstick).x, OVRInput.Get(OVRInput.Axis2D.SecondaryThumbstick).y)
```

Code snippet showing how the framework receives the position from touch controller's thumbstick.

MouseAcceleration.cs: Controls cursor acceleration using a user defined transfer function.

```
float curMouseX = Input.GetAxisRaw("Mouse X");
float curMouseY = Input.GetAxisRaw("Mouse Y");

//Debug.Log("Normal: " + Mathf.Sqrt(Mathf.Pow(curMouseX,2) + Mathf.Pow(curMouseY,2)));

curMouseX = (k / (1 + Mathf.Exp(a + b * Mathf.Abs(curMouseX))));
curMouseY = (k / (1 + Mathf.Exp(a + b * Mathf.Abs(curMouseY))));
```

Code snippet showing how the framework uses a transfer function to accelerate cursor movement.

References

1. Ferran Argelaguet and Carlos Andujar. 2008. Improving 3D selection in VEs through expanding targets and forced disocclusion. In *Proceedings of the 9th international symposium on Smart Graphics (SG '08)*, 45–57. https://doi.org/10.1007/978-3-540-85412-8_5
2. Ferran Argelaguet and Carlos Andujar. 2009. Efficient 3D pointing selection in cluttered virtual environments. *IEEE Computer Graphics and Applications* 29, 6: 34–43. https://doi.org/10.1109/MCG.2009.117
3. Ferran Argelaguet and Carlos Andujar. 2013. A survey of 3D object selection techniques for virtual environments. *Computers and Graphics (Pergamon)* 37, 3: 121–136. https://doi.org/10.1016/j.cag.2012.12.003
4. François Bérard, Jessica Ip, Mitchel Benovoy, Dalia El-Shimy, Jeffrey R. Blum, and Jeremy R. Cooperstock. 2009. *Did "minority report" get it wrong? Superiority of the mouse over 3D input devices in a 3D placement task*. https://doi.org/10.1007/978-3-642-03658-3_45
5. Doug A. Bowman, C Wingrave, J Campbell, and V Ly. 2001. Using pinch gloves (TM) for both natural and abstract interaction techniques in virtual environments. *HCI International*, 0106: 629–633. https://doi.org/citeulike-article-id:2864674
6. Frederick P. Brooks. 1999. What's real about virtual reality? *IEEE Computer Graphics and Applications* 19, 6: 16–27. https://doi.org/10.1109/38.799723
7. Gerd Bruder, Frank Steinicke, and Wolfgang Stuerzlinger. 2013. Touching the void revisited: Analyses of touch behavior on and above tabletop surfaces. In *Lecture Notes in Computer Science*, 278–296. https://doi.org/10.1007/978-3-642-

40483-2_19

8. Géry Casiez and Nicolas Roussel. 2011. No more Bricolage! Methods and Tools to Characterize, Replicate and Compare Pointing Transfer Functions. In *Proceedings of the ACM symposium on User interface software and technology*, 603–614. https://doi.org/10.1145/2047196.2047276

9. Géry Casiez, Daniel Vogel, Ravin Balakrishnan, and Andy Cockburn. 2008. The impact of control-display gain on user performance in pointing tasks. *Human-Computer Interaction* 23, 3: 215–250. https://doi.org/10.1080/07370020802278163

10. Inrak Choi, Elliot W. Hawkes, David L. Christensen, Christopher J. Ploch, and Sean Follmer. 2016. Wolverine: A wearable haptic interface for grasping in virtual reality. In *IEEE International Conference on Intelligent Robots and Systems*, 986–993. https://doi.org/10.1109/IROS.2016.7759169

11. Andéol Évain, Ferran Argelaguet, Géry Casiez, Nicolas Roussel, and Anatole Lécuyer. 2016. Design and evaluation of fusion approach for combining brain and gaze inputs for target selection. *Frontiers in Neuroscience* 10. https://doi.org/10.3389/fnins.2016.00454

12. Robert Farmani, Yasin and Teather. 2018. Viewpoint Snapping to Reduce Cybersickness in Virtual Reality. In *Proceedings of Graphics Interface Conference*, 159–166. https://doi.org/10.20380/GI2018.21

13. Paul M. Fitts. 1954. The information capacity of the human motor system in controlling the amplitude of movement. *Journal of Experimental Psychology* 47, 6:

381–391. https://doi.org/10.1037/h0055392

14. Tovi Grossman and Ravin Balakrishnan. 2005. The bubble cursor: enhancing target acquisition by dynamic resizing of the cursor's activation area. *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*: 281–290. https://doi.org/10.1145/1054972.1055012

15. Sujin Jang, Wolfgang Stuerzlinger, Satyajit Ambike, and Karthik Ramani. 2017. Modeling Cumulative Arm Fatigue in Mid-Air Interaction based on Perceived Exertion and Kinetics of Arm Motion. In *Proceedings of the CHI Conference on Human Factors in Computing Systems*, 3328–3339. https://doi.org/10.1145/3025453.3025523

16. David Antonio Gómez Jáuregui, Ferran Argelaguet, and Anatole Lecuyer. 2012. Design and evaluation of 3D cursors and motion parallax for the exploration of desktop virtual environments. In *Proceedings of the IEEE Symposium on 3D User Interfaces*, 69–76. https://doi.org/10.1109/3DUI.2012.6184186

17. Leroy Jenkins, Willian & B Connor, and Minna. 1949. Some design factors in making settings on a linear scale. *Journal of Applied Psychology* 33: 395–409.

