

FACULDADE DE ENGENHARIA DA UNIVERSIDADE DO PORTO

Object Selection in DeskVR

João Miguel Monteiro



Mestrado em Engenharia Informática e Computação

Supervisor: Daniel Mendes

Second Supervisor: Rui Rodrigues

July 29, 2022

Object Selection in DeskVR

João Miguel Monteiro

Mestrado em Engenharia Informática e Computação

July 29, 2022

Abstract

DeskVR is a form of virtual reality in which the user can experience Virtual Reality while sitting at a desk without the need for extensive movement. This makes it better suited for professional work environments, such as architecture or data analysis, where productivity over extended periods of time is important. If a standing VR scenario were implemented in these cases, it would cause, after extended use, exhaustion which would in turn reduce productivity.

One of the of the most important tasks that can cause such exhaustion is object selection. In the context of DeskVR, the exploration of such a task has not been deeply researched. While in a standing VR environment the user could use its hand to either grab a nearby object or aim at a distant one, these kinds of movements may not be the most adequate for a seated down experience.

In this thesis we focused on ways of selecting out of reach objects in a seated desk VR environment without the need of exhausting movements. We explored existing object selection techniques and assessed their suitability and adaptability for DeskVR, and proposed our own approach to the problem, the TouchRay, an out of reach object selection technique, that also allows for hierarchy selection: selection of objects' sub-components by traversing their hierarchy trees.

In the end, we conducted experiments that revealed that, while our method was not an overall improvement on existing ones, there are situations in which the TouchRay is favourable for a DeskVR scenario.

With this study, we aimed to contribute to the eventual wider adoption of VR technologies in more professional settings.

Keywords: virtual reality, selection, DeskVR

Resumo

DeskVR é uma forma de realidade virtual na qual o utilizador pode experimentar realidade virtual enquanto sentado numa mesa sem a necessidade de muito movimento. Este facto torna-o mais adequado para ambientes de trabalho profissionais, como arquitetura ou análise de dados, onde a produtividade por longos períodos de tempo é importante. Se um cenário de VR em pé fosse implementado nesses casos, isso causaria, após uso prolongado, exaustão que, por sua vez, reduziria a produtividade.

Uma das tarefas mais importantes que pode causar tal exaustão é a seleção de objetos, uma vez que é necessária de modo a poder interagir com qualquer objeto. No contexto de DeskVR, a exploração de tal tarefa não foi pesquisada profundamente. Enquanto num ambiente de VR em pé, o utilizador pode usar a mão para agarrar um objeto próximo ou apontar para um distante, esses tipos de movimentos podem não ser os mais adequados para uma experiência sentada.

Nesta tese vamos-nos focar em formas de selecionar objetos fora de alcance num ambiente de VR numa secretária, sem a necessidade de movimentos exaustivos. Vamos explorar técnicas de seleção de objetos existentes e avaliar a sua adequação e adaptabilidade para DeskVR, e propor a nossa própria abordagem ao problema, o TouchRay, uma técnica de seleção de objetos fora de alcance, que também permite a seleção de sub-componentes dos objetos ao percorrer as suas árvores hierárquicas.

No final, realizamos experiências que revelaram que, embora o nosso método não tenha sido uma melhoria geral em relação aos existentes, há situações em que o TouchRay é favorável para um cenário DeskVR.

Com este estudo, temos como objetivo contribuir para a eventual adoção mais ampla de tecnologias de VR em ambientes mais profissionais.

Keywords: realidade virtual, seleção, DeskVR

*“Mistakes are not shackles that halt one from stepping forward.
Rather, they are that which sustain and grow one’s heart.”*

Mavis Vermillion

Contents

1	Introduction	1
1.1	Context and Motivation	1
1.2	Problem	2
1.3	Objectives	2
1.4	Document Structure	2
2	Related Work	3
2.1	DeskVR	3
2.1.1	Cybersickness	3
2.1.2	Presence	4
2.1.3	Interaction with the environment	4
2.2	Object Selection in Immersive Virtual Environments	5
2.2.1	Object selection in 3D representations of the IVE	5
2.2.2	Object selection in 2D representations of the IVE	8
2.3	Discussion	10
2.4	Summary	13
3	The TouchRay	14
3.1	Design	14
3.1.1	Candidate Step	14
3.1.2	Refinement Step	16
3.1.3	Hierarchy	16
3.2	Implementation	18
3.2.1	Visual feedback	18
3.2.2	Hierarchy	19
3.3	Summary	20
4	Method Evaluation	21
4.1	Setup	21
4.2	Additional Methods	22
4.2.1	Depth Ray	23
4.2.2	RayCursor	24
4.3	Methodology	24
4.4	Tasks	25
4.5	Metrics	26
4.6	Participants	27
4.7	Results	28
4.7.1	Objective Results	28

4.7.2	Subjective Results	32
4.7.3	Hierarchy	33
4.7.4	Discussion	33
4.8	Summary	34
5	Conclusions and future work	35
5.1	Future work	35
	References	37

List of Figures

2.1	Virtual representation of the desk in the virtual world [18]	4
2.2	A) A typical radiology reading room; B) Vrrrroom’s approach combining virtual reality and desktop touch interactions. [15]	5
2.3	RayCursor with different visual feedbacks [4]	6
2.4	PRECIOUS technique [12]: (a) selection cone intersecting various objects, (b) refinement phase, moving the user closer to the objects	7
2.5	Example of a volumetric display	8
2.6	Selection techniques proposed by Grossman and Balakrishnan [8]	9
2.7	The Vacuum [5]	9
2.8	Left: Experimental setup with sphere-casting. Right: quad- menu stage of SQUAD in the experiment. [9]	10
3.1	Visual representation of the ray	15
3.2	Example of the end of the candidate step. Because the ray was intersecting the objects <i>A</i> and <i>B</i> , those objects are added to the candidate list.	15
3.3	Example of an object hierarchy tree	16
3.4	Candidate list of the chair	17
3.5	Final candidate list of the chair	17
3.6	Architecture of the TouchRay	18
3.7	Candidate Step: Intersected objects change color for visual feedback	19
3.8	The closest object is green while the others are translucent	19
3.9	After moving the finger the top candidate changed	19
4.1	A participant of the experiment	21
4.2	Oculus Rift S controller	22
4.3	When the ray first intersects with multiple objects, the closest one becomes the candidate object.	23
4.4	After using the joystick to move the depth marker the candidate object is updated.	23
4.5	When the ray intersects an object for the first time, the marker is moved to the point of intersection.	24
4.6	After the first intersection, a white line connects the marker with its closest object.	24
4.7	Scenario used for adaptation	25
4.8	Scenarios used in trial. A: DisperseClose; B: DisperseFar; C: AggregateClose; D: AggregateFar	26
4.9	Rocket engine used for hierarchy task	27
4.10	Example of Hierarchy selection, GroupLast	28
4.11	Total time to complete each task using the three methods, in milliseconds.	30
4.12	Total number of selections needed to complete each task using the three methods	31

List of Tables

2.1	Selection Methods	12
4.1	Questionnaire answers (media, interquartile range). * and † represent statistically significant differences between the methods.	32

Abreviaturas e Símbolos

CS	Cybersickness
HMD	Head Mounted Display
IVE	Immersive Virtual Environment
VR	Virtual Reality

Chapter 1

Introduction

Virtual Reality (VR) has been the subject of study and discussion for decades, but, due to its price and complexity, it has not been able to gain much traction as a relevant technology. More recently, however, with major companies, like Oculus and Steam, heavily investing in this type of technology, VR has become more affordable and easier to use, making it more appealing for mass consumption and increasing its popularity.

1.1 Context and Motivation

Virtual Reality (VR) is a technology that allows its users to experience a 3D virtual world by placing them inside the environment, allowing them to perceive the world as if they were a part of it. For this reason, many professional workers, like doctors, architects, and data scientists, can benefit from the use of this technology, as it allows them to have a better 3D representation of their 3D data.

VR encourages its users to interact with the environment in a natural way, such as moving around, and picking objects like they would in the real world. Because of this, VR often requires a lot of physical movement that, after long periods of use, can cause fatigue in its users. Due to this problem, VR is not optimal for a work environment, as tiredness can lead to a reduction in productivity and work performance.

DeskVR appeared to combat the need for tiring movements in VR, having as its objective the creation of a VR experience in which the user can be seated at a desk without having the need to perform movements that might be considered draining. With this DeskVR hopes to contribute towards the seamless integration of VR technologies into more professional work environments.

1.2 Problem

In order to make DeskVR a reality, certain interactions, that are essential to VR, need to be adapted in such a way that they do not depend on large amounts of muscular effort. Object selection is one of these interactions.

To be able to interact with a virtual environment, a person needs to select the object they want to interact with. For that reason, object selection is a crucial task in any interactive environment, VR being no different, and requires special attention and techniques to be capable of selecting any selectable object present in the application.

While in a standing VR experience the user could simply move towards the desired object and grab it with their virtual hand, this is not practical in DeskVR as the user's movement is restricted by the office desk. Furthermore, even though out of reach object selection techniques exist, they were not designed with DeskVR in mind, and, therefore, most of them rely on the user pointing at the desired object, an action that, after extended periods of time, can become tiring.

This is a problem, since without proper out of reach object selection techniques, designed specifically for the defined environment, the use of DeskVR is hindered and not suitable for professional work environments.

1.3 Objectives

With this study, we aim to identify an object selection technique that is suitable for a DeskVR scenario. We explore existing techniques that, despite not having been developed for this purpose, can be used while seated, and a new method that we propose, the TouchRay, a method that allows for out of reach object selection with an additional feature: the ability to select intersected sub-components of the intersected object.

Because a person's reach, while seated, is small, we treat all selections as out of reach selections.

1.4 Document Structure

In the *Related Work*, chapter 2, we begin by presenting articles from different researchers that focus on the themes explored by this thesis, the environment of DeskVR, and out of reach object selection in virtual environments. Furthermore, in chapter 3, *Proposed Solution*, we present our approach and solution to the given problem.

In chapter 4, we explain the experiments that were conducted in order to be able to compare our solution to previously existing ones, as well as present and analyse the results of the experiments. Finally, in the *Conclusion*, chapter 5, we give our final thoughts about the conducted study, as well as point out possible directions towards future research.

Chapter 2

Related Work

To better grasp the concept of our work and the problems and difficulties surrounding it, we start by presenting the literature encompassing the areas of DeskVR. Subsequently, we cover previous works mentioning different ways of selecting distant objects inside an immersive virtual environment.

2.1 DeskVR

Over the years, extensive studies have been made about the possible application of Virtual Reality to a desk environment, one in which the user is seated at an office desk while interacting with a VR environment. It has been shown that, by seamlessly integrating an office desk with the hardware usually required by VR, for example, the Head Mounted Display (HMD), this technology can aid the performance of certain professions, such as data analysts and researchers [15, 1]. Zielasko *et al.* [1] describe the scenario of fully immersive DeskVR in an attempt to smoothly integrate virtual reality technology with existing workflows, as well as some of the challenges of this kind of setup, namely the prevention of cybersickness, how to maximize the user's presence, and how to interact with the presented virtual environment.

2.1.1 Cybersickness

One major disadvantage of VR environments is the phenomenon of cybersickness (CS), commonly present in VR applications [7]. CS, often confused with motion sickness, occurs when there is a conflict between the information the user receives through their senses and the ones that would be expected in a certain situation. Like motion sickness, CS can cause many negative effects such as nausea, headaches, and eye strain, to name a few. Because of its negative effects, the prevention of CS is an important part of integrating DeskVR to professional environments, as its existence greatly reduces use time, productivity, and presence felt in the virtual environment.

