AUTOMATIC SPEED CONTROL FOR NAVIGATION IN 3D VIRTUAL ENVIRONMENT

DOMOKOS M. PAPOI

A THESIS SUBMITTED TO THE FACULTY OF GRADUATE STUDIES
IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE DEGREE OF
MASTER OF SCIENCE

GRADUATE PROGRAMME IN COMPUTER SCIENCE AND ENGINEERING
YORK UNIVERSITY
TORONTO, ONTARIO

APRIL 2016

ABSTRACT

As technology progresses, the scale and complexity of 3D virtual environments can also increase proportionally. This leads to multiscale virtual environments, which are environments that contain groups of objects with extremely unequal levels of scale. Ideally the user should be able to navigate such environments efficiently and robustly. Yet, most previous methods to automatically control the speed of navigation do not generalize well to environments with widely varying scales. I present an improved method to automatically control the navigation speed of the user in 3D virtual environments. The main benefit of my approach is that automatically adapts the navigation speed in multi-scale environments in a manner that enables efficient navigation with maximum freedom, while still avoiding collisions. The results of a usability tests show a significant reduction in the completion time for a multi-scale navigation task.

Acknowledgements

I would like to thank my supervisor Dr. Wolfgang Stuerzlinger for all his help. His knowledge and guidance contributed immensely to the completion of this work. He always found time to discuss my research progress and suggest new approaches, despite a busy schedule.

Thanks to Dr. Petros Faloutsos for taking time out of his schedule to serve on my committee.

I would like to thank my wife, Coco, and my daughter, Hannah, for supporting me in this achievement and for relentlessly pushing me to new heights.

Lastly, I would like to thank my parents for bringing me up in a world filled with books and magic.

TABLE OF CONTENTS

LIST OF FIGURES

Chapter 1
Introduction

Navigation, which is movement in and around an environment, is the most common interactive task performed in three-dimensional virtual environments (VEs). But, often it is a challenging tasks for users as it requires both spatial orientation as well as interaction to actually navigate. Technically speaking, 3D navigation involves two main tasks, namely wayfinding and travel.

Wayfinding is the cognitive component of navigation and relies on spatial cognition. It involves planning, and decision making related to user movement. Tools that aid wayfinding are maps, directional signs, landmarks and so on. Wayfinding plays an important role in virtual environments. For example, in large complex environments an efficient travel technique would not be very useful if the user has no idea where to go. Wayfinding techniques support the execution of the task only in the user's mind, whereas the user has to use travel techniques to actually move the viewpoint.

Travel is the motor component of navigation. It can be defined as the actions that the user makes (through the user interface) to control the position and orientation of his view-point. In virtual environments, travel techniques allow the user to transform his viewpoint through translation and / or rotation and modification of other attributes of movement, such as the speed or in some systems acceleration. In this thesis I present a new travel technique for multi-scale environments.

Three-dimensional virtual environments (virtual reality worlds) are capable of providing rich visual information, but there is strong evidence that visual information on its own is not sufficient if we want to navigate in a VEs as (seemingly) easily as we can navigate in the real world.

1.1 Motivation

Due to the rapid evolution of graphics hardware, interactive 3D graphics has become prevalent on desktop personal computers and even mobile devices. However, efficient and natural navigation in an architectural environment remains a challenging task for a novice user equipped with a 2D mouse.

Since most VEs encompass more space than can be viewed from a single vantage point, users have to be able to navigate efficiently within the environment in order to obtain different views of the scene. In fact, a 3D world is only as useful as the user's ability to get around and interact with the information within it.

Many 3D UIs ignore the aspect of changing the speed of the travel and simply set what seems to be a reasonable constant velocity. This works reasonably well as long as the size and detail of the environment does not vary much. However, this can lead to a variety of problems in multi-scale virtual environments, because a constant velocity will always be too slow in some situations (when the user wants to travel to far away destinations) and too fast in others (when the user wants to investigate geometric detail).

There are many different ways a user can control speed. For example, in gaze-directed steering, the orientation of the head is used to specify navigation direction, so the position of the head (relative to the body) can be used to specify speed. This is called *lean-based* velocity [1, 27, 51]. Similarly, a technique that bases speed on hand position relative to the body [47] integrates well with a pointing technique. A discrete technique for speed control might use two buttons, one to increase the speed by a predefined amount and the other to decrease the speed. All such manual controls have the benefit that it gives the user direct control over the speed of movement.

The main drawback to allowing the user to control speed directly is that it adds complexity to the user interface, as the user has to constantly monitor the speed and adapt it to the current environment. In cases where speed control would be overly distracting to the user, a system-controlled speed technique may be more appropriate. For example, to allow both short, precise movements with a small speed and larger movements with high speed, the system could automatically change the speed depending on the surrounding geometry. This idea is the main motivation for my research.

1.2 Contributions

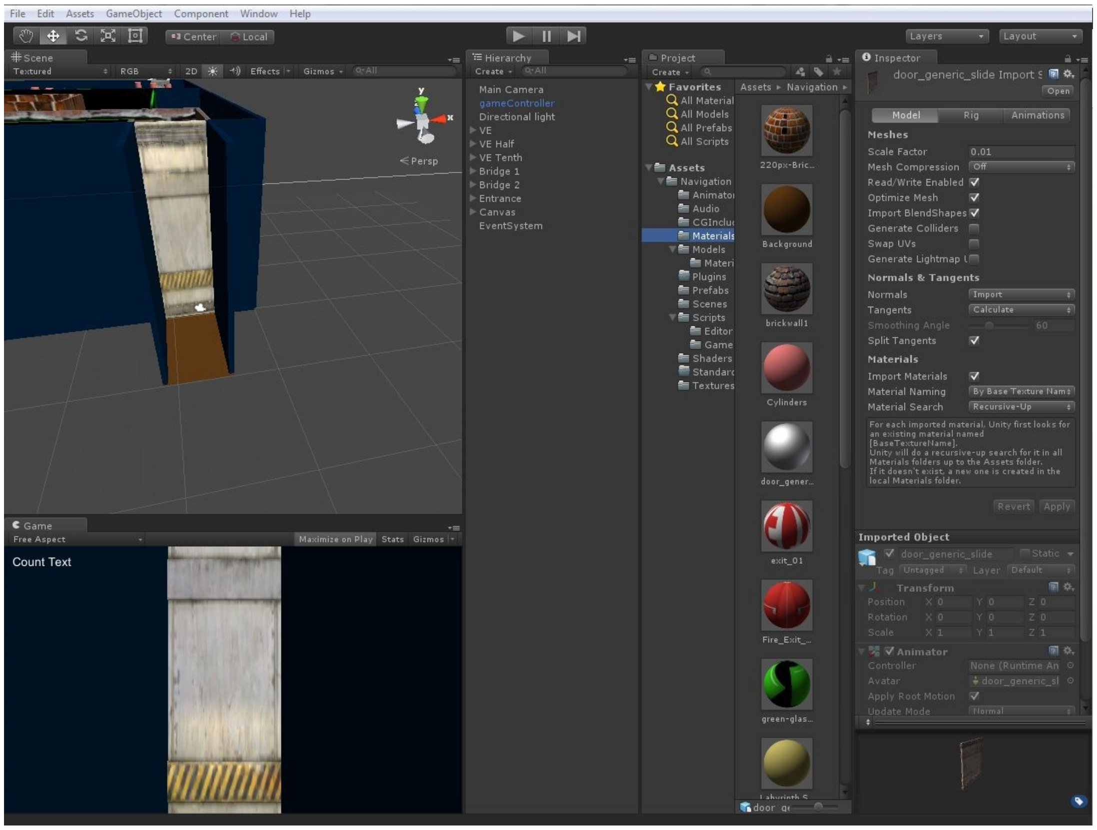
My contributions are:

- A new, efficient, and robust way to automatically adapt the user's speed depending on both the content in the camera's direct view and the surrounding environment with by using a smoothing function (Gaussian or an approximation of Gaussian) that attenuates the effect of geometry in the view direction to control speed.
- A user study evaluating the new automatic speed control relative to speed control via the global minimum described by Trindade [72] and automatic speed control

with ray tracing. The results outline the benefits of the new and improved speed control.

1.3 Virtual Environment (Unity 3D Game Engine)

I used the Unity 3D Game Engine (see Figure 1-1) to create and simulate my multi-scale virtual environments. Unity is a flexible and powerful development platform for creating multiplatform 3D and 2D games and interactive experiences.



Meshes
Scale Factor 0.01
Mesh Compression Off
Read/Write Enabled ✓
Optimize Mesh ✓
Import BlendShapes ✓
Generate Colliders
Swap UVs
Generate Lightmap U
Normals & Tangents
Normals Import
Tangents Calculate
Smoothing Angle 60
Split Tangents ✓
Materials
Import Materials ✓
Material Naming By Base Texture Nam
Material Search Recursive-Up

For each imported material, Unity first looks for an existing material named [BaseTextureName].
Unity will do a recursive-up search for it in all Materials folders up to the Assets folder.
If it doesn't exist, a new one is created in the local Materials folder.