18. Daniel Kharlamov, Brandon Woodard, Liudmila Tahai, and Krzysztof Pietroszek. 2016. TickTockRay: smartwatch-based 3D pointing for smartphone-based virtual reality. In *Proceedings of the ACM Conference on Virtual Reality Software and Technology*, 363–364. https://doi.org/10.1145/2993369.2996311

19. Regis Kopper, Felipe Bacim, and Doug A. Bowman. 2011. Rapid and accurate 3D selection by progressive refinement. In *Proceedings of the IEEE Symposium on 3D*

User Interfaces, 67–74. https://doi.org/10.1109/3DUI.2011.5759219

20. Joseph J. LaViola, Ernst Kruijff, Ryan P. McMahan, Doug Bowman, and Ivan P. Poupyrev. 2017. *3D user interfaces : Theory and Practice*. Addison-Wesley Professional, Redwood City, CA, USA.
21. Sangyoon Lee, Jinseok Seo, Gerard Jounghyun Kim, and Chan-mo Park. 2003. Evaluation of pointing techniques for ray casting selection in virtual environments. In *Proceedings of the International Conference on Virtual Reality and Its Application in Industry*, 38–44. https://doi.org/10.1117/12.497665
22. Paul Lubos, Gerd Bruder, and Frank Steinicke. 2014. Analysis of direct selection in head-mounted display environments. In *Proceedings of the IEEE Symposium on 3D User Interfaces*, 11–18. https://doi.org/10.1109/3DUI.2014.6798834
23. I. Scott MacKenzie. 1992. Fitts' Law as a Research and Design Tool in Human-Computer Interaction. *Human–Computer Interaction* 7, 1: 91–139. https://doi.org/10.1207/s15327051hci0701_3
24. I. Scott MacKenzie. 2013. *Human-Computer Interaction: An Empirical Research Perspective*. https://doi.org/10.1007/s13398-014-0173-7.2
25. Victoria McArthur, Steven J. Castellucci, and I. Scott MacKenzie. 2009. An empirical comparison of "wiimote" gun attachments for pointing tasks. In *Proceedings of the ACM SIGCHI Symposium on Engineering Interactive Computing Systems*, 203. https://doi.org/10.1145/1570433.1570471
26. Mark R Mine, Frederick P Brooks, Jr., and Carlo H Sequin. 1997. Moving Objects in Space: Exploiting Proprioception In Virtual-Environment Interaction. In

Proceedings of ACM SIGGRAPH, 19–26. https://doi.org/10.1145/258734.258747

27. Mathieu Nancel, Emmanuel Pietriga, Olivier Chapuis, and Michel Beaudouin-Lafon. 2015. Mid-Air Pointing on Ultra-Walls. In *ACM Transactions on Computer-Human Interaction*, 1–62. https://doi.org/10.1145/2766448
28. Daniel Natapov and I Scott MacKenzie. 2010. The trackball controller: improving the analog stick. In *Proceedings of the International Academic Conference on the Future of Game Design and Technology*, 175–182. https://doi.org/10.1145/1920778.1920803
29. JY Oh and Wolfgang Stuerzlinger. 2005. Moving objects with 2D input devices in CAD systems and Desktop Virtual Environments. *Proceedings of Graphics Interface 2005*. https://doi.org/10.1145/1089508.1089541
30. Jeffrey S Pierce, Andrew Forsberg, Matthew J Conway, Seung Hong, Robert Zeleznik, and Mark R Mine. 1997. Image Plane Interaction Techniques in 3D Immersive Environments. In *Proceedings of ACM SIGGRAPH Symposium on Interactive 3D Graphics*, 39–44.
31. I Poupyrev, T Ichikawa, S Weghorst, and M Billinghurst. 1998. Egocentric Object Manipulation in Virtual Environments: Empirical Evaluation of Interaction Techniques. *Computer Graphics Forum* 17, 3: 41–52. https://doi.org/10.1111/1467-8659.00252
32. Ivan Poupyrev and Mark Billinghurst. 1996. The go-go interaction technique: non-linear mapping for direct manipulation in VR. *Proceedings of the ACM Symposium on User Interface Software and Technology*: 79–80.