Zielasko *et al.* [1] talks about cybersickness and proposes many ways to fight it, such as reducing the overall latency of the system, reducing the field of view (FOV) of the user, making an

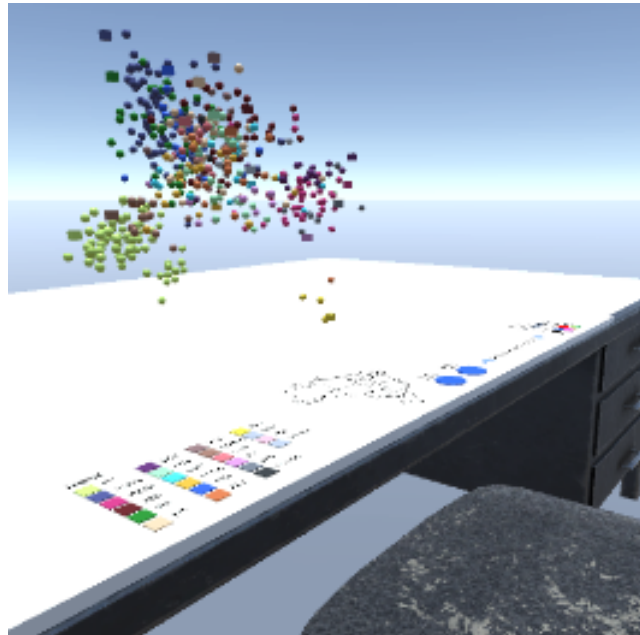


Figure 2.1: Virtual representation of the desk in the virtual world [18]

application that is enjoyable to use, and reducing the factors that may cause cybersickness when the user is using an immersive display for the first time.

Zielasko *et al.* [22] also shows that by creating a virtual representation of the real-world desk, a very dominant part of the real world environment, the effects of CS can be reduced. This can be easily achieved by creating an object with a similar shape and size to the office desk, as seen in 2.1, and then attaching it to the user so that it is always in front of them.

2.1.2 Presence

Presence, in VR, represents the sense of how much a user feels present inside the virtual environment [1]. A lack of presence can cause discomfort in the user and, consequently, it can cause a loss of performance and efficiency.

To create the feeling of presence, it is common to create a virtual body to represent the real user's body. This technique, however, is not feasible for a DeskVR setting as it requires an advanced tracking setup [1]. Hence, inspired by this method, presence in DeskVR can be achieved by creating a virtual representation of the real-world desk (Fig. 2.1). By having this anchor to the real world, while inside the virtual one, the user experiences a more comfortable and efficient way of exploring the environment [18].

2.1.3 Interaction with the environment

While user input can be easily obtained with the use of controllers, much like the ones provided by Oculus or Steam, when it comes to a professional work environment these options are not reliable

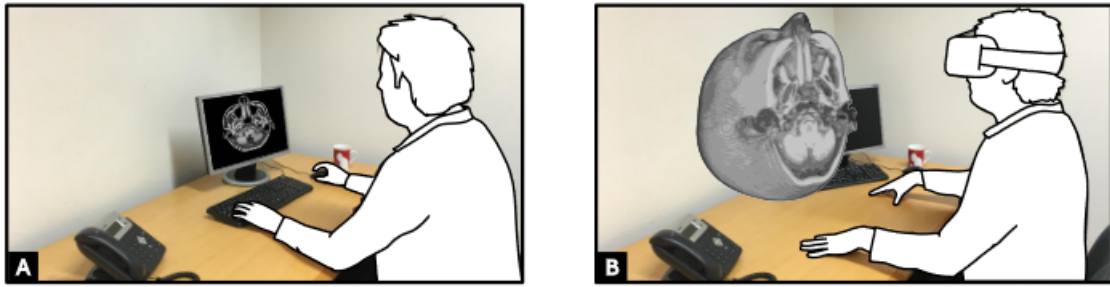


Figure 2.2: A) A typical radiology reading room; B) Vrrrroom's approach combining virtual reality and desktop touch interactions. [15]

as they would require extensive movement from the users. Interactive surfaces have been proven to be useful when interacting with virtual objects, as demonstrated by the work of Surale *et al.* [17], a study in which the authors test the applicability of multi-touch tablets towards object interaction in VR.

Sousa *et al.* [15], attempted to integrate VR with a hospital environment, to help radiologists have a better work performance, removing the problem of improper room conditions. Because "physician resistance to novel systems and technologies is a well-known issue", the authors used an interactive surface, on top of a desk (Fig. 2.2), to allow the practitioners to control the virtual environments.

This study wielded good results in favor of this approach meaning that a possible solution for the problem, described above, would be to either use a touchpad placed on top of the desk, as seen in [15], or use the desk itself [19, 21], by taking advantage of its virtual representation, to allow the user to interact with the virtual environment.

2.2 Object Selection in Immersive Virtual Environments

Object selection in IVEs may seem like a trivial task, but, since it is needed to access more complex features, it is also one of the most important functionalities. Over the years, many techniques for distant object selection have been developed, studied, and analyzed in all kinds of immersive virtual environments (IVEs). These methods are dependent on how the environment is displayed to the user, as a change of perspective could alter the technique's functionality.

We will present and explore object selection techniques, designed for both 3D and 2D representations of the IVEs, that might be relevant for the work of this thesis.

2.2.1 Object selection in 3D representations of the IVE

Since VR has been a topic of discussion and a subject of research for many years, object selection in these types of environments has been explored in detail. Therefore, many selection methods

have been created. Argelaguet and Andujar [3] conducted a survey in which they present, review, and discuss multiple out of reach object selection methods.

Many of these techniques are based on the RayCasting object selection method, where a laser is shot from a specific point, usually the user's hand or eyes, in a specific direction, controlled by the person, and in which the selected object is the first object that the laser collides with [2, 11]. Due to its easy implementation, by the developers, and use, by the users, RayCasting is used in many VR applications as a solution for distant object selection, for example, in Scaled-Homer [20] and in the object manipulation techniques proposed by Lee *et al.* [10]. A clear problem with this method, however, is its requirement for high precision and accuracy [9].

Due to these requirements, the performance of the raycast is impacted when targeting small and distant objects. In order to combat these restrains, Baloup *et al.* [4] proposed the RayCursor, an improvement on RayCasting that uses a user controlled cursor, displayed on the ray, to select objects in a virtual environment.

Baloup *et al.* introduced two version of the method: the manual RayCursor, and the semi-auto RayCursor. In the manual RayCursor the user controls the position of the cursor along the ray, highlighting the closest object to it. In the semi-auto RayCursor, the ray, initially, works like normal RayCasting, positioning the cursor in the intersection between the ray and the first intersected object, highlighting it in the process. After this step, the semi-auto RayCursor works the same way as the manual RayCursor, highlighting the closest object to the cursor. In both cases, the highlighted object is selecting by pressing a confirmation button.

Additionally, the technique tests a variety of gain functions, to control the movement of the cursor along the ray, as well as a few of ways of providing visual feedback about the selection to the user, including highlighting the nearest target to the cursor, displaying a rope, represented by a white line, between the nearest target and the cursor, and displaying a 3D bubble, centered in the cursor, that contains the nearest target object (Fig. 2.3).

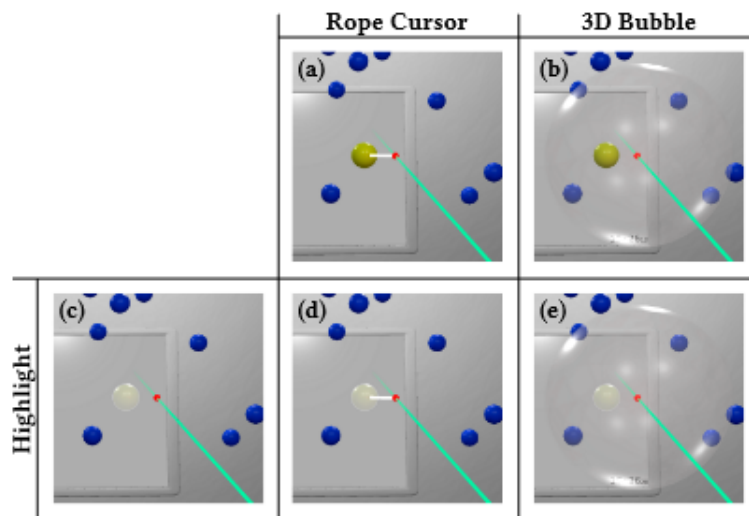


Figure 2.3: RayCursor with different visual feedbacks [4]

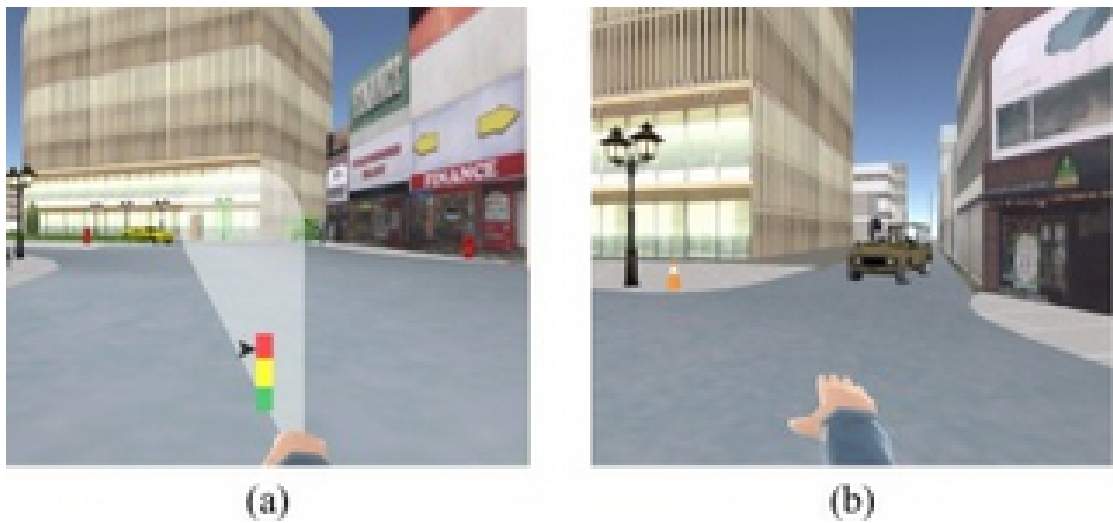


Figure 2.4: PRECIOUS technique [12]: (a) selection cone intersecting various objects, (b) refinement phase, moving the user closer to the objects

There are also some techniques that use the ConeCasting method. Similar to RayCasting, ConeCasting projects a cone from the user to select the target object. Due to its larger selection volume, ConeCasting has the setback of selecting multiple objects at the same time. Consequently, techniques that use ConeCasting need ways to refine the selection. As a solution for distant object selection, Mendes *et al.* developed PRECIOUS [12] (Fig. 2.4), a selection method that uses ConeCasting with iterative refinement. This technique allows its users to perform an accurate object selection by moving them towards the objects selected by the cone. This process is then repeated until only one object is selected, or the selection is canceled by the user. Another out of reach object selection technique that uses ConeCasting is the Shadow Cone Selection technique, proposed by Steed and Parker [16]. This approach begins by selecting all the objects that are inside of the cone and, by moving the cone around, objects that leave the selection volume are deselected. The refinement is achieved by removing all the initially selected objects except the desired one.

Casting techniques are very popular due to their simplicity and convenience, however, there have been studies on the selection of distant objects that do not rely on them. The Go-Go Interaction technique [14] uses an arm growth metaphor that allows the user to "change arm length at will" [14] to grab the desired object. In other words, the method allows its users to extend their virtual arm, making it longer, in order to interact with objects that were initially out of reach.

Another selection technique, that does not rely on casting, was proposed by Dalsgaard *et al.* [6], with the use of motion capture the authors created a database containing features based on how people naturally point. Given this database, the method is able to predict the object, inside a virtual scene, that the user is pointing at.

3D IVE representations are not unique to virtual reality. With specialized hardware it is possible to display images in a true 3D space. Such is the case of the volumetric display, a device that



Figure 2.5: Example of a volumetric display

allows for a visual representation of an object in 3D by emitting and scattering light (Fig. 2.5).

Grossman and Balakrishnan [8] studied this environment and developed four different methods to select an object presented inside the device - Depth Ray, Lock Ray, Flower Ray, and Smart Ray. In all the methods, a ray is cast through the entirety of the display, selecting multiple objects along the way. Hence, the difference between the many techniques is the way in which they refine their selection.