Imported Object
✓ door_generic_slide Static
Tag Untagged Layer Default
Transform
Position X 0 Y 0 Z 0
Rotation X 0 Y 0 Z 0
Scale X 1 Y 1 Z 1
✓ Animator
Controller None (Runtime An
Avatar door_generic_sl
Apply Root Motion ✓
Update Mode Normal

Figure 1-1: The Unity 3D Editor.

The Unity 3D game engine provides standard 3D navigation tools as well as a programmable view to create engaging navigation thorough the virtual environment.

1.4 Navigation and Travel

Navigation is a fundamental human task in our physical environment. According to Bowman et al. [3], we rely on unconscious cognition in the physical environment travel, which is the motor component of navigation. Navigation in 3D virtual environments allows the user to move in the virtual world. Generally, the user can move in all three dimensions by translation, that is moving along the horizontal, vertical and depth axes, and rotation through yaw, pitch and roll motions (see Figure 1-2).

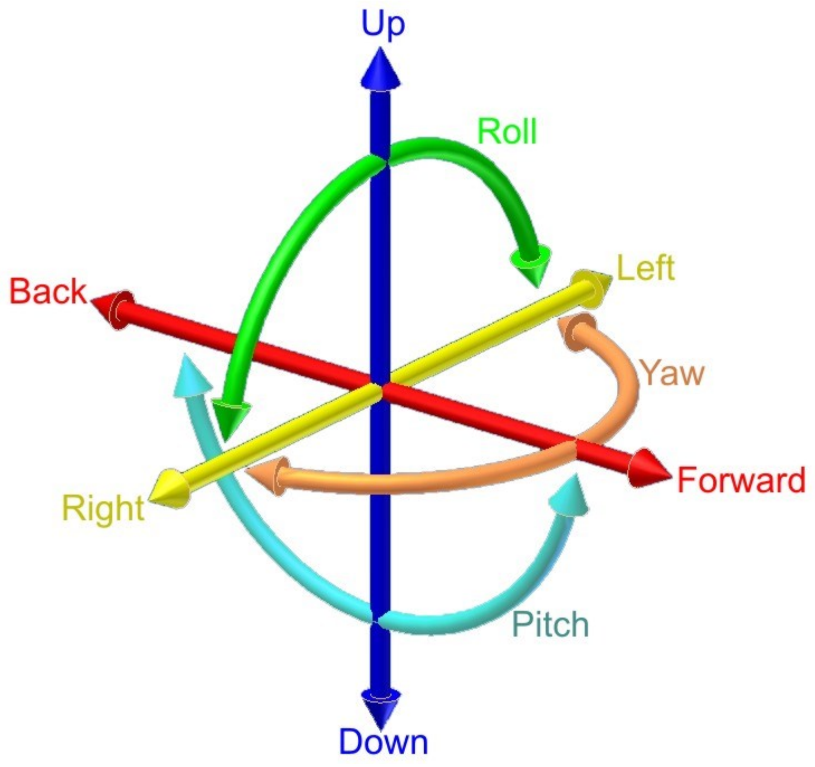


Figure 1-2 Illustration of the six Degrees-Of-Freedom (DOF)

To support these types of movements directly, the user needs to control 6 degrees of freedom (DOF). Yet, such direct control is difficult. Compare the skills required to pilot a car or plane (which many can master) to those required to control a helicopter (which fewer possess). One can also observe such reduced user interfaces in most computer games, where navigation typically involves control over four (4) or fewer DOF (typically rotate left/right and up/down, move forward/backward and both ways sideways, all at predefined speeds).

Chapter 2 Background Literature

In this chapter I will first review navigation-related research and then present work associated with automatic speed control.

2.1 3D User Interface Design

3D user interface design is an essential component of any virtual environment application. A good design is based on a set of recognized principles. In this section, I am reviewing certain common principles for input device design and 3D user interfaces.

There has been some research work about usability of input devices that can manipulate 6 degrees of freedom (DOF). The simplest and most common input device is the mouse. Zhai [83] conducted a number of interesting studies on 3D input devices and introduced six performance measures for 6 DOF input devices:

- Speed
- Accuracy
- Easy of learning
- Fatigue
- Coordination
- Device persistence and acquisition

All input devices have the first four user performance measures in common. Coordination is unique to multiple degrees of freedom input control and can be measured based on the ratio between the length of the actual trajectory and the ideal trajectory in all spaces, along with translation space and rotation space. Device persistence and acquisition pertains to the ease of device acquisition. For example, what made the mouse so successful and well adapted was the fact that a mouse can be easily acquired (stays in place when not in use) compared to a pen that needs to be picked up in order to be used.

Moerman et al. [52] introduced a new locomotion technique named Drag'n Go that exhibits a good compromise between intuitiveness, easy to use and efficient navigation that allows the user to achieve his task in a short time. Drag'n Go is based on steering the users' viewpoint towards the target position (point of interest). Because this technique requires only a 2D input it can be used with a large variety of devices like mouse, touch or pen screen.

Marchal et al. [43] who is also co-author of the Drag'n Go details guidelines for developing multi-touch 3D navigation techniques introduced the concept of:

- Move around
- Look around
- Circle around
- Scrutinize

The user must be able to move around the virtual environment as he/she does in the real world. Common moves are translations in the z axis: forward or backward moves and rotations around y axis: turn left or right. Of lesser importance are sidestepping or strafing (translation along x axis) and altitude control (translation along y axis).

Another important task is adjusting the viewpoint orientation or look around. This can be achieved by rotation around y axis (look left and right) and around x axis (look up and down). Because people rarely tilt their heads, the rotation around z axis does not seem necessary.

If the user wants to focus on a particular object or an area, he/she must be able to look from different sides. This can be achieved by orbiting around a particular object in the horizontal plane.

When orbiting around an object is not enough, the user might want to look at the object more closely. In the virtual environment the user must be able to modify his field of view (FOV). This can be translated as using an optical tool such as a magnifying glass or bending over an object in real life.

The Move & Look viewpoint control technique emerged from these four tasks.

An extensive set of guidelines for 3D user interface design was developed by Stuerzlinger [67] to help 3D interface designers in creating robust and usable 3D user interfaces. Here are the ten guidelines:

- 2D input devices are advantageous
- Perspective and occlusion are the most appropriate depth cues
- Interact only with visible objects
- People see the object, not the cursor
- Floating objects are the exception
- Objects don't interpenetrate

- 2D and 2½D tasks are simpler than 3D
- Constrained navigation and rapid transportation is good
- Full 3D rotations aren't always necessary
- Reality simulation isn't always appropriate

The importance of easy navigation is outlined in *"Interact only with visible objects"* as users need to navigate to objects in order to interact with them. This preference to navigate before reaching an object was researched by Phillips et al. and Ware et al. [56, 76]. Taking into account the aforementioned criteria I designed my navigation technique to be very easy, allowing the user to position itself in the environment. My virtual environment includes features like navigation through environments with multiple scales, automatic speed control, and collision detection.

Based on the guideline *"People see the object, not the cursor"*, users do not only focus on the tip of the tool, which in our case would be the mouse cursor or the tip of their fingers, but also perceive their entire hand as the manipulation tool in the environment. This fact is the main reason behind a common issue of devices with smaller touch screens, such as tablets and smartphones. Because these devices have limited screen real estate, usually the user's hand occludes most of the display area in the middle of the screen when he/she performs a task. However, because users commonly position the POI in the center of the screen, the most interesting part of the scene tends to be in the middle of the screen, which might be occluded by their hand. To address this issue, I devised and adopted a system that enables the user to place their fingers or mouse cursor wherever they want on the screen to perform a desired task. Therefore, the user can focus on the object(s) and not on the cursor. Adopting the suggested method enables users to keep the POI in the centre of the screen when they navigate toward it by placing their fingers or mouse in another section of the display. As a result, the POI would not be occluded by the user's hand.

The idea that floating objects are the exception, which is stated in the fifth guideline, was highly influential in my research on automatic 3D navigation. In our daily lives, when we look around we see that all of the objects are in contact with each other. This is because the gravity that exists between each and every two physical objects. We must note that there are exceptions to this fact; for instance, when a helium filled balloon is floating in the air or an airplane is flying, they are not in contact with any other object. Considering that such cases are exceptions and not a general rule, it is not very reasonable to design an entire system that caters to the exception at the expense of the general rule. Therefore, I made the design decision to guide the user along the desired path by placing floating cubes that can be "picked up" like "keys" in game environments. The user is attracted to these floating objects thus guiding the user through the desired path.

In the real world two objects cannot not occupy the same space. Realistic virtual objects must obey the same physical laws. Thus the *"Objects don't interpenetrate"* guideline is an application of common sense. Yet, the implementation of this guideline in a virtual environment is a challenging task. In fact, in the real world the majority of the objects we touch or handle are solid and it is known that one of the properties of a physical object is that they cannot interpenetrate another solid physical object. With enough force or pressure a solid object can and will deform, but we cannot push ourselves hard enough to get halfway through a wall so we can see in two different rooms at once while keeping the wall intact. The automatic 3D navigation system that I designed and implemented for this research work prevents object interpenetration all the times. This means that the user should not be allowed to move through solid objects in the same way that a person cannot arbitrarily choose to walk through a wall in the real world. This rule applies not only to backward and forward navigational movements, but also to panning and orbiting movements as well.