https://doi.org/10.1145/237091.237102

33. Yuanyuan Qian and Robert J. Teather. 2017. The eyes don't have it : An empirical comparison of head based and eye-based selection in virtual reality. In *Proceedings of the ACM Symposium on Spatial User Interaction*, 91–98. https://doi.org/10.1145/3131277.3132182

34. A. Ramcharitar and R.J. Teather. 2017. A Fitts' law evaluation of video game controllers: Thumbstick, touchpad and gyrosensor. In *Proceedings of the ACM Conference on Human Factors in Computing Systems*. https://doi.org/10.1145/3027063.3053213

35. Adrian Ramcharitar and Robert.J. Teather. 2018. EZCursorVR: 2D Selection with Virtual Reality Head-Mounted Display. In *Proceedings of Graphics Interface 2018*, 114-- 121. https://doi.org/10.20380/GI2018.17

36. Leila Schemali and Elmar Eisemann. 2014. Design and evaluation of mouse cursors in a stereoscopic desktop environment. In *In Proceedings of the IEEE Symposium on 3D User Interfaces*, 67–70. https://doi.org/10.1109/3DUI.2014.6798844

37. I. Scott MacKenzie and Stan Riddersma. 1994. Effects of output display and control—display gain on human performance in interactive systems. *Behaviour and Information Technology* 13, 5: 328–337. https://doi.org/10.1080/01449299408914613

38. International Standard. 2000. Ergonomic requirements for office work with visual display terminals (VDTs) - Part 9: Requirements for non -keyboard input devices.

Iso 2000 2000: 54.

39. Anthony Steed. 2006. Towards a general model for selection in virtual environments. In *Proceedings of the IEEE Symposium on 3D User Interfaces*, 103–110. https://doi.org/10.1109/VR.2006.134

40. Anthony Steed and Chris Parker. 2004. 3D Selection Strategies for Head Tracked and Non-Head Tracked Operation of Spatially Immersive Displays. *International Immersive Projection Technology Workshop.*: 1–8.

41. Robert J. Teather, Andriy Pavlovych, and Wolfgang Stuerzlinger. 2009. Effects of latency and spatial jitter on 2D and 3D pointing. In *Proceedings IEEE Virtual Reality*, 229–230. https://doi.org/10.1109/VR.2009.4811029

42. Robert J. Teather and Wolfgang Stuerzlinger. 2007. Guidelines for 3D positioning techniques. In *Proceedings of the Conference on Future Play*, 61. https://doi.org/10.1145/1328202.1328214

43. Robert J. Teather and Wolfgang Stuerzlinger. 2011. Pointing at 3D targets in a stereo head-tracked virtual environment. In *Proceedings of the IEEE Symposium on 3D User Interfaces*, 87–94. https://doi.org/10.1109/3DUI.2011.5759222

44. Robert J. Teather and Wolfgang Stuerzlinger. 2013. Pointing at 3D target projections with one-eyed and stereo cursors. *Proceedings of the ACM Conference on Human Factors in Computing Systems*: 159–168. https://doi.org/10.1145/2470654.2470677

45. Robert J Teather and Wolfgang Stuerzlinger. 2014. Visual aids in 3D point selection experiments. In *Proceedings of the ACM Symposium on Spatial User*

Interaction, 127–136. https://doi.org/10.1145/2659766.2659770

46. Lode Vanacken, Tovi Grossman, and Karin Coninx. 2007. Exploring the effects of environment density and target visibility on object selection in 3D virtual environments. In *Proceedings of the IEEE Symposium on 3D User Interfaces*, 115–122. https://doi.org/10.1109/3DUI.2007.340783

47. Colin Ware and Kathy Lowther. 1997. Selection using a one-eyed cursor in a fish tank VR environment. *ACM Transactions on Computer-Human Interaction* 4, 4: 309–322. https://doi.org/10.1145/267135.267136

48. Jonathan Wonner, Antonio Capobianco, and Dominique Bechmann. 2012. Starfish : a Selection Technique for Dense Virtual Environments. *Proceedings of the ACM Symposium on Virtual Reality Software and Technology*: 101–104. https://doi.org/10.1145/2407336.2407356

49. Thomas S. Young, Robert J. Teather, and I. Scott Mackenzie. 2017. An arm-mounted inertial controller for 6DOF input: Design and evaluation. In *Proceedings of the IEEE Symposium on 3D User Interfaces*, 26–35. https://doi.org/10.1109/3DUI.2017.7893314

50. Majed Al Zayer, Sam Tregillus, and Eelke Folmer. 2016. PAWdio: Hand Input for Mobile VR using Acoustic Sensing. In *Proceedings of the Symposium on Computer-Human Interaction in Play*, 154–158. https://doi.org/10.1145/2967934.2968079