In Depth Ray and Lock Ray, refinement is done by adding a sphere to the middle of the ray, which chooses the intersected object closest to it. While in Depth Ray, the user controls the ray and the sphere at the same time (Fig. 2.6a), Lock Ray separates the selection phase from the refinement phase, allowing the user to have a better control of the position of the sphere (Fig. 2.6b).

Flower Ray performs its refinement by pulling all the selected objects towards the user and displaying them in a flower-like pattern. The user is then able to perform a second raycast to select the desired object (Fig. 2.6c).

Finally, the Smart Ray method adds weights to the different objects. These weights are updated based on their proximity to the ray. The user casts the ray towards the desired object from multiple angles over a specific length of time, adding to the objects' weights. In the end, the object with the highest weight is the selected one (Fig. 2.6d).

Even though all these techniques showed promising results, the Depth Ray was the most successful. By displaying a small sphere along the ray, that the user could move, it was able to more accurately define which object was intended for selection.

2.2.2 Object selection in 2D representations of the IVE

When an IVE is shown on a 2D surface, like a computer screen or a table, it possesses different characteristics from 3-dimensional IVEs and, consequently, previously mentioned object selection techniques might not work as intended. While in a 3D representation the user can rely on their

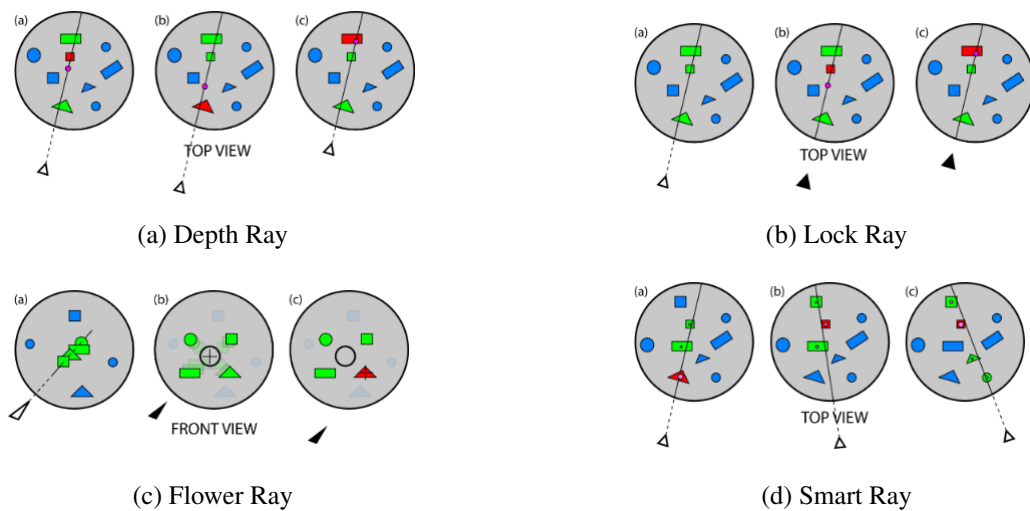


Figure 2.6: Selection techniques proposed by Grossman and Balakrishnan [8]

depth-perception to know where an object is, in a 2D one the user must use other factors, for example, the size of the objects since a smaller size usually corresponds to a farther away object.

In this type of scenario, to select a specific object the user simply needs to touch it, either with their hand, in the case of an interactive surface, or with the mouse cursor, in the case of a computer screen. The problem occurs when the desired object is displayed on areas of large displays that are out of the person's reach. In these cases, it becomes impractical to select the object as it would require significant physical movement.

To overcome this obstacle, Bezerianos and Balakrishnan [5] developed *The Vacuum* (Fig. 2.7), an object selection technique that pulls distant objects, inside an arc of influence that spans out to the edges of the display, closer to them. This arc originates from a point that the user selects and is created after they drag their cursor beyond a predefined threshold.

In the same context, the *TractorBeam* selection method [13] was proposed as a way to select both nearby and distant objects on a tabletop display. With the use of a stylus, the user can simply select an object by touching it, if it is within distance, or by casting a virtual ray, using the pen as a pointer, when the object is far away.

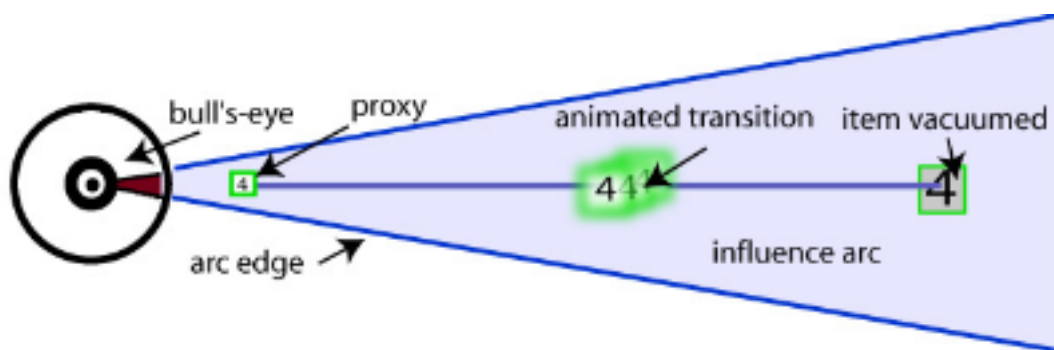


Figure 2.7: The Vacuum [5]

When scenarios contain a large number of cluttered objects, it can be hard to select a specific object without proper refinement techniques. Kopper *et al.* introduced SQUAD (Sphere-casting refined by QUAD-menu) [9], which uses progressive refinement, as a proposed solution to this problem. With SQUAD (Fig. 2.8), the user can specify a volume limited by a sphere that contains the target object. The group of selected objects would then be displayed in front of the user, divided into 4 smaller groups. Subsequently, the user can select the group that contains the target object, repeating this process until only the desired object is left.

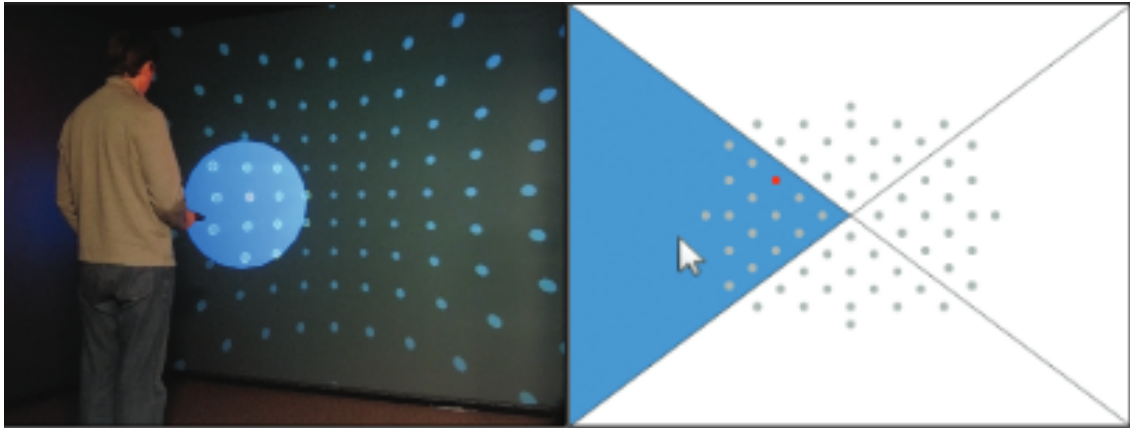


Figure 2.8: Left: Experimental setup with sphere-casting. Right: quad-menu stage of SQUAD in the experiment. [9]

2.3 Discussion

In the beginning of this chapter, we analyzed several works that studied the concept of DeskVR, the challenges it presents, and the techniques made to overcome them. We noticed that there exists a relationship between cybersickness and presence, and how the former negatively impacts the latter. We also learn that, while interaction techniques that use a VR controller can be used in a DeskVR scenario, they are not optimal as their use may require unnecessary movements from the user. For that reason, many studies produced the idea to use touch surfaces to interact with the virtual environment.

All object selection techniques mentioned in the previously are essentially different from each other. However, there exist some common characteristics between some of the methods. In this section, we compare the many selection methods according to their input and perceived space, the selection tool used by each technique, and, finally, the provided refinement, if needed. The results of this discussion can be seen in Table 2.1.

Starting with the input and perceived spaces, these can be distinguished into two branches, depending on if they are a 2D space or a 3D one. The input space represents how the user interacts with the environment. Techniques that have a 2D input space can either use an interface on a computer screen with a mouse and keyboard input or use touch input on a touchpad surface [5, 13].

For methods that use a 3D input space, interaction is done with tracking of the user's hands or head, the use of specialized controllers, or motion sensors [4, 6, 8, 9, 11, 12, 14, 16]. As for the perceived space, defined as the space in which the IVE can be displayed or, in other words, how the user sees the environment, it can be 2D, if the IVE is constrained to a 2D display, such as a tabletop or a computer screen [5, 9, 13], or it can be 3D, if the display is done via the use of an HMD or a volumetric display [4, 6, 8, 11, 12, 14, 16].

After analyzing the input and perceived space of each technique, we notice that while techniques that have a 3D input space can work on either a 2D or 3D perceived space, the same does not apply to techniques with a 2D input space, as these only work on a 2D perceived space.

The selection tool refers to how the method defines which object is to be selected. Some of the methods use RayCasting, shooting a ray in a specific direction and selecting the first object hit, either directly [8, 11, 13] or with some modifications to aid the selection process, by adding a user controlled adjustable sphere to the ray [4, 8, 9]. Others rely on ConeCasting, similar to RayCasting but instead of shooting a ray the user casts a cone to perform the object selection [12, 16]. When working on a 2D input space, it is possible to select out of reach objects with the use of an arc of influence, a 2D version of ConeCasting [5]. It is not necessary to rely on some type of casting. By extending the user's virtual arm, they can simply select the object directly with their virtual hand [14]. Finally, out of reach object selection can also be made by predicting which object the user is pointing at [6].

Refinement is only required when the selection process returns more than one object and, therefore, is not needed in techniques that select a singular object [11, 13, 14]. In these situations, a mechanism is needed to differentiate which one is the target object. The refinement process of each technique, if available, can be divided into Manual, Behavioral, and Heuristic [3]. Manual refinement is done when the user has to decide which of the selected objects is the desired one [4, 5, 8, 9, 12]. The behavioral refinement happens when the selection technique takes the user's actions before the selection into consideration [8, 16]. Finally, some techniques employ a heuristic refinement when this process is done automatically by the technique [6].

While all of the works proposed, apart from the ones that use a 2D input space [5, 13], could be adapted to work in a DeskVR environment, some of them are not viable for prolonged activity. Techniques that use the user's arm movement require a certain amount of effort from the user to work [6, 14]. Techniques that rely on casting, have a similar disadvantage, since they usually require the use of a controller to point at the desired object. Approaches that adopt behavioral refinement techniques [8, 16], choosing an object based on previous user's actions, require exhaustive and uncomfortable movement from the user, making them unfeasible for a DeskVR environment.

After discussing all the literature work, we found that a technique that uses a 3D input space may not be optimal for DeskVR, as it would require the use to perform large movements of the arm. On the other hand, we also found that selection techniques that operate on a 2D input space do not work on a 3D perceived space. This conclusion presents a challenge for our study, as we will have to find a way to combine both these aspects, a 2D input space and a 3D perceived space.

Table 2.1: Selection Methods

Techniques	Input Space		Perceived Space		Selection tool	Refinement
	2D	3D	2D	3D		
Go-go [14]		•		•	Virtual hand	N/A
Model Pointing [6]		•		•	Prediction	Heuristic
RayCasting [11]		•		•	Ray	N/A
RayCursor [4]		•		•	Ray and adjustable sphere	Manual
Shadow Cone Selection [16]		•		•	Cone	Behavioral
PRECIOUS [12]		•		•	Cone	Manual
Depth Ray [8]		•		•	Ray and adjustable sphere	Manual
Lock Ray [8]		•		•	Ray and adjustable sphere	Manual
Flower Ray [8]		•		•	Ray	Manual
Smart Ray [8]		•		•	Ray	Behavioral
SQUAD [9]		•	•		Ray and adjustable sphere	Manual
The Vacuum [5]	•		•		Arc of influence	Manual
TractorBeam [13]	•		•		Ray	N/A

2.4 Summary

In this chapter, we provided a literature review related to the focus of this study. We saw the challenges that a DeskVR scenario can bring, and, in some cases, how to overcome them. Additionally, we also examined many object selection methods created for IVEs and how they differ from each other.