Furthermore, due to the recognized importance of the eighth guideline for a navigation system, I reduced the DOF available to the user in my system by not allowing the user to rotate around the view direction. This design decision was made based on the fact that in real life people rarely tilt their head, and when they do they often cannot hold this position for extended periods of time due to the strain on the neck this action causes. Reducing the DOF available to the user reduces the complexity of the system and thus makes it easier to use. As discussed above for guideline seven, my suggested navigation system also supports rapid transportation by adopting an automatic multi-scale navigation system that adjusts its speed based on the environment. This enables users to traverse large spaces quickly without the need for instantaneous teleportation. As suggested by Bowman et al. [8], teleportation systems have the big drawback that they often leave the user disoriented as they attempt to ascertain a new position and orientation.

When developing the navigation system presented here, I applied the following main considerations:

- Ease of learning
- Ease of remembering
- Ease of use with maximum level of intuition as possible
- Rich features with minimum key input

2.2 Navigation

In virtual environments, a user action must be mapped in some more or less natural way to travel. Interactions with virtual environments can be decomposed into elementary tasks [10] such as navigating to change the viewpoint or selection and manipulation of virtual objects. A toolset of techniques based on principles of navigation derived from the real world is presented by Darken et al [23], and their weaknesses and relative strengths are compared. One of the navigation techniques presented was similar to two-dimension maps, but extended into the third dimension through the World in Miniature (WIM) interface. In WIM, objects are brought into reach through a miniature copy of the environment floating in front of the user, see Pausch et al. [55], Stoakley et al. [66], Mine [48,49] and related earlier work by Teller [69]. In the WIM technique, in order to plan a route, the user manipulates a virtual representation of himself. A small human figure represents the user's position and orientation in the miniature world. The user selects and manipulates this small human figure in the miniature environment in order to define a path for the viewpoint to move along, then the system executes the motion in the full scale environment. Pausch et al [55] found that this technique is most intuitive when the user literally moves into the miniature world, replacing the full-scale world and then creating another miniature world. One important advantage of this technique relative to other route planning techniques is that the user representation has orientation as well as position so that viewpoint rotations, not only translations, can be defined. WIMs have shown excellent promise in areas such as remote object manipulation and wayfinding. One drawback of WIMs was found to be the display real estate that needed to be shared between miniature copy and the original environment. In addition, Mine et al. [51] found that fine-grained manipulations can be difficult. Mine [50] offers an overview of motion specification interaction techniques. He and Robinett [60] also discuss issues relevant to their implementation in immersive virtual environments. Several user studies regarding immersive travel techniques have been described in the literature, for instance comparing different travel modes and metaphors for specific virtual environment applications, Chung [20], Mercurio et al. [47]. There are various types of travel tasks. Understanding this is important because the usability of a particular navigation technique often depends on the task for which it is used for. Bowman et al. [10] identified three main tasks which are exploration, search, and manoeuvring.

Exploration is performed when the user is browsing the environment, this is used at the beginning of an interaction with the environment (i.e., looking around), but it may become important later. Because the user wanders around in the world this technique should allow continuous and direct control of the viewpoint movement or the minimum the ability to stop the current movement. Not being able to deviate from the current path would depreciate the user’s discovery process. In some applications this must be balanced in order to provide an enjoyable experience in a given amount of time, Pausch et al. [53]. Some design decisions must be made in order to avoid the viewpoint to be “flipped over” (looking at the scene upside down) as users can most certainly become confused when the view transitions quickly from normal view to reversed view. The user must be able to focus cognitive resources on spatial knowledge acquisition and information gathering, so techniques should impose little cognitive load on the user.

Search task involve travel to a specific target location within the environment (i.e., driving or flying with steering). The user in a search task knows a priori the location to which he/she wants to navigate. There is a distinction between naïve search task, where the user does not know the position of the target or the path to follow to reach the target, and a primed search task, where the user has knowledge of the target position. A naïve search can ultimately be considered a simple exploration, assuming that this exploration is being done with a specific goal in mind. This may start out as a simple exploration, but clues or wayfinding aids may direct the search so it becomes more focused than exploration. Several 3D interfaces require search via travel. For example, the user in an architectural environment may wish to navigate to a window to check sight lines. The techniques for search tasks are generally more goal oriented than the techniques for exploration. As an example, the user can specify the target location directly on a map instead of incremental movements. Nonetheless such techniques do not apply to all situations. Map-based techniques were inefficient if the target location is not present on the map as found by Bowman et al. [3].

Manoeuvring is utilized when the user needs to observe a specific object in detail, this involves small and precise movements (i.e., panning parallel to a view plane or orbiting around one or more objects). For example, the user may wish to check the positioning of an object it has been manipulating in a 3D modeling system and needs to view it from different angles. Compared to large scale movements through the environment this task seems trivial, but it is exactly these small scale movements that can cost the user a lot of time also causing frustration if the interface does not support it. Some applications may require special travel techniques only for maneuvering. Travel techniques for this task should allow high precision of motion but not in the detriment of speed. One of the best solutions for maneuvering tasks can be the physical motion of the user's head or body because this is efficient, precise, and natural. If precise work is important in an application and head or body tracking is not available, then other techniques for maneuvering, such as object focused travel techniques must be considered.

The above tasks are classified by the user's goal for the travel task. There are other characteristics of the task that should be considered when considering travel techniques:

- Distance to be traveled
- Amount of turning required in the travel task
- Target visibility from the starting point
- Number of DOF required for the movement

- Accuracy required for the movement
- Other primary tasks that take place while travel

The navigation task has been researched vastly and an attempt to classify and categorize interaction techniques into structures has been made. For the task of navigation a minimum of four different classifications have been proposed:

- Active versus Passive Techniques
- Physical versus Virtual Techniques
- Task Decomposition
- Interaction Metaphor

Differentiating between active navigation techniques, in which the user directly controls the movement of the viewpoint, and passive navigation techniques, in which the viewpoint's movement is controlled by the system is one way to classify navigation techniques.

In the physical navigation technique, the user's body physically translates or rotates (using head tracking) in order to translate or rotate the viewpoint. In virtual navigation the user's body stays stationary while the virtual environments viewpoint moves.

The navigation task was decomposed by Bowman et al. [7] into three subtasks:

- Direction or target selection
- Speed/acceleration selection
- Conditions of input

direction or target selection, this refers to the primary subtask in which the user specifies how/where to move, speed / acceleration selection describes how the user controls his speed, and conditions of input, refers to how navigation is started, continued and stopped. Each subtask can be achieved using multiple techniques.

Direction or target selection can be performed using gaze directed steering, or pointing / gesture steering, or discrete selection via menus, or 2D pointing.

Speed / acceleration can be done by constant speed /acceleration, or gesture based, or explicit selection, or user scaling, or automatic.

Input conditions can be implemented using constant travel (no input), or continuous input, or start and stop, or automatic start and stop input technique component.

This requires the user to be able to control the speed and the direction. When walking on a surface the user can be bound to the plane thus leaving the user with control only in one DOF. Flying around requires control of two DOF and in some systems speed control requires an additional DOF. A fixed constant speed leads to a variety of problems because constant speed will always be too slow in some situations and too fast in others. If the perceived speed is too slow, frustration of the user quickly sets in. If the speed is too fast the user easily overshoots the target, forcing the user to turn around adjust the viewpoint and navigate back.

Allowing the user to control the speed adds complexity to the interface. A discrete technique for speed control would be the use of two keys, one to increase the speed by a specified amount and the other to decrease the speed. The problem with this approach is that the user easily overshoots or undershoots the target [72] and might be forced to take corrective actions [67]. This causes the users to spend additional time for adjusting the speed and viewpoint. Another issue with manual speed control is that users might easily fly into objects using this technique. This issue has been observed mostly when users are moving backwards mainly because they cannot see what is behind them (such as a rear-view mirror/camera in the car). Usually, users cannot estimate the distance to the objects correctly; therefore, they might adjust the speed inappropriately which can lead to unwanted landings inside the objects. This can be very frustrating for users and might cause usability issues. A potential technique for solving this problem is to slow down the user when close to the target object. Trapp et al. [71] present strategies that aim to visualise 3D points-of-interest and guides the user towards it.

Classification by metaphor is more of an informal classification and it is easy to understand, primarily from the user's point of view. As an example if a navigation technique is described as "flying carpet" metaphor [54] the user can assume that it allows movement in all three dimensions and to steer using hand motions.