We notice that, while there is a large number of works both in DeskVR and in object selection in IVEs, there are none that combine them.

Chapter 3

The TouchRay

As previously mentioned, we are searching for a way to select objects that are out of the user's reach in a DeskVR environment. Our problem emerges from a lack of selection methods and techniques specifically developed for this kind of environment. For that reason, we created the TouchRay, an object selection technique that allows the user to select simple objects present in the scene but also, by traversing an object's hierarchy tree, allows the selection of more complex objects.

3.1 Design

This technique is an adaptation of the Ray Casting method adapted to be used with a touch surface and, therefore, uses a ray that is casted by the user to select an object.

Furthermore, to allow for the selection of partially or totally occluded objects, the TouchRay is divided into two steps: the Candidate step, where the ray is casted, allowing the user to determine a group of candidate objects; and, if necessary, the Refinement step, in which the user finalizes the selection by choosing one of the candidate objects selected in the previous step.

Each step is divided into three phases, depending on the type of input given by the user to the interactive surface: placing the finger on it, moving the finger across it, and removing the finger.

3.1.1 Candidate Step

The candidate step begins the moment the user places a finger on the touch frame. This action creates a selection ray, which is then used to select an object present in the scene. The ray begins under the user's view, as to not disturb them while it is being displayed, and extends forwards, towards the scene, in the same direction the user is looking at (Fig. 3.1).

Furthermore, if the origin of the ray was positioned closer to the camera, the ray would either cover the user's view and, consequently, the scene would not be visible, or it would not be rendered, and the user would not be able to see it. Additionally, the ray is independent from the user,

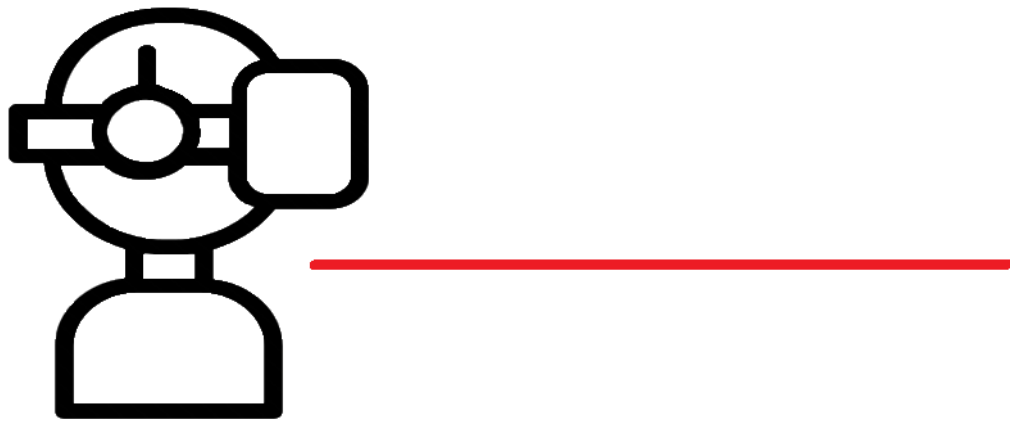


Figure 3.1: Visual representation of the ray

which allows them to move their head freely without interfering with the direction and positioning of the ray.

By moving the finger across the touch surface, the user can rotate the ray around its point of origin. This way the ray's origin remains in place and only its direction changes. Furthermore, the degree at which the ray rotates is dependent on the speed at which the user moves their finger. Slower movements lead to smaller rotations, and faster movements lead to larger rotations.

When the user removes their finger from the touch surface, all of the current intersected objects are placed in a candidate list, ordered by their distance to the user with the closest one being first, and furthest one last (Fig. 3.2), and the candidate step ends. If there are no candidate objects, the selection process ends and no object is selected; if there is only one object in the list, it becomes the selected object as no further refinement is necessary.

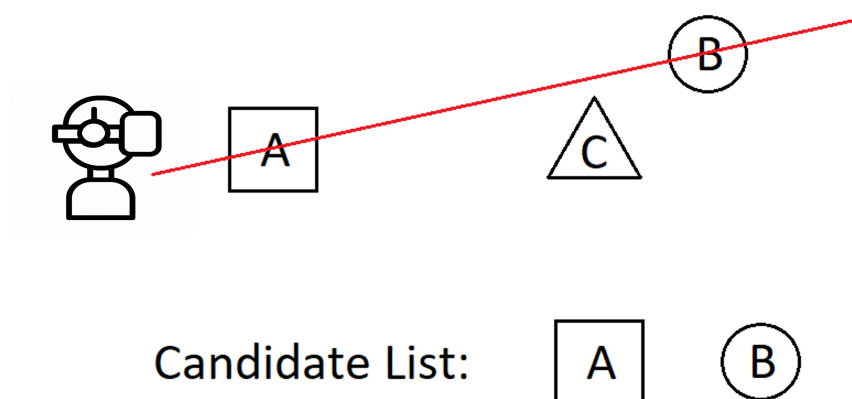


Figure 3.2: Example of the end of the candidate step. Because the ray was intersecting the objects A and B, those objects are added to the candidate list.

3.1.2 Refinement Step

If, in the first step, multiple objects are selected at the same time, either different objects entirely or several sub-components of the same object, further refinement is required. When this happens, the method takes the candidate list given by the candidate step and allows the user to select one of its elements by using distance refinement. In these cases, the refinement step begins immediately after the end of the candidate step.

When this step begins, the closest object to the user, from the candidate list, becomes the top candidate. By moving their finger up and down, the user can traverse the objects in the candidate list, updating the top candidate. By moving the finger forwards, away from themselves, the user traverses the objects according to their order in the list, going from the nearest object to the farthest one. By moving the finger backwards, towards the themselves, the user traverses the objects in the opposite order.

Finally, when the top candidate becomes the desired object, the user can remove their finger from the touch surface and the top candidate becomes the final selected object and the method ends.

3.1.3 Hierarchy

As previously mentioned, TouchRay also supports the selection of sub-components of more complex objects, large objects that are composed of smaller, simpler objects (a chair can be decomposed into legs, seat, and backrest), forming an object hierarchy. This hierarchy can be represented in the form of a hierarchy tree, where a group of simple objects and/or smaller groups of objects is called a *node* and the simple objects themselves are called *leaves*. The top node of the tree, representing the entire complex object, is called the *root*. An example of an hierarchy tree can be seen in Fig. 3.3.

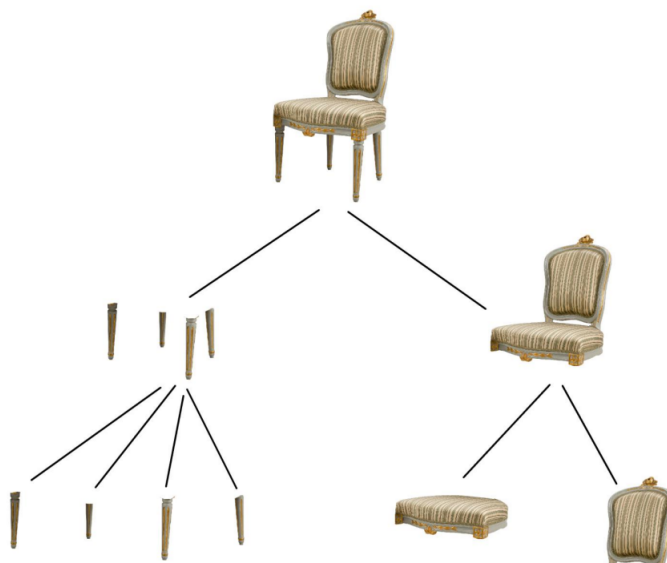


Figure 3.3: Example of an object hierarchy tree

For each object in a tree, the objects at higher levels are the parents, and the objects in the lower levels are the children.

Our hierarchy selection is achieved by traversing the candidate list, created at the end of the candidate step, and for each object present in it, checking if there are any nodes at higher levels of their respective tree. If it encounters any, they are added to the candidate list, immediately after the object that started the search.

As an illustration, if the ray created in the candidate step intersects one of the legs and the seat of the previously shown chair (Fig. 3.3), the initial candidate list would contain the leg and the seat. The method would then, for each object in the list, check for the existence of nodes in the tree, adding them to the list if they exist. In this example, it would start its search at the leg of the chair, adding the nodes above it to the list (group of legs, and completed chair) and then it would do the same for the seat (seat + backrest and completed chair). The candidate list would then be represented as seen in Fig. 3.4



Figure 3.4: Candidate list of the chair

As we can see in Fig. 3.4, there are objects that appear more than once in the list (completed chair). To prevent this from happening, before adding an object to the list, the method checks if it is already present in it. If it does exist in the list, the first instance of the object is removed, and the new one is added to the list. By doing this, we can assure that any node of any level of the tree will always appear after all of its children. The final candidate list would then be represented as seen in Fig. 3.5.



Figure 3.5: Final candidate list of the chair

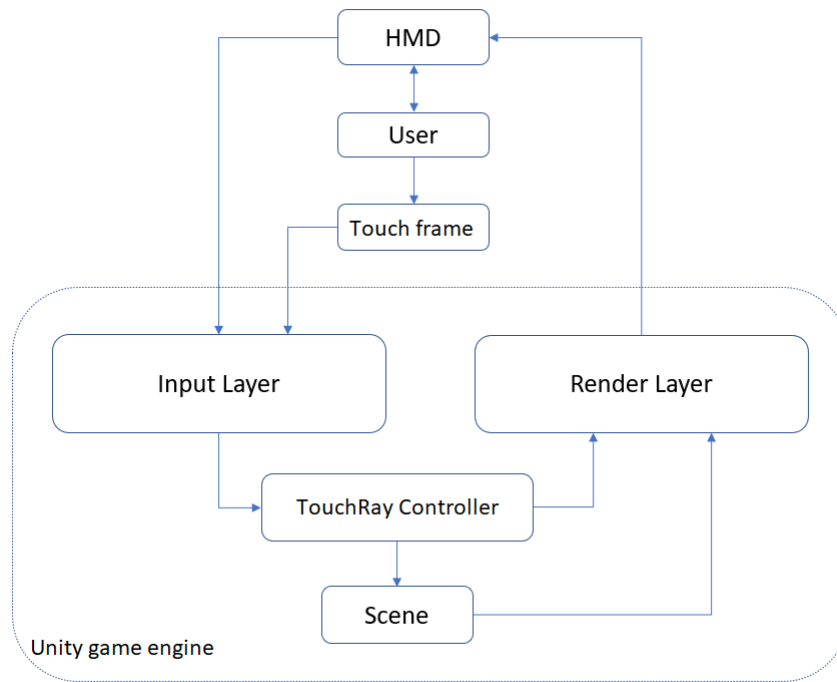


Figure 3.6: Architecture of the TouchRay

3.2 Implementation

In order to put our technique into practice, we used the *Oculus Rift S* and a 32 inch infrared multi touch frame to manage both the inputs given by the users and the outputs perceived by them. Additionally, we used the C# language together with the Unity game engine, with the OpenXR plugin, to develop the code to transform the given inputs to relevant outputs. The architecture for the implementation can be seen in Fig. 3.6.

3.2.1 Visual feedback

Our implementation of the TouchRay relies on visual feedback to tell the user what is happening.

During the candidate step, whenever an object is intersected by the selection ray, its color changes to dark green, as seen in Fig. 3.7. If the object stops being intersected by the ray, it is given its original color back. This change makes it easier to identify intersected objects and separate them from the rest.

If there is the need for the refinement step when the candidate step ends, all objects in the candidate list have their colors changed to yellow, except for the one closest to the user. The closest object becomes the top candidate and is given a different color, bright green, to differentiate it from the other candidates. Furthermore, because some of the candidate objects can be partially or totally occluded, all candidate objects, apart from the top candidate, become translucent (Fig. 3.8). This change allows the user to see the top candidate even in the case of it being occluded.

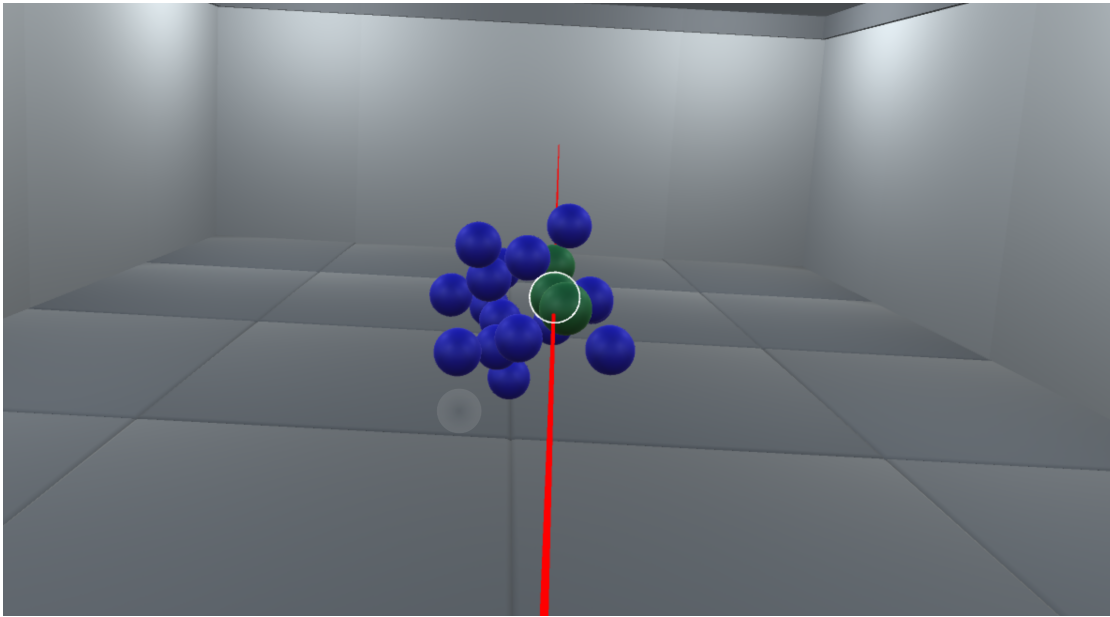


Figure 3.7: Candidate Step: Intersected objects change color for visual feedback

When the user moves their finger up and down, the top candidate is updated and the colors of the objects change accordingly (Fig. 3.9).

3.2.2 Hierarchy

An important part of the TouchRay technique is the ability to select a complex object by intersecting parts of it and climbing the hierarchy tree, as explained in section 3.1.3. For that reason, the function that handles that feature is also relevant.

The following function, *CheckGroup*, written in C#, is responsible for checking if a given object, *gameObject*, is a group of objects and, consequently, part of a hierarchy tree. If the object in question is a group of objects, a parent in the hierarchy tree, the function then checks if the object

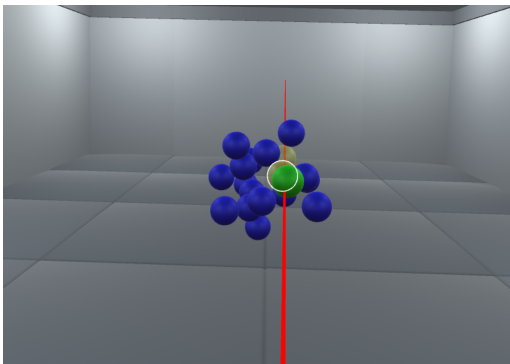


Figure 3.8: The closest object is green while the others are translucent

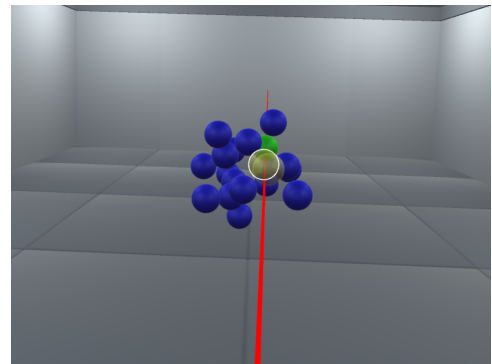


Figure 3.9: After moving the finger the top candidate changed

already exists in the candidate list, removing it if it is present, and finally adds the object to the end of the list. This process guarantees that an object will always appear in the list after all of its children. Finally, if the object itself has a parent, a larger group of objects, the function is called again with the parent.

```
void CheckGroup(GameObject gameObject)
{
    if (gameObject.tag == "Group")
    {
        if (candidateList.Contains(gameObject))
            candidateList.Remove(gameObject);

        candidateList.Add(gameObject);
    }

    GameObject parent = gameObject.transform.parent.gameObject;

    if (parent.tag != "Root")
        CheckGroup(parent);
}
```

3.3 Summary

In this chapter, we introduced the TouchRay, an object selection technique designed to work in a DeskVR scenario.

By using a touch surface as input, the user can select any object in the scene by casting a ray onto it and intersecting it with the desired object. In the case multiple objects are intersected, the method offers a refinement step in which the user can specify which object they want to select.

Additionally, by searching the hierarchy tree of an object, the TouchRay allows its users to select complex objects by intersecting one of its sub-components.

Chapter 4

Method Evaluation

In order to correctly evaluate the effectiveness of our solution, we performed a set of experiments where we asked participants to perform certain tasks, to measure the necessary metrics, quantitative and qualitative, to evaluate the solution.

In order to compare the results of our solution, we implemented two other methods from previous articles: the Depth Ray [8], and the RayCursor [4]. We chose these methods as they have proven to be effective techniques for object selection.

4.1 Setup

For the hardware, we used the *Oculus Rift S* as our virtual reality headset and a 32 inch infrared multi touch frame as our touch surface.

In all experiments, the participant was seated at a desk with the touch frame on top of it (Fig. 4.1). Due to the high sensitivity of the touch surface, it was turned off when not in use, as to not interfere with the other methods that do not need it.



Figure 4.1: A participant of the experiment

4.2 Additional Methods

As previously mentioned, we implemented two other object selection methods, so that we could make a comparison between them and our proposed solution. Like our implementation of the TouchRay, these techniques were implemented using the Unity game engine with the OpenXR plugin and are designed to work with any virtual reality rig consisting of at least a headset and a controller.

Based on previous literature, the two implemented methods were:

- Depth Ray [8]
- RayCursor [4]

While both of these methods were designed to work in 3D environments, they required some changes to make them compatible with our available hardware. Both of these methods were adapted to work with the *Oculus Rift S* controller and, therefore, have the same input scheme (Fig. 4.2).

When the user presses and holds the trigger button, a ray appears, with its origin on the remote, going towards the direction the user is pointing. Additionally, a depth marker, represented by a red sphere, will appear attached to the ray. The user can change the position of the sphere along the ray by moving the joystick present on the remote. Finally, when the desired object becomes bright green (highlighted), the user can press the grip button to confirm their selection.

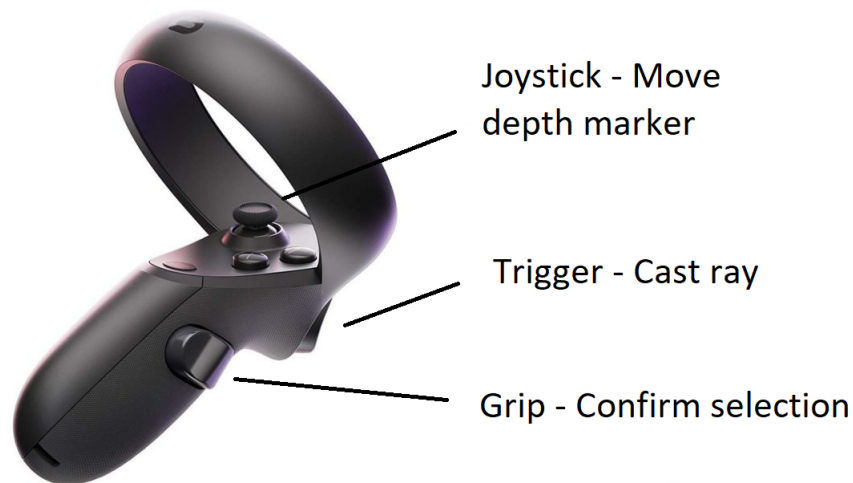


Figure 4.2: Oculus Rift S controller

Because they have the same controls, the difference between each of the methods is on how they choose which object is selected.

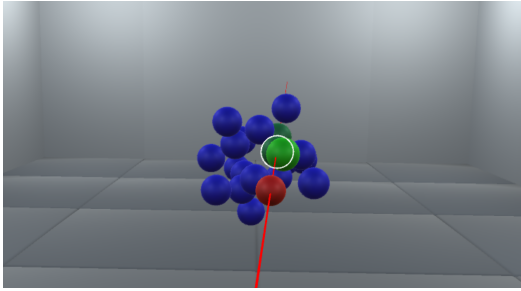


Figure 4.3: When the ray first intersects with multiple objects, the closest one becomes the candidate object.

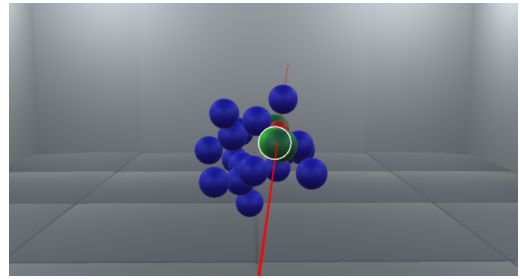


Figure 4.4: After using the joystick to move the depth marker the candidate object is updated.

4.2.1 Depth Ray

In our implementation of the Depth Ray, when the ray intersects multiple objects, the closest one to the depth marker becomes the candidate object. With the use of the joystick, the user can move the marker along the ray, and if a different object becomes the closest object to the marker, it becomes the candidate object. When the user confirms the selection, the candidate object becomes the selected object. If the ray only intersects one object, that object is always the candidate.

In order to provide visual feedback to the user, all objects that are being intersected by the ray, have their color changed to dark green, except for the candidate object, that has its color changed to bright green. If the user moves the marker along the ray and a different object becomes the closest one too the marker, the colors are updated accordingly.

Furthermore, because the Depth Ray was initially developed to work on volumetric displays (Fig. 2.5), a small representation of a 3D environment, the method had two problems when integrated to virtual reality that needed to be solved:

1. Because the virtual world, in the volumetric display, is limited, the depth marker initially appeared in the middle of the ray. This is not possible in VR, since the virtual world can be infinite.
2. Due to the small size of the volumetric display, the movement of the depth marker was defined by the absolute movement of the user's hand. This is not viable in VR for the same reason as the previous point.

To solve both problems, we place the depth marker at a fixed initial position, near to the controller, and then use the joystick to dictate its movement, by using a linear mapping between the input given by joystick (value between -1.0 and 1.0), a predefined sensitivity (20) and the speed of the marker:

$$velocity = 20 * input$$

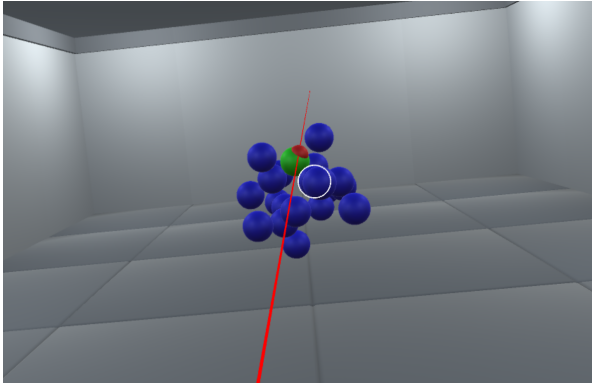


Figure 4.5: When the ray intersects an object for the first time, the marker is moved to the point of intersection.