Physical locomotion techniques is an imitation of natural method of locomotion in the physical world and it is intended for immersive virtual environments. This technique uses the user's physical effort to navigate through the virtual world. Walking is the most trivial technique for navigating in a 3D virtual environment, it is natural and provides the user with a good sense of equilibrium and spatial understanding. Real walking is not always suitable because of technological or space limitations. Also a real issue arises with cabling as if not carefully handled the user can easily become entangled while walking in the virtual world. Current wireless devices can mitigate these concerns. For example the Vizuality system (Figure 2-1) uses a wireless headset and a high tech motion tracking device giving the users the ability to walk and run around the virtual environment, greatly reducing the motion sickness feeling associated with users that are seated while navigating in the virtual environment [18]. Research at the University of North Carolina produced the HiBall tracking system [79,78], an optical tracking system that allows tracking a wide area by employing a scalable tracking grid on the ceiling similar with Vizuality (see Figure 2-1). Two main approaches are prevailing: *outside–in* and *inside–out*. Outside-in approach uses fixed well know locations in the environment of the optical or ultrasonic sensors and sense locations (markers) on the user [27]. The inside-out approach the markers are positioned in the environment and the sensors on the user [78]. Mobile augmented reality [29] uses real walking in very large areas where users have additional graphical information superimposed on their view of the real world. A modern version of Höllerer's system would be Google Glass.

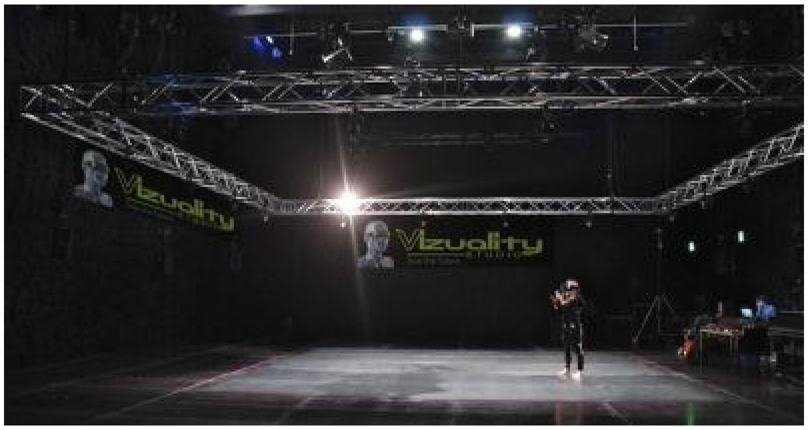


Figure 2-1. Vizuality system offers a completely immersive virtual experience that combines VR with motion tracking technology to enable users to walk and run around their virtual environments.

Walking in place can substitute real walking. Here the user simulates walking by moving their feet up and down without actually translating. This technique does not require a large physical environment to explore a VE, while still supporting a sense of presence for users. Caveats are that the motion cues provided by walking in place are different from real walking resulting in diminished sense of presence and that, even if the environment is theoretically unlimited, a user cannot walk unlimited distances.

To enable walking in place researchers have devised multiple technologies. Using position trackers on the user's feet and a neural network to analyze the up and down motions of the feet, Slater, Usoh and Steed [64] have built a system that can distinguish walking in place from other types of foot motion. On average, the neural network was able to detect the walking motion correctly 91% of the time. Templeman's [70] Gaiter system uses multiple sensors and a more sophisticated algorithm to recognize a natural walking motion (see Figure 2-2).

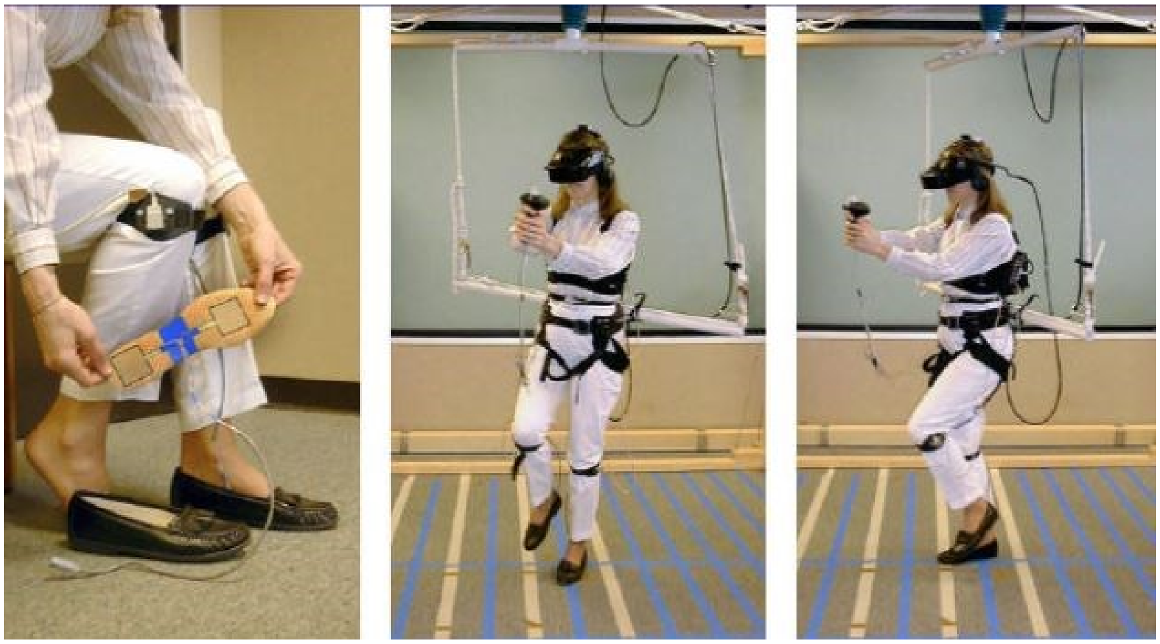


Figure 2-2. Templeman's GAITER system [70]

Iwata and Fujii [33] have developed special sandals that permit the user to shuffle in place to move forward on a low friction surface instead of the up and down motion of other walking methods.

Compared to virtual travel, walking in place maintains an increased level of presence in the virtual environment, but is still outperformed by real walking [73]. There are several issues with this kind of interaction technology, such as recognition errors and user fatigue. Still, in general these systems perform well when the user must navigate further than their physical reach and when a high level of realism is required.

Another form of navigation techniques are devices that simulate walking. These are special locomotion devices that “provide a real walking motion and feel while not actually translating the user’s body” [10] similar to walking on a stepper [45] or a treadmill. One important limitation of such devices is that the users cannot turn, requiring an additional device to accomplish this, i.e., a joystick [16].

Other, more advanced techniques used a tracker to track the user's head and feet allowing the user to slowly turn their head to change the direction which would then cause the treadmill, mounted on a large motion platform, to rotate as well [53]. However, limitations of the hardware will not allow the user to turn quickly or to sidestep. Another innovative design is the Omni-Directional Treadmill (ODT) [22] and the Torus treadmill [32]. These build on the idea of two sets of rollers moving orthogonally to each other, giving the treadmill the ability to move in any arbitrary horizontal direction. The above-mentioned devices work well, but still do not support sudden turns or sidestepping.

The Gait Master system [31] detected the user's motion through force sensors and moved several small platforms around so that the user always felt a hard "ground" surface at the correct location of each step. However, this device is very complex and has potentially serious safety issues to resolve [10].

In cases when waking is not all-important but some level of physical activity is preferred, a common exercise bicycle can be used as an interaction device [14]. The speed can be naturally controlled through the speed of pedaling on these devices. More advanced versions even give force-feedback on the handlebar during turning or even leaning of the whole bike, favoring a natural turn.

The Uniport consists of a unicycle type mobility platform, which allows a person to 'pedal' his or her way through the virtual environment, as seen through a head mounted display (HMD) (see Figure 2-3).

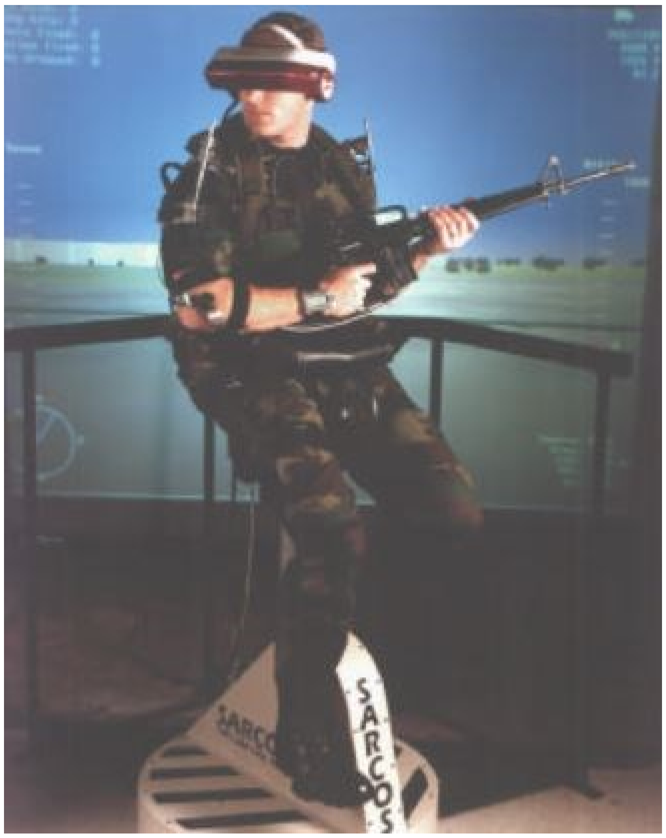


Figure 2-3. Sarcos Uniport locomotion device [80]

Steering techniques are an important approach to navigation in 3D virtual environments and support the “continuous control of direction of motion by the user” [10]. Through the provided interface the user specifies an absolute or relative direction of movement. While such interfaces are generally easy to understand and provide a high level of control [10], steering requires practice, can be slow for long distances and can cause disorientation [61].