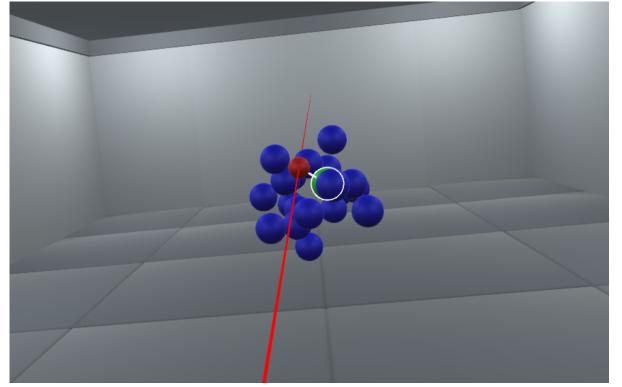


Figure 4.6: After the first intersection, a white line connects the marker with its closest object.

4.2.2 RayCursor

When implementing the RayCursor, we decided to use the semi-auto version of the method, since it showed the best results in the experiments conducted in its article [4]. When the ray intersects with an object for the first time, the depth marker is positioned at the point of intersection (Fig. 4.5). The user can then move the ray and the closest object to the marker becomes the target object. In order to provide visual feedback to the user, the target object becomes bright green and a white ray, acting as a rope, is added connecting the marker to the closest object (Fig. 4.6). This type of feedback was used as it was proven to have the better results [4].

Even though the original RayCursor implements gain functions to control the velocity the cursor moves along the ray, they were not required in our implementation as the velocity could be controlled with the use of the joystick present in the controller.

4.3 Methodology

Each experiment was divided into three parts, each corresponding to one of the implemented methods: TouchRay, RayCursor and Depth Ray.

Before beginning the experiments, we used a Latin square design to determine the order in which the users would use the implemented methods. This way, we could guarantee that each combination was evaluated, as well as reduce the bias that might be caused by testing the methods in a specific order.

For each of method, a small presentation detailing its features and how to use it was given to the participant. Additionally, the user was also given 5 minutes in a test setting, resembling an office (Fig. 4.7), for them to try out and adapt to the method before the trial. This setting contained an object with a white outline, resembling the tasks, that the user could select. Upon a successful selection, the outline would move to a different object.

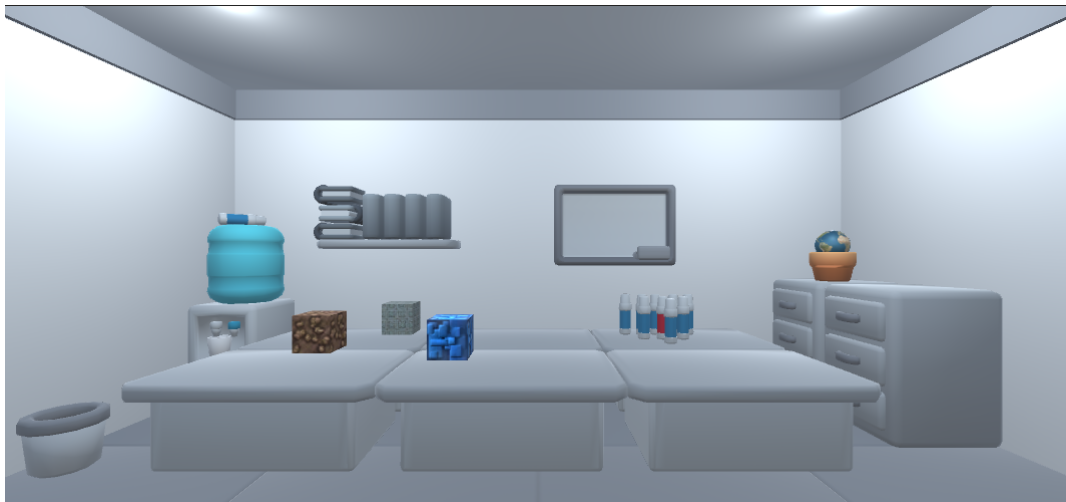


Figure 4.7: Scenario used for adaptation

During the trial, the user was asked to perform a series of tasks as explained in 4.4. After completing all the tasks regarding a certain method, the user is asked to answer a questionnaire about their experience while performing the tasks.

Finally, after completing all the tasks for all methods, the participant was given an environment containing a complex object and asked to give their opinion on two different versions of our hierarchy selection method.

4.4 Tasks

The trial consisted of four tasks that were repeated for each of the methods. In each task, the user was presented with 20 blue spheres, one of which had a white outline. Each task has a different configuration of spheres that depends on the distance between the spheres and the user and the density, distance between each sphere (Fig. 4.8). The four tasks are thus defined as:

- DisperseClose: spheres are close to the user and distant from each other;
- DisperseFar: spheres are far away from the user and distant from each other;
- AggregateClose: spheres are close to the user and close to each other;
- AggregateFar: spheres are far away from the user and close to each other.

In each task, the user was asked to select the sphere with the white outline. After a successful selection, the outline was removed and added to a new sphere. The process was repeated until a total of 50 selections were accomplished, resulting in a total of 600 selections per user (3 methods x 4 tasks x 50 selections).

The order in which the tasks appeared was fixed and equal among every experiment, as to make it consistent across every participant. This order consisted of scenarios containing dispersed

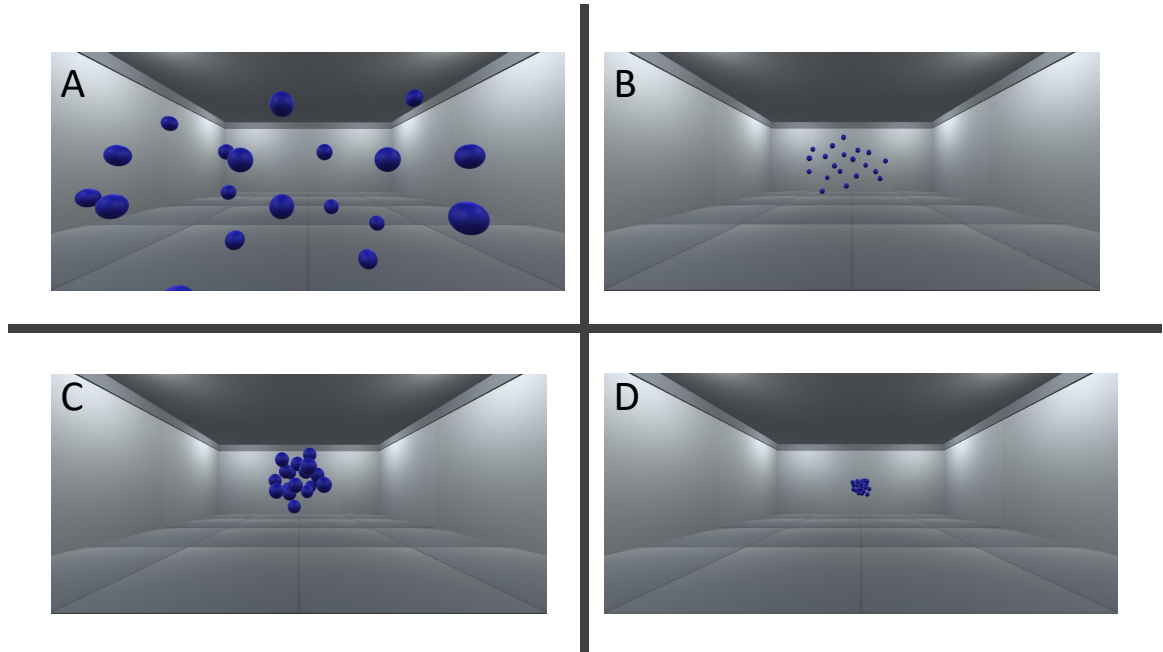


Figure 4.8: Scenarios used in trial. A: DisperseClose; B: DisperseFar; C: AggregateClose; D: AggregateFar

spheres followed by the ones containing aggregated spheres, and scenarios where the spheres are closed to the user followed by the ones where the spheres are distant from the user.

DisperseClose $\rightarrow$ DisperseFar $\rightarrow$ AggregateClose $\rightarrow$ AggregateFar

After completing all of the required selections, and answering the corresponding questionnaires, the user was shown a scenario containing the engine of a rocket¹ (Fig. 4.9), and was asked to select a highlighted section of the object, using two versions of our hierarchy selection method. In the first version, the user could select a simple object and move their way up the hierarchy tree towards more complex objects, which we called GroupLast (GL), and in the second version, the opposite would occur: the user could select the entire object and move their way down the tree towards more simple objects, which we called GroupFirst (GF).

4.5 Metrics

In order to facilitate the comparison between methods, we collected values from each experiment. We collected both quantitative and qualitative data from the users.

For the objective data, we measured the time (in milliseconds) and number of attempts, needed to make a correct selection, the user took to select each ball in all the scenarios, and the total time (in milliseconds) and total number of selections needed to successfully finish each task. This data was measured and logged automatically by the system.

¹ source: <https://www.cgtrader.com/items/2492616/download-page>

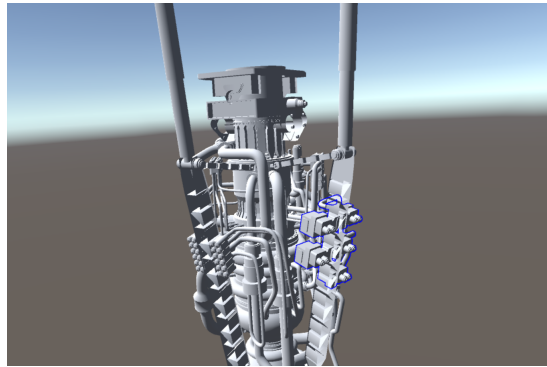


Figure 4.9: Rocket engine used for hierarchy task

The number of attempts increased every time the participant made an incorrect selection. In the Depth Ray and RayCursor methods, this happened every time the user pressed the confirmation button while selecting the incorrect sphere, and for the TouchRay, it happened when the user removed their finger from the frame in the candidate step, if the ray did not intersect any sphere, or in the refinement step, if the select sphere was not the highlighted one.

For the subjective data, we wanted to know how comfortable and demanding the methods were, as well as how easy they were to use. Since this data is subjective and changes from user to user, we used a form containing 6 sentences to which the user answered from 1 (“Strongly Disagree”) to 5 (“Strongly Agree”), depending on which one better described their experience with the method.

The sentences were:

- I think the method was easy to use.
- I think the method was easy to learn.
- I think the method was uncomfortable to use.
- I think the method was mentally demanding.
- I think the method was physically demanding.
- I think I was fast at completing the tasks.

Additionally, after answering all of the questionnaires, the user was asked to give their opinion and a grade, of one to five, to each of the versions of our hierarchy selection method.

4.6 Participants

A total of 18 people participated in the experiments, 16 were male and two were female, two of which were left-handed. All the participants consented in taking part in the experiment.

Out of the 18 participants, 16 were between the ages of 21 and 30, with one participant being younger, between 16 and 20, and one participant being older, more than 50 years old. Of these

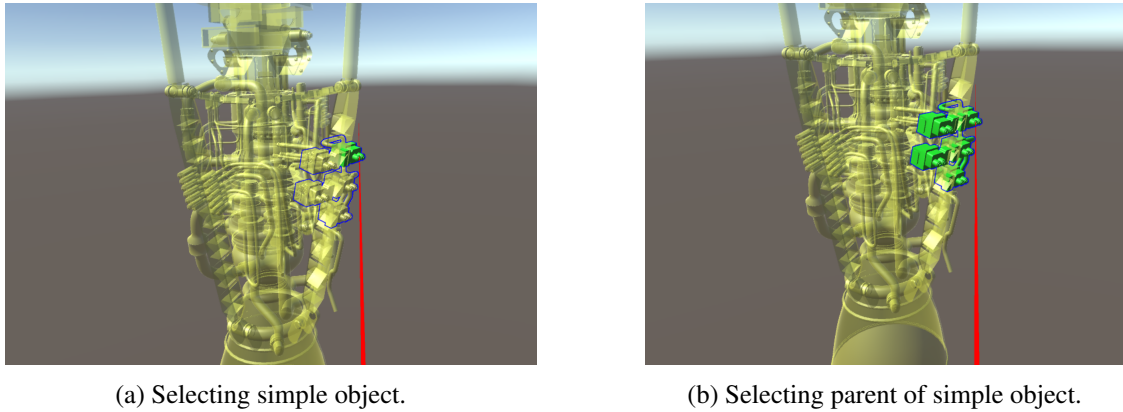


Figure 4.10: Example of Hierarchy selection, GroupLast

participants, 14 had little to no experience with VR, and 15 had little to no experience with 3D modelling and editing software.

At the end of the experience, each participant was given a chocolate bar as a reward.

4.7 Results

As previously mentioned, we collected data, both quantitative (objective) and qualitative (subjective), from each experiment performed. Before analysing the results, we used the Shapiro-Wilk test to see if the data followed a normal distribution. Afterwards, because we want to compare the results of three different techniques, we used the Repeated Measures ANOVA test with a Greenhouse-Geisser correction to find statistically significant differences when the data followed a normal distribution, and the Friedman non-parametric test with Wilcoxon Signed Ranks post-hoc test when the data did not follow a normal distribution. In both cases, post-hoc tests with the Bonferroni correction were used to find which of the methods had statistically significant differences.

4.7.1 Objective Results

For the quantitative results, we measured the time (in milliseconds) taken by each participant to successfully finish each task, as well as the total number of selections required to do so.

4.7.1.1 Completion Time

The total time taken, in milliseconds, to successfully complete each task can be viewed in the box-plot graph present in Fig. 4.11.

For the first task (DisperseClose), a Repeated Measures ANOVA test with a Greenhouse-Geisser correction showed that the time taken with each method has statistically significant differences ($F(1.408, 22.530) = 170.765, p < 0.001$). Post-hoc tests, using the Bonferroni correction, showed us that the TouchRay was statistically significantly slower than the Depth Ray (69.02s,

$p < 0.001$) and than the RayCursor (45.38s, $p < 0.001$), and that the Depth Ray was statistically significantly faster than the RayCursor (23.65 s, $p < 0.001$).

In the second task (DisperseFar), there was also a statistically significant difference between the total time needed for each method, $\chi^2(2) = 28.000$, $p < 0.001$. The median of the total time needed for the TouchRay, Depth Ray, and RayCursor were, respectively, in milliseconds, 88545.495 (82474.964 to 100417.289), 47482.256 (42480.559 to 48707.720), and 71686.356 (62854.097 to 75446.446). Post-hoc analysis with the Wilcoxon signed rank test showed that there was a statistically significant reduction in time needed between the TouchRay and the Depth Ray ($Z = -3.296$, $p < 0.001$), the TouchRay and the RayCursor ($Z = -3.724$, $p < 0.001$), and between the RayCursor and the Depth Ray ($Z = -3.296$, $p < 0.001$).

As for the third task (AggregateClose), we found statistically significant differences between the methods ($F(1.322, 19.824) = 38.772$, $p < 0.001$). After having performed the post-hoc tests, we noticed that the Depth Ray was statistically significantly faster than both the TouchRay (58.27s, $p < 0.001$) and the RayCursor (51.2s, $p < 0.001$), and that there were no statistically significant differences between the TouchRay and the RayCursor (70.67s, $p = 1.000$).

Finally, for the fourth task (AggregateFar), we again used the Repeated Measures ANOVA test and found that the results for each method are statistically significant differences ($F(1.398, 19.575) = 23.184$, $P < 0.001$). Much like the third task, after performing the post-hoc tests, we noticed that the Depth Ray was statistically significantly faster than the other two methods: TouchRay (45.50s, $p < 0.001$), and RayCursor (52.48 s, $p < 0.001$). Additionally, it was noticed that the total time needed to complete a task had no statistically significant differences between the TouchRay and the RayCursor (69.72 s, $p > 0.999$).

After analysing the results, we notice that the Depth Ray and the RayCursor outperform the TouchRay in the first and second task, being able to accomplish both tasks with faster times. This is possibly due to fact that the methods allow their users to easily manipulate the direction of the ray with simple and small movements of the wrist, which allows for faster intersections between the desired objects and the ray, while the TouchRay may require larger movements of the arm.

In the third and fourth tasks, the Depth Ray outperformed the other two methods, with the TouchRay and the RayCursor showing no significant differences between each other. This likely happened, once again, because of the ability to intersect the desired object with small movements of the hand, offered by the DepthRay. However, even though the RayCursor controls its ray the same way as the Depth Ray, it did not perform in the same manner. This decline in performance, in the aggregated scenarios, is possibly due to how the method chooses which object will be selected: while in the Depth Ray the selected object is the intersected object closest to the cursor, in the RayCursor it is the closest one to the cursor.

Even though the TouchRay did not performed optimally in terms of total time needed, it performed more consistently across the different tasks. While in both the Depth Ray and the RayCursor there is a large discrepancy between the time needed to complete the first and second pair of tasks, this does not happen in the TouchRay, as the times are more uniform. This uniformity shows that, were the tasks more complex, the total time needed to complete them would not be

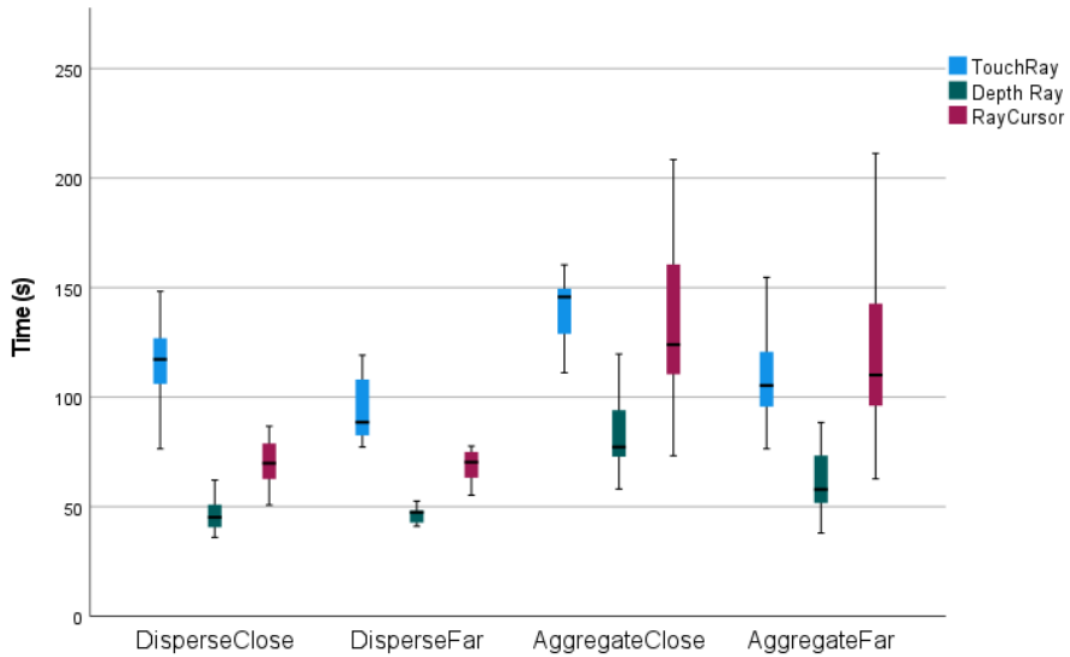


Figure 4.11: Total time to complete each task using the three methods, in milliseconds.

extremely affected, while, on the other hand, the total time required to complete the tasks, using the other two methods, would significantly increase.

4.7.1.2 Number of selections

The number of selections represent the number of times that the user confirmed a selection, either correctly or incorrectly, and can be viewed in the box-plot graph present in Fig. 4.12. Since there were only 50 selections per task, the minimum number of selections possible is 50, with the remaining ones representing incorrect selections. As an example, if a user performed 54 selections in a task, that means that they did four incorrect selections. For that reason, lower values represent fewer errors and, consequently, better performance.

In all tasks, with the exception of the the second (DisperseFar), there were no statistically significant differences between the methods (First task (DisperseClose): $F(1.252, 18.78) = 3.857$, $p = 0.57$; Third task (AggregateClose): $F(1.754, 24.552) = 1.417$, $p = 0.26$; Fourth task (AggregateFar): $\chi^2(2) = 5.148$, $p = 0.076$).

For the second task (DisperseFar), however, we noted a statistically significant differences between methods ($\chi^2(2) = 17.211$, $p < 0.001$). The median total number of selections needed for the TouchRay, the Depth Ray, and the RayCursor were 52.50 (51.75 to 53.00), 62 (57.25 to 69.00), and 55.50 (53.50 to 58.25). After performing the post-hoc tests, it was found that the TouchRay had a statistically significant lower number of selections than the other two methods (Depth Ray:

($Z = -2.807$, $p = 0.005$); RayCursor: ($Z = -2.814$, $p = 0.005$)), and that the RayCursor had a statistically significant lower number of selections than the Depth Ray ($Z = -2.814$, $p = 0.003$).

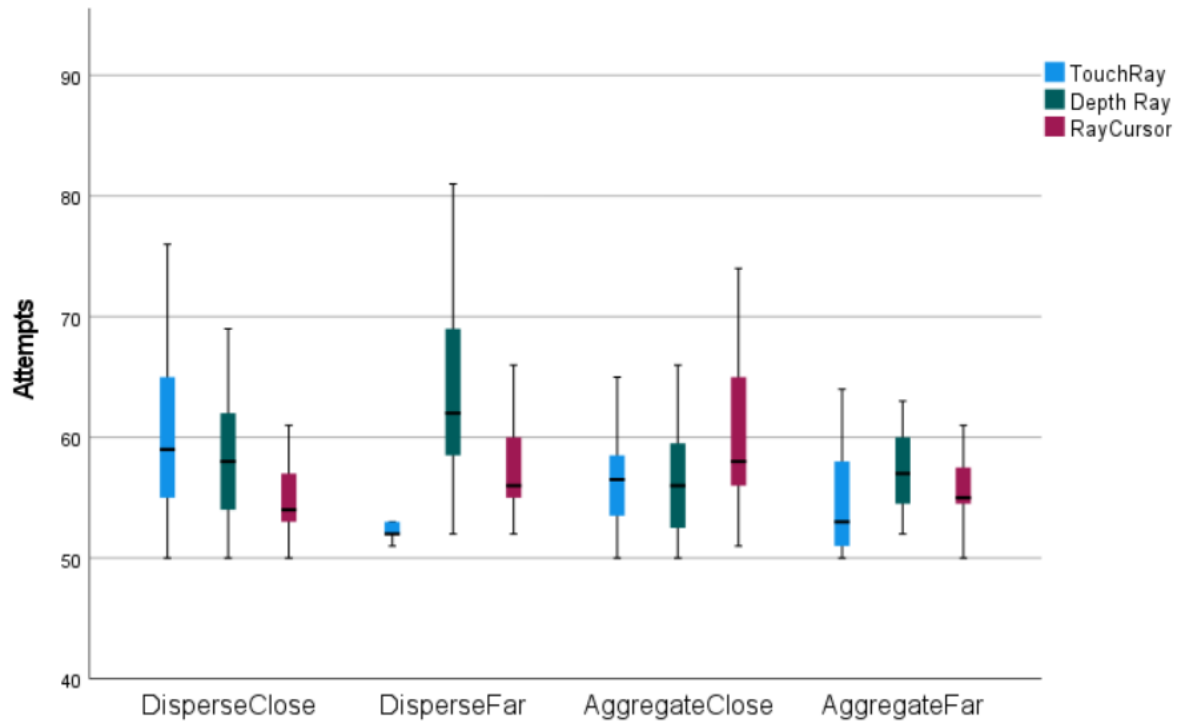


Figure 4.12: Total number of selections needed to complete each task using the three methods

When analysing the data for the total number of selections made, we can see that there was no significant difference between the methods. In the second task, however, we note that the TouchRay had a much lower total number of selections, and therefore less incorrect selections, than the other two methods. This might have occurred due to the need of high precision to select far away objects. Small hand tremors, present in the Depth Ray and RayCursor, can easily remove the ray from the desired object before the user presses the confirmation button, leading to an incorrect selection. These tremors are absent from the TouchRay as a result of the hand being rested on the table, leading to a better control of the ray's direction, which, in turn, results in an increase of precision.