Gaze/head directed steering is the most widely used travel technique in many 3D toolkits [35]. A real gaze directed steering method would use an eye tracker, but most implementations use a head tracker and determine gaze through a ray that goes from the orientation of the head tracker towards the virtual camera position. This technique lets the user to move in the direction towards which he/she is looking.

People comprehend gaze/head directed steering very easily and it is generally considered a fairly natural and intuitive travel technique, at least when the navigation is restricted to a 2D horizontal plane in an immersive VE. However, in 3D navigation gaze directed steering encounters two issues: one is that when the user wants to travel in a horizontal plane he/she may be slightly off as it is hard to determine if the head is exactly upright. The second issue is that it is not natural to navigate up or down by looking straight up or down.

The major disadvantage of this technique is the fact that the gaze/head direction is coupled with the navigation direction, meaning that the user cannot navigate in one direction while looking at another.

Hand directed steering or pointing resolves the issue of coupling gaze direction and navigation direction. The pointing technique [50] for travel gets its name from an immersive VR implementation where the user holds a tracker in his hand. The forward vector of the hand tracker is first transformed into a world coordinate vector, which is then normalized and scaled by speed. The user is then translated with the resulting vector.

Mine [51] extended this concept by using two hands to specify the vector. Here, the vector defined by the two hand positions is used instead of the hand orientation for the travel direction. The issue with this technique is that one of the hands must be designated to define the “forward” direction. Bowman [11] used Pinch Gloves to implement this technique and choose the hand that initiated the navigation gesture to represent the “forward” direction. This technique can be used to easily define any 3D vector and also supports speed control linked to the distance between the two hands.

Because the user controls now two values (direction and speed), this pointing technique is more flexible, but requires also a higher level of cognitive load which can lead to reduced performance in complex tasks, i.e., information gathering [7]. The pointing technique gives the user the capability to look in any direction while navigating to a preferred target [9].

Torso directed steering uses the user’s torso to define the direction of travel. It exploits the fact that typically people turn their bodies towards the direction of movement. To realize this, a tracker is normally attached to the user’s waist (i.e. belt). The implementation of this technique is then similar to the gaze directed steering, but uses the waist tracker instead of a head tracker.

Like the pointing technique this technique has the advantage that it decouples the user’s gaze direction from the direction of travel, as well as leaving the user’s hands free to perform other activities. One essential disadvantage is that this technique can be applied only to immersive virtual environments that permits motion in the horizontal plane, as it is not easy to point the torso up or down.

Lean directed steering is a somewhat more elaborate technique that allows the user to specify the travel direction by leaning. A technique developed by von Kapri et al. [74] uses the metaphor of "leaning in to view objects" to specify the travel direction through the direction that the user leans towards (see Figure 2-4).

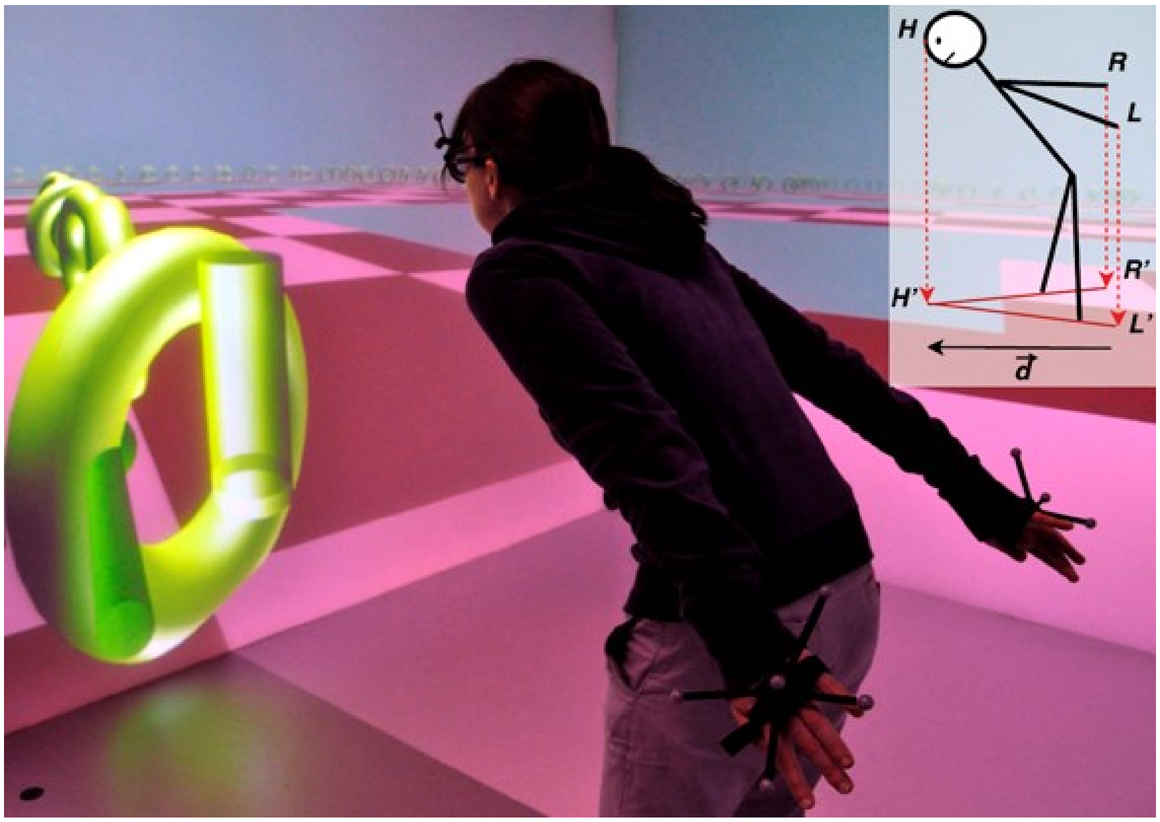


Figure 2-4 PenguFly is a bi-manual lean directed navigation technique for virtual environments that tracks the head and hands. Navigation direction is computed from right hand to head vector and left hand to head vector, while speed is proportional with size of the direction vector. [74]

The PenguFly technique tracks the user's hands and head, computing the navigation direction from the average of the two vectors defined by the head and right hand as well as the head and left hand respectively. Navigation speed is proportional to the average length of these vectors. The researchers concluded [74] that lean directed steering is more accurate than pointing thanks to the higher discrete steps in speed control. One surprise that emerged from the research was the high nausea caused by the leaning motions.

Beckhaus et al. [4] presented the ChairIO interface to implement a lean directed steering technique using an ergonomic stool that can shift, tilt, rotate and bounce. Using magnetic trackers all these movements are captured and transformed into a virtual environment navigation interface used for steering (see Figure 2-5). Furthermore, Marchal et al. [44] invented a novel human scale joystick interface named "joyman". This involved a human standing on a rigid surface placed on a trampoline, surrounded by a safeguard rail. All movement data is collected through an inertial sensor. When the user leans in any direction, the rigid surface also leans toward one of the sides of the trampoline's framework. Through the inertial sensor, the orientation of the lean is transformed into a direction and speed for steering (see Figure 2-6).

All lean-directed steering techniques support natural proprioceptive and kinesthetic senses to the user to depend on, permitting an excellent navigation understanding within the virtual environment. Another advantage of the lean-directed steering technique is that it combines the navigation direction and speed into a single movement. On the other hand, the major disadvantage is that it is limited to navigation only on a horizontal plane, i.e., 2D.