Furthermore, some of the incorrect selections counted in the TouchRay were caused when the user accidentally removed their finger from the touch frame and when the user ran out of room in the touch frame, colliding with one of its walls. Therefore, future improvements to the TouchRay method could reduce the overall number of erroneous selections and, consequently, the total number of selections needed to complete the tasks.

	TouchRay	Depth Ray	RayCursor
Easy to use	4 (1)*	5 (1) [†]	3 (1.25)* [†]
Easy to learn	5 (1)	5 (0)	5 (1)
Uncomfortable	2 (2.25)	2 (2)*	3 (2)*
Mentally demanding	1.5 (1.25)	2 (1)	2 (2)
Physically demanding	1 (1)	1 (1)	2 (1)
Fast	4 (1)*	5 (0.25) [†]	3 (1)* [†]

Table 4.1: Questionnaire answers (media, interquartile range). * and [†] represent statistically significant differences between the methods.

4.7.2 Subjective Results

For the qualitative results, we assumed a non-normal distribution of results, and, therefore, use the Friedman with Wilcoxon Signed Ranks test to check if there are any statistically significant differences between the methods. A table representing the analysis of these results can be seen in 4.1.

When asked if the method was easy to use, it was shown that there was a statistically significant difference between methods, $\chi^2(2) = 17.261$, $p < 0.001$. After running the post-hoc tests, there were found no statistically significant differences between the TouchRay and the Depth Ray ($Z = -0.138$, $p = 0.890$). It was found, however, that the RayCursor was statistically significantly harder to use than the TouchRay ($Z = -3.132$, $p = 0.002$) and than the Depth Ray ($Z = -3.069$, $p = 0.002$).

When asked about how uncomfortable the methods were to use, we found that there was a statistically significant difference between the methods, $\chi^2(2) = 8.857$, $p = 0.012$. Furthermore, post-hoc tests revealed that, while there were no statistically significant differences between the TouchRay and the Depth Ray ($Z = -0.591$, $p = 0.554$) and between the TouchRay and the RayCursor ($Z = -1.826$, $p = 0.068$), the RayCursor was statistically significantly more uncomfortable to use than the Depth Ray ($Z = -2.835$, $p = 0.005$).

Finally, when the user was asked if they thought they were fast at completing the tasks, there were found statistically significant differences in the results, $\chi^2(2) = 20.981$, $p < 0.001$. By running the Wilcoxon Signed Ranks tests, we found that there were no statistically significant differences between the TouchRay and the Depth Ray ($Z = -1.310$, $p = 0.190$), and that the RayCursor was statistically significantly slower than both the TouchRay ($Z = -2.517$, $p = 0.012$) and the Depth Ray ($Z = -3.508$, $p < 0.001$).

After analysing these results, we can see that the participants did not show a significant preference between the TouchRay and the Depth Ray, and that there was an overall preference between these two methods over the RayCursor.

We can see that the TouchRay not only proved to not be a demanding method, both mentally and physically, but also proved to be comfortable making it suitable to use for extended periods of time. It is, however, important to note that if the experiment took more time, more than one hour to complete, the questions related to how demanding each method was might show different results.

Furthermore, after combining and reading the comments given by the participants, we learn that people found it easier to select the objects in the aggregated environments when using the TouchRay rather than the other two methods, due to the high level of precision offered by the TouchRay. This difference in precision between the TouchRay and the other two methods was the most common factors favouring the TouchRay.

4.7.3 Hierarchy

As previously mentioned, we asked the participants to test two versions of our hierarchy selection as described in 4.4, GroupLast and GroupFirst, and then asked them to give each version a grade from 1 to 5, 1 being the worst and 5 the best.

After analysing the results for this tasks, it was found that there was a statistically significant preference for the GroupLast version over the GroupFirst version ($Z = -2.105$, $p = 0.35$) with the median grade being 4.00 (4.00 to 5.00) for the first and 3.00 (2.00 to 4.00) for the second.

However, despite these results, many participants claimed that each version had their own advantages, and that the utility of both versions was highly dependent on the task. The GroupLast version was better if the intention were to select a small part of a specific object, while the GroupFirst version was better if the task were to select the entire object.

4.7.4 Discussion

After analysing all the results, we can finally make a connection between the objective data and subjective one.

Firstly, we can note that despite the RayCursor being faster, in some situations, than the TouchRay, some participants perceived the former as slower and harder to use than the latter, a contradiction that did not occur in the other two methods. Moreover, the opposite can be said about the TouchRay. Even though the TouchRay resulted in longer completion times, participants thought that they were fast at completing the tasks.

Furthermore, even though each method had different, although not significant (between the TouchRay and the Depth Ray), evaluations to how easy they were to use, there were found no differences on the number of wrong selections made between methods, apart from the second task, in which the TouchRay outperformed the others.

The same can be said about the total time needed. Even though the Depth Ray was faster than both other methods in all the tasks, it still showed the same number of erroneous selections. In the second task, however, despite having a much faster time than the other two techniques, it had a larger number of total selections and, consequently, a larger number of wrong selections.

4.8 Summary

After implementing our solution to the given problem, we performed experiments in which we assessed our method against previous existing ones. We provided the setting in which the experiments took place as well as how they were conducted.

After analysing the results of each experiment, we found that our solution, the TouchRay, while slower, in some situations, than the two additional methods used for testing, it was one of the preferred methods, due to it being easy, and comfortable to use.

Chapter 5

Conclusions and future work

Current VR technologies show potential to help various professions gain an increase in their productivity. However, VR is often limited by factors, such as fatigue, cybersickness, and presence, which might have negative implications on the user and their work.

DeskVR might be a solution for this problem. However, it is limited by the amount of physical space available which constrains the amount of movement that can be done by the user. Therefore, to make DeskVR a reality, certain actions, already used in other VR applications, need to be adapted into this new environment to be effectively use, object selection being one of them.

In this dissertation, we developed the TouchRay, an object selection technique, with the intent of helping the wider adoption of DeskVR in more professional settings. After comparing the TouchRay with previously existing methods, the Depth Ray and the TouchRay, we concluded that our method could provide a significant level of precision and, by consequence, a smaller margin of error, at the cost of increased time needed to perform a successful selection. Even though our method did not outperform the other two methods, in some cases, it did not worsen the overall performance. Furthermore, results show that the TouchRay is an easy, comfortable, and non-demanding method to use, and, therefore, a suitable candidate for object selection in a DeskVR environment.

5.1 Future work

Even though the TouchRay was not the optimal method at selecting objects in a VR scene, we believe that small adjustments and improvements to the method could improve its performance. We suggest adjusting the method to work with two separate fingers instead of only one. By allowing the user to use one of the fingers as an anchor point, the second finger can the be used to change the direction of the ray, in the candidate step, and change the top candidate, in the refinement step, without cancelling the technique when removed from the frame. This change could reduce the

number of wrong selections made and reduce the time it takes to make a correct selection, as well as reduce the fatigue caused by using the method.

Additionally, we also suggest the study and exploration of object manipulation in a DeskVR scenario. Now that we have a way of selecting objects in this environment, object manipulation is the next step to make DeskVR possible.

References

- [1] Remain seated: towards fully-immersive desktop vr. In *2017 IEEE 3rd Workshop on Everyday Virtual Reality (WEVR)*, pages 1–6, 2017.
- [2] Ferran Argelaguet and Carlos Andujar. Visual feedback techniques for virtual pointing on stereoscopic displays. In *Proceedings of the 16th ACM Symposium on Virtual Reality Software and Technology*, pages 163–170, 2009.
- [3] Ferran Argelaguet and Carlos Andujar. A survey of 3d object selection techniques for virtual environments. *Computers & Graphics*, 37(3):121–136, 2013.
- [4] Marc Baloup, Thomas Pietrzak, and Géry Casiez. Raycursor: A 3d pointing facilitation technique based on raycasting. In *Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems*, pages 1–12, 2019.
- [5] Anastasia Bezerianos and Ravin Balakrishnan. The vacuum: facilitating the manipulation of distant objects. In *Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 361–370, 2005.
- [6] Tor-Salve Dalsgaard, Jarrod Knibbe, and Joanna Bergström. Modeling pointing for 3d target selection in vr. In *Proceedings of the 27th ACM Symposium on Virtual Reality Software and Technology*, pages 1–10, 2021.
- [7] Simon Davis, Keith Nesbitt, and Eugene Nalivaiko. A systematic review of cybersickness. In *Proceedings of the 2014 conference on interactive entertainment*, pages 1–9, 2014.
- [8] Tovi Grossman and Ravin Balakrishnan. The design and evaluation of selection techniques for 3d volumetric displays. In *Proceedings of the 19th annual ACM symposium on User interface software and technology*, pages 3–12, 2006.
- [9] Regis Kopper, Felipe Bacim, and Doug A Bowman. Rapid and accurate 3d selection by progressive refinement. In *2011 IEEE symposium on 3D user interfaces (3DUI)*, pages 67–74. IEEE, 2011.
- [10] Chia-Yang Lee, Wei-An Hsieh, David Brickler, Sabarish V Babu, and Jung-Hong Chuang. Design and empirical evaluation of a novel near-field interaction metaphor on distant object manipulation in vr. In *Symposium on Spatial User Interaction*, pages 1–11, 2021.
- [11] Sangyoon Lee, Jinseok Seo, Gerard Jounghyun Kim, and Chan-Mo Park. Evaluation of pointing techniques for ray casting selection in virtual environments. In *Third international conference on virtual reality and its application in industry*, volume 4756, pages 38–44. SPIE, 2003.

- [12] Daniel Mendes, Daniel Medeiros, Maurício Sousa, Eduardo Cordeiro, Alfredo Ferreira, and Joaquim A. Jorge. Design and evaluation of a novel out-of-reach selection technique for vr using iterative refinement. *Computers Graphics*, 67:95–102, 2017.
- [13] J Karen Parker, Regan L Mandryk, and Kori M Inkpen. Tractorbeam: seamless integration of local and remote pointing for tabletop displays. pages 33–40, 2005.
- [14] Ivan Poupyrev, Mark Billinghurst, Suzanne Weghorst, and Tadao Ichikawa. The go-go interaction technique: non-linear mapping for direct manipulation in vr. In *Proceedings of the 9th annual ACM symposium on User interface software and technology*, pages 79–80, 1996.
- [15] Maurício Sousa, Daniel Mendes, Soraia Paulo, Nuno Matela, Joaquim Jorge, and Daniel Simões Lopes. Vrrrroom: Virtual reality for radiologists in the reading room. In *Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems*, CHI '17, page 4057–4062, New York, NY, USA, 2017. Association for Computing Machinery.
- [16] Anthony Steed and Chris Parker. 3d selection strategies for head tracked and non-head tracked operation of spatially immersive displays. 2, 2004.
- [17] Hemant Bhaskar Surale, Aakar Gupta, Mark Hancock, and Daniel Vogel. Tabletinvr: Exploring the design space for using a multi-touch tablet in virtual reality. In *Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems*, pages 1–13, 2019.
- [18] Jorge A Wagner Filho, Carla Maria Dal Sasso Freitas, and Luciana Nedel. Virtualdesk: A comfortable and efficient immersive information visualization approach. 37(3):415–426, 2018.
- [19] Pierre Wellner. The digitaldesk calculator: tangible manipulation on a desk top display. In *Proceedings of the 4th annual ACM symposium on User interface software and technology*, pages 27–33, 1991.
- [20] Curtis Wilkes and Doug A Bowman. Advantages of velocity-based scaling for distant 3d manipulation. In *Proceedings of the 2008 ACM symposium on Virtual reality software and technology*, pages 23–29, 2008.
- [21] Robert Xiao, Julia Schwarz, Nick Throm, Andrew D Wilson, and Hrvoje Benko. Mrtouch: Adding touch input to head-mounted mixed reality. *IEEE transactions on visualization and computer graphics*, 24(4):1653–1660, 2018.
- [22] Daniel Zielasko, Benjamin Weyers, and Torsten W. Kuhlen. A non-stationary office desk substitution for desk-based and hmd-projected virtual reality. In *2019 IEEE Conference on Virtual Reality and 3D User Interfaces (VR)*, pages 1884–1889, 2019.