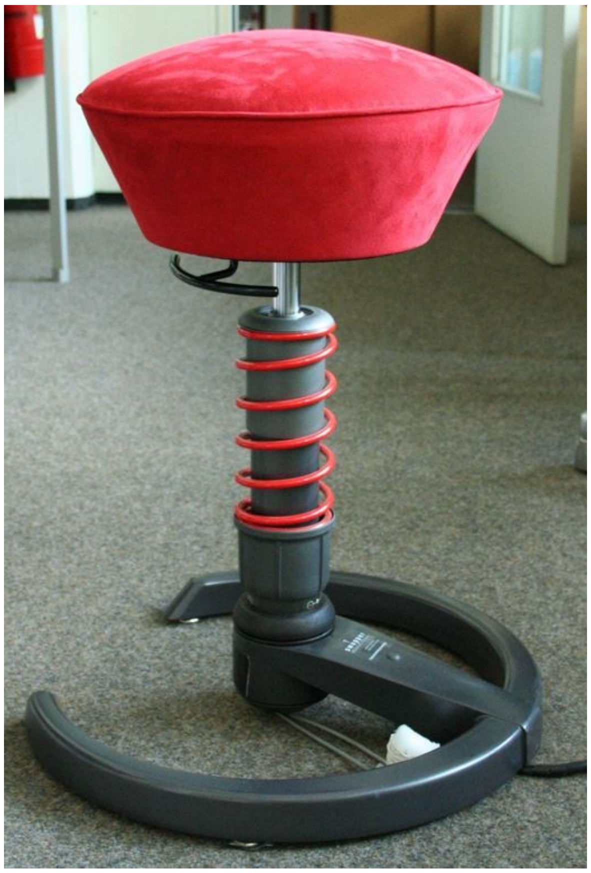


Figure 2-5 The ChairIO is a chair based interface to control navigation in 3D environments or to control cursor movement on the desktop [4]

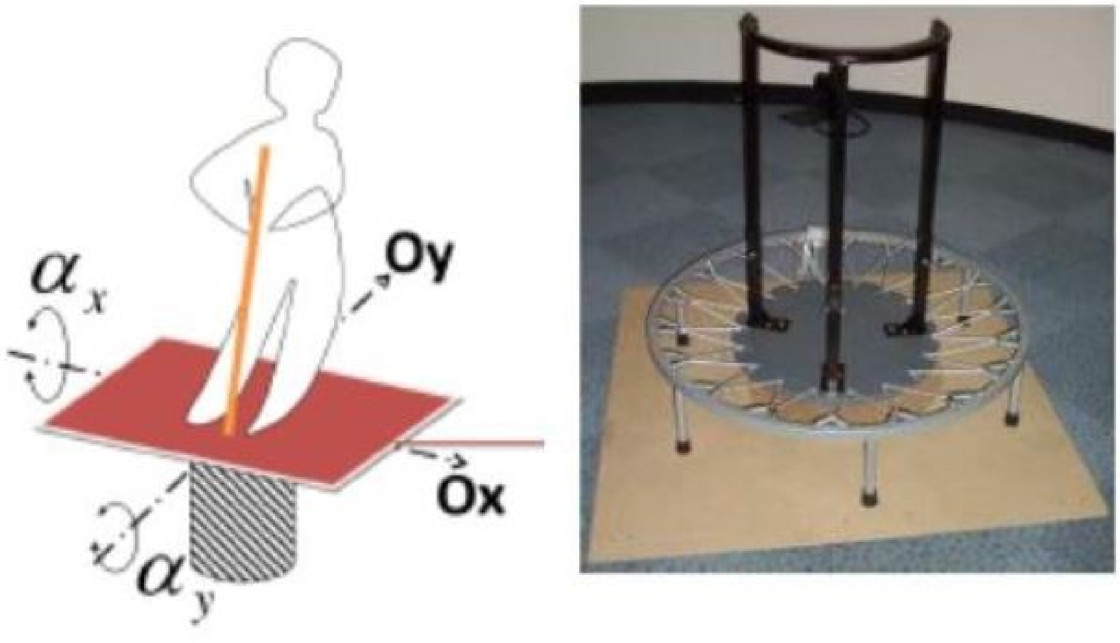


Figure 2-6. The novel Joyman peripheral device [44]

Physical steering props are specially designed devices for steering, which are then used in virtual environments for controlling the travel direction. A very familiar device is the steering wheel, similar to the one found in a car. It can even be supplemented with an accelerator and brake pedals for virtual driving. These devices can be used in immersive or desktop virtual environments and are understandable by any user who has driven a car.

To simulate real or imaginary vehicles other specialized steering props can be used. For instance, to pilot a virtual merchant vessel [1] a bridge simulator uses realistic ship controls (see Figure 2-7). Another example is the ERGONAUT, [24], a tractor cockpit used to control a virtual tractor.

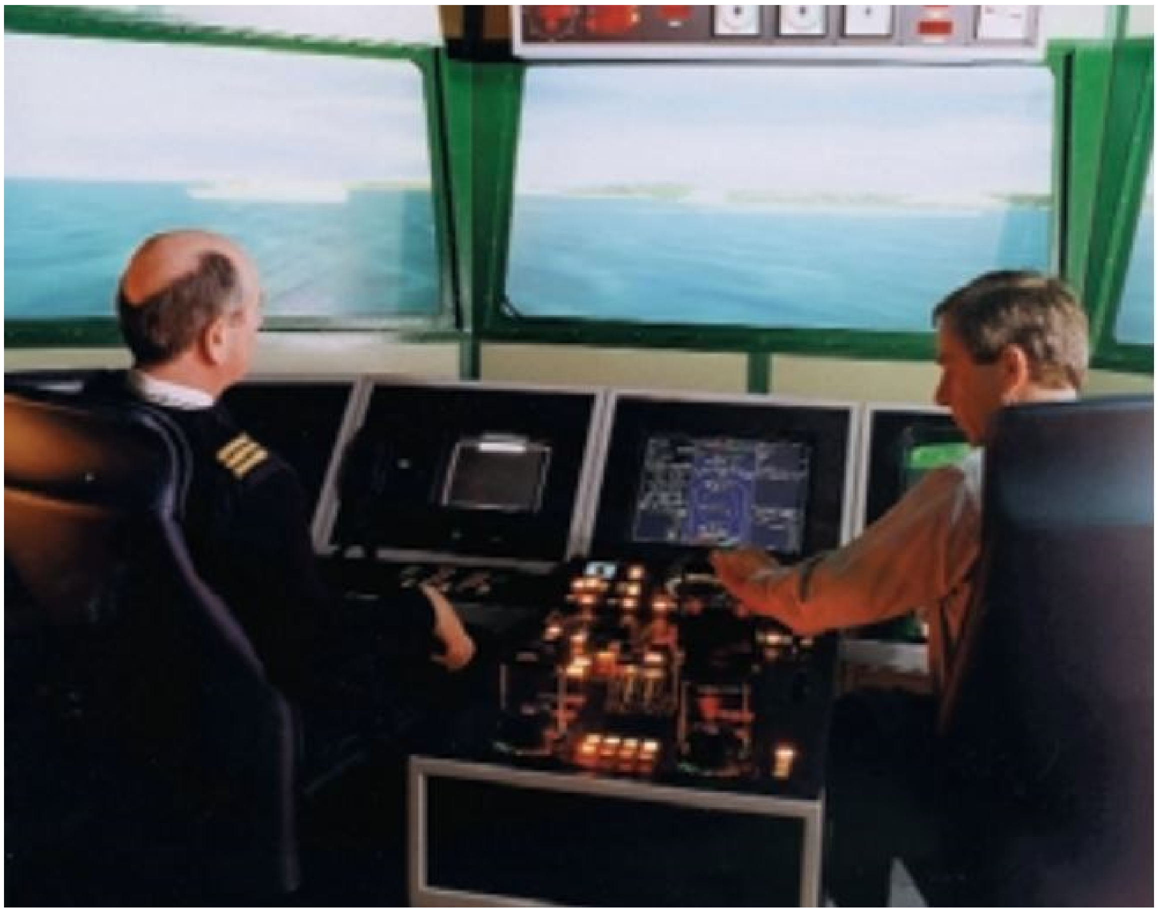


Figure 2-7 Ship bridge simulator at Warsash Maritime Centre [15]

Flight simulators used this “near-field haptics” approach [15] to train pilots for years without the risk of crashing. Using high-fidelity display adapters and quick to respond hydraulic systems, the simulators are able to provide pilots with an experience close to actually flying an aircraft [25]. The classic Pirates of the Caribbean attraction at Disneyland uses a steering wheel and throttle for the virtual ship [162]. Another attraction at Disney Quest, the Virtual Jungle Cruise simulates the effect of rafting on white-water rapids using physical oars to steer and control the speed of the virtual raft [139]. In driving and racing games steering wheels and motorcycle handle bars are used. Moving interfaces are used to simulate skateboarding, snowboarding and skiing.

Physical steering props are desired when steering is a significant component of the whole user experience. A possible drawback is that props may produce unrealistic expectations of realistic control and feedback in users familiarized to operating the same steering interface in a real vehicle.

A distinct physical steering device developed at HIT lab is the Virtual Motion Controller, or VMC [80]. This interface is based on a subset of the real world walking motion called "sufficient-motion" and consists of four weight sensors encapsulated underneath the working surface. The concave shape of the working surface provides essential feedback to the user about his physical location. The center of this platform is flat and corresponds to a standstill in the virtual world, when the user steps away from the center, him/her starts traveling in the direction of the step. The speed is dependent of the distance of the user from the center.

The VMC is very intuitive combining travel direction and speed in one movement, also "allows the user to rely on natural proprioceptive and kinesthetic senses to maintain spatial orientation and understanding of movement within the environment" [10]. One drawback of the VMC is the 2D motion limitation, which may be overcome by adding a vertical motion interface.

Semi-automated steering techniques are used when the UI designer wants the user to have the feeling of control while at the same time guides the user toward an end destination and maintaining user's attention to the essential features of the environment. A good example is the Virtual Jungle Cruise attraction at Disney Quest, which simulates a raft traveling down a river [39] or the Disney's Aladdin attraction [54] where users fly a magic carpet. Both attractions offer the user the feel of control when actually a limited control over the speed and steering is given.

The idea of semiautomatic steering is that the user steers within constrains provided by the system. This is applicable to both immersive and desktop 3D UIs. The metaphor of a boat/raft traveling down a river was used by Galyean [28] and at Disney Quest the Virtual Jungle Cruise attraction [6239]. The boat/raft moves continuously even if the user is not actively steering or speeding, allowing all users to reach the final destination.

Route planning techniques allow the user to specify a path through the virtual environment before the actual movement takes place. The actual navigation takes place after the user defined, reviewed or edited the path.

Drawing a path is one of the route planning techniques. An example of this technique was demonstrated in a desktop 3D virtual environment using a mouse by drawing the intended path directly on the 3D scene [30] and the user avatar automatically moves along the path. The height of the avatar is fixed and the user's point of view is decupled from the movement view allowing the user to look around and explore the virtual environment.

A second technique for specifying a path is to spread marker points along the path. After placing the markers either directly in the scene or on a 2D or 3D map the system will create a path that traverse all the marker locations. The user can increase the granularity of the path by placing more markers or leave it to the system to pick a path by placing fewer markers. One important feature of this technique is the feedback to the user, a good design will encompass interactive feedback to show the user the path through the virtual environment or on the map.

In some applications where both navigation and object manipulations are required it can be more appropriate to use the manipulation based navigation techniques. Hand based manipulation metaphors are used by these techniques that manipulate the viewpoint or the whole world, such as Hand-Centered Object Manipulation Extending Ray-Casting (HOMER) or arm-extension (Go-Go).

One of the viewpoint manipulation technique called “camera-in-hand” technique [177] uses position trackers to navigate in a desktop virtual environment. The bat was used as a tracker device and the absolute coordinates of the bat specified the coordinate of the virtual camera from which the 3D scene is viewed. This technique is best for navigating in desktop 3D UIs because the input device is actually 3D, and the user gets a feeling for the spatial relationship between objects in the 3D virtual environment using his/her proprioceptive sense. This technique can get confusing at times because the user has a third person view of the whole environment but the 3D scene viewed/displayed is from a first person point of view.

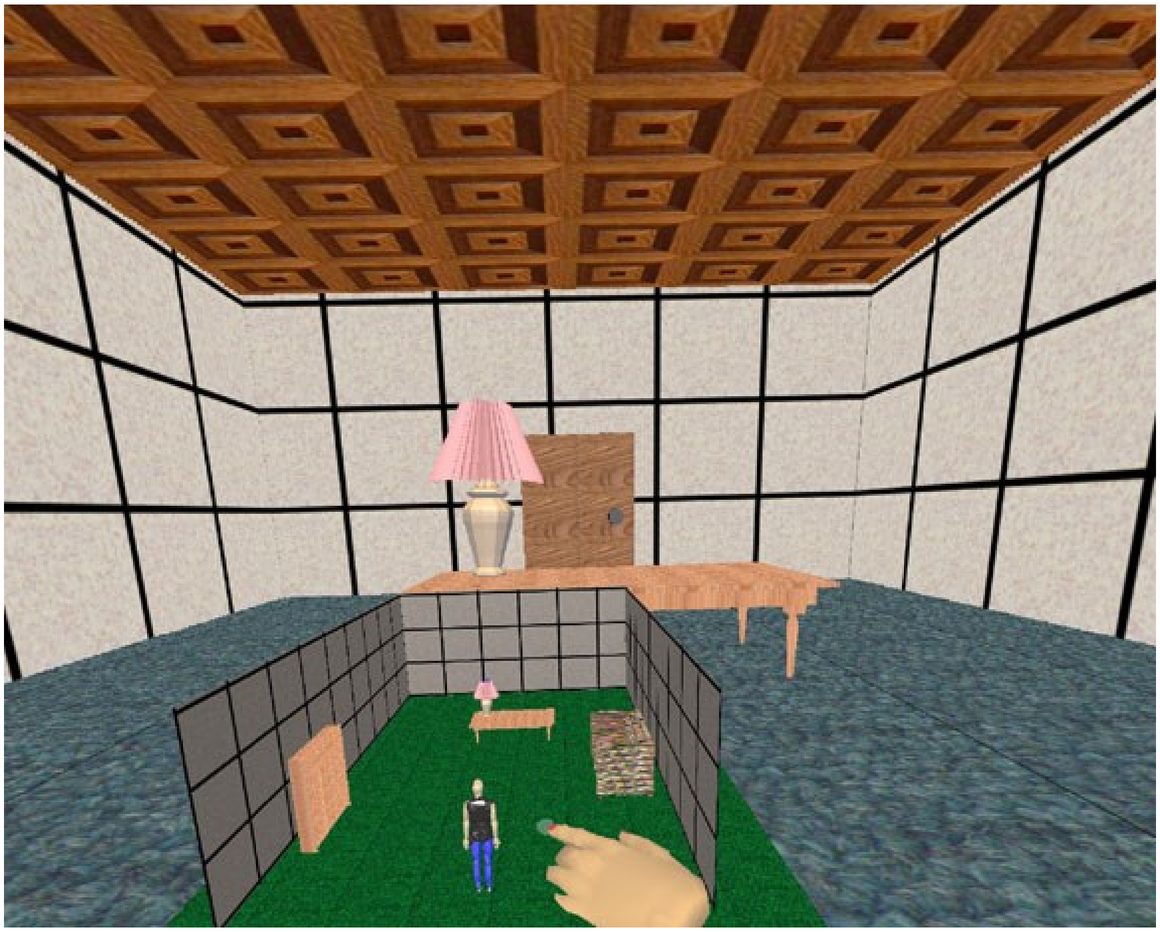


Figure 2-8. An example of a WIM.

In place of a camera the user can manipulate a virtual representation of himself/ herself (avatar) in order to navigate similar to the map based technique, but including the third dimension. The WIM technique uses a small version of the world, including a small version of a human that represents the user's position and orientation in the miniature world to allow the user to do indirect manipulations of the avatar in the virtual environment (see Figure 2-8). This technique is better understood when the user's view actually zooms into the miniature world, replacing the full-scale world and then creating a new WIM [55]. Each

WIM acts as a portal into a different part of the virtual environment allowing for a quick navigation in the virtual environment. This technique allows the definition of user's viewpoint rotations not just translations.

Fixed object manipulation is a technique that allows navigation by "letting a selected object serve as focus for viewpoint movement" [110]. By selecting an object in the virtual environment the user's viewpoint moves relative to the object as the user manipulates the object which remains stationary.

A good example of fixed object manipulation technique was designed by Pierce et al. [57] called image-plane technique. Manipulating objects in the environment was done by hand movements after selecting the object. Closer examination of the objects was done by retracting the hand towards the body however when in navigation mode the same gesture would cause the user to move toward the selected object. Moving the viewpoint around the selected object was accomplished by hand rotation. The scaled-world grab technique [51] and the LaserGrab technique [82] have the same concept.

The technique described above presents a smooth interaction experience in mixed navigation/manipulation task designs. There is however a need of the user awareness of the interaction active mode (navigation or manipulation).

An alternative to manipulating the viewpoint to navigate is to manipulate the entire world relative to the current viewpoint. One method for using manipulation techniques for navigation tasks is to allow the user to manipulate the world about a single point. An example of this is the "grab the air" or "scene in hand" technique [42, 77]. In this concept, the entire world is viewed as an object to be manipulated. When the user makes a grabbing gesture at any point in the world and then moves his/her hand, the entire world moves while the viewpoint remains stationary. Of course, to the user this appears exactly the same as if the viewpoint had moved and the world had remained stationary. In its simplest form, this technique requires a lot of arm motion on the part of the user. Enhancements to the basic technique can reduce this. First, the virtual hand can be allowed to cover much more distance using an arm-extension technique such as Go-Go [58]. Second, the technique can be implemented using two hands instead of one, as discussed in the next section.

Manipulating the world has also been implemented by defining two manipulation points instead of one. The commercial product SmartScene, which evolved from a graduate research project [42], allowed the user to navigate by using an action similar to pulling oneself along a rope. The interface was simple - the user continuously pulled the world toward him/her by making a simple grab gesture with his/her hand outreached and bringing the hand closer before grabbing the world again with his/her other hand. This approach distributed the strain of manipulating the world between both of the user's arms instead of primarily exerting one.

Another advantage of dual-point manipulation is the ability to also manipulate the view rotation while navigating. When the user has the world grabbed with both hands, the position of the user's non-dominant hand can serve as a pivot point while the dominant hand defines a vector between them. Rotational changes in this vector can be applied to the world's transformation to provide view rotations in addition to navigating using dual-point manipulations.

Another major category of navigation metaphors depends on the user selecting either a target to navigate to or a path to navigate along. These selection-based navigation metaphors often simplify navigation by not requiring the user to continuously think about the details of navigation. Instead, the user specifies the desired parameters of navigation first and then allows the navigation technique to take care of the actual movement. While these techniques are not the most natural, they tend to be extremely easy to understand and use.

In some cases, the user's only goal for a navigation task is to move the viewpoint to a specific position in the environment. The user in these situations is likely willing to give up control of the actual motion to the system and simply specify the endpoint. Target-based navigation techniques meet these requirements. Even though the user is concerned only with the target of navigation, however, this should not be construed to mean that the system should move the user directly to the target via "teleportation." An empirical study [5] found that teleporting instantly from one location to another in a VE significantly decreases the user's spatial orientation (users find it difficult to get their bearings when instantly transported to the target location). Therefore, continuous movement from the starting point to the endpoint is always recommended.

A 2D map or 3D WIM can be used to specify a target location or object within the environment to navigate to. A typical map-based implementation of this technique [11] uses a pointer of some sort (a tracker in an immersive 3D UI, a mouse on the desktop) to specify a target, and simply creates a linear path from the current location to the target, then moves the user along this path with a constant speed. The height of the viewpoint along this path is defined to be a fixed height above the ground.

Dual-target navigation techniques allow the user to easily navigate between two target locations. Normally, the user directly specifies the first target location by using a selection technique while the second target location is implicitly defined by the system at the time of that selection. For example, the ZoomBack technique [82] uses a typical ray-casting metaphor to select an object in the environment, and then moves the user to a position directly in front of this object. Ray-casting has been used in other 3D interfaces for target-based navigation as well [6]. The novel feature of the ZoomBack technique, however, is that it retains information about the previous position of the user and allows users to return to that position after inspecting the target object.

As noted above, the most natural and intuitive method for navigating in a 3D virtual world is real walking, but real walking is limited by the tracking range or physical space. One way to alleviate this problem is to allow the user to change the scale of the world so that a physical step of one meter can represent one nanometer, one kilometer, or any other distance. This allows the available tracking range and physical space to represent a space of any size.

There are several challenges when designing a technique for scaling the world and navigating. One is that the users need to understand the scale of the world so that they can determine how far to move and can understand the visual feedback they get when they move. Use of a virtual body (hands, feet, legs, etc.) with fixed scale is one way to help the user understand the relative scale of the environment. Another issue is that continual scaling and rescaling may precipitate the onset of cyber sickness or discomfort [6]. In addition, scaling the world down so that a movement in the physical space corresponds to a much larger movement in the virtual space will make the user's movements much less precise.

The most common approach to scaling and navigating is to allow the user to actively control the scale of the world. Several research projects and applications have used this concept. One of the earliest was the 3DM immersive modeler [17], which allowed the user to "grow" and "shrink" to change the relative scale of the world. The SmartScene application [42] also allowed the user to control the scale of the environment in order to allow rapid navigation and manipulation of objects of various sizes. The scaled-world grab technique [51] scales the user in an imperceptible way when an object is selected (Figure 2-9).

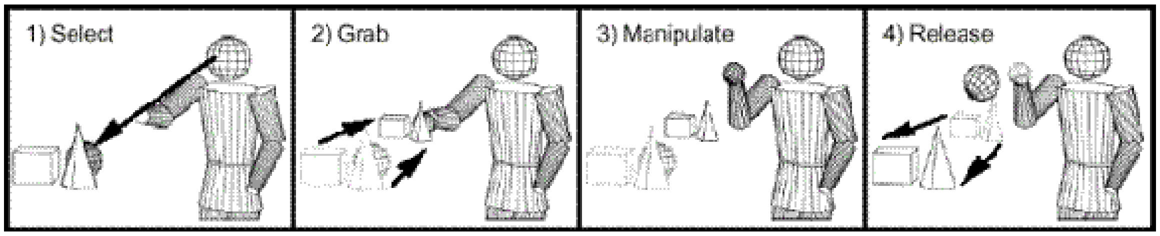


Figure 2-9. Scaling in the scaled-world grab technique. [51]

While active scaling allows the user to specify the scale of the world, it requires additional interface components or interactions to do so. Alternatively, 3D UIs can be designed to have the system change the scale of the world based on the user's current task or position. This automated approach affords scaling and navigating without the user needing to specify the scale. An example of automated scaling are the multi-scale virtual environments (MSVEs) developed by [37]. Each MSVE contains a hierarchy of objects, with smaller objects nested within larger objects. As the user navigates from a larger object to a smaller object, the system detects that the user is within the smaller object's volume and scales the world up. For example, a medical student learning human anatomy could navigate from outside the body and into an organ (Figure 2-10). During this navigation, the system detects the move and scales the world up to make the organ the same size as the medical student. Alternatively, when the student travels from the organ to outside the body, the system scales the world back down.

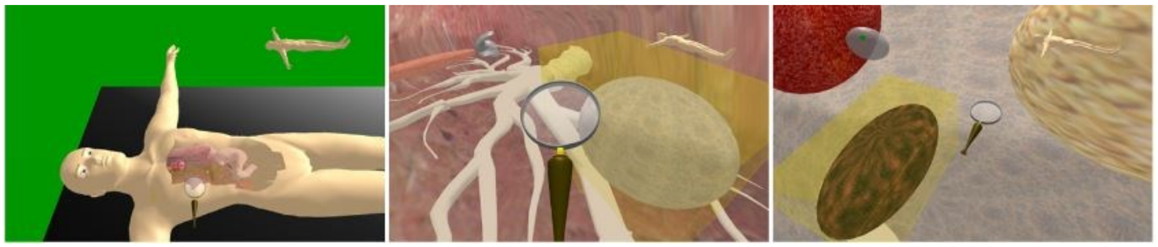


Figure 2-10. MSVE example: body scale, lung scale and a third level of scale. [34]

MSVEs allow the user to concentrate on navigating instead of scaling while still gaining the benefits of having the world scaled up or down. However, such VEs require careful design, as the hierarchy of objects and scales need to be intuitive and usable for the user.

3D UI designers should be concerned with how to orient the viewpoint, how to specify the speed of navigation, how to provide vertical navigation, whether to use automated or semi-automated navigation, whether to scale the world while navigating, how to transition between different navigation techniques, using multiple cameras and perspectives, and considerations of using non-physical inputs, such as brain signals.

For immersive VR, there is usually no need to define an explicit viewpoint orientation technique, because the viewpoint orientation is taken by default from the user's head tracker. This is the most natural and direct way to specify viewpoint orientation, and it has been shown that physical turning leads to higher levels of spatial orientation than virtual turning [3, 18].

A slight twist on the use of head tracking for viewpoint orientation is orbital viewing [36]. This technique is used to view a single virtual object from all sides. In order to view the bottom of the object, the user looks up; in order to view the left side, the user looks right; and so on.

There are certain situations in immersive VR when some other viewpoint orientation technique is needed. The most common example comes from projected displays in which the display surfaces do not completely surround the user, as in a three-walled surround-screen display. Here, in order to see what is directly behind, the user must be able to rotate the viewpoint (in surround-screen displays, this is usually done using a joystick on the “wand” input device). The redirected walking technique [59] slowly rotates the environment so that the user can turn naturally but avoid facing the missing back wall. Research on non-isomorphic rotation techniques [38] allows the user in such a display to view the entire surrounding environment based on amplified head rotations.

For desktop 3D UIs, setting viewpoint orientation is usually a much more explicit task. The most common techniques are the Virtual Sphere [19] and a related technique called the ARCBALL [63]. Both of these techniques were originally intended to be used for rotating individual virtual objects from an exocentric point of view. For egocentric points of view, the same concept can be used from the inside out. That is, the viewpoint is considered to be the center of an imaginary sphere, and mouse clicks/drags rotate that sphere around the viewpoint.

In the next part considerations are given for speed changing techniques of navigation. Many 3D UIs ignore this aspect of navigation and simply set what seems to be a reasonable constant speed. However, this can lead to a variety of problems, because a constant speed will always be too slow in some situations and too fast in others. When the user wishes to navigate from one side of the environment to another, frustration quickly sets in if he/she perceives the speed to be too slow. On the other hand, if the user desires to move only slightly to one side, the same constant speed will probably be too fast to allow precise movement. Therefore, considering how the user or system might control speed is an important part of designing a navigation technique.

2.3 Speed Control

The speed control of a navigation technique is linked with the scale of the environment and the user's preferences. The maximum allowed speed is dictated by the scale of the environment. Users can adjust the speed through the navigation interface by using a number of input commands [27] and speed mappings [1]. If the scale and level of detail of the environment is known a priori, then the maximum and minimum speed can be adjusted accordingly using the interface.

Virtual environments have become very complex and incorporate multiscale features. This can cause many problems for users and introduce a variety of usability issues. Mackinlay [41] first observed that the current distance to a target point is an appropriate way to control viewer speed. This was an important observation and was investigated thoroughly by Ware and Fleet [75]. They studied how well the minimum, average, directionally weighted average, and maximum distances to any visible point in the camera image worked to control the speed. They concluded that in most situations, the minimum generally worked best, but noted that averages were also competitive. An improved version of Ware and Fleet [75] interface is the cubemap-based navigation approach proposed by McCrae et al. [46].

McCrae et al. [46] used a six side cubical distance approach called cubemap that considers the entire surroundings of the user by computing a depth cubemap from the camera viewpoint. His approach consists of rendering six images in six deferent directions from the camera viewpoint, each one corresponding to a side of the cube. The field of view used by the cubemap is 90º in camera perspective (see Figure 2-11), permitting the blending of the resulting frustums to cover the entire environment located between the clipping planes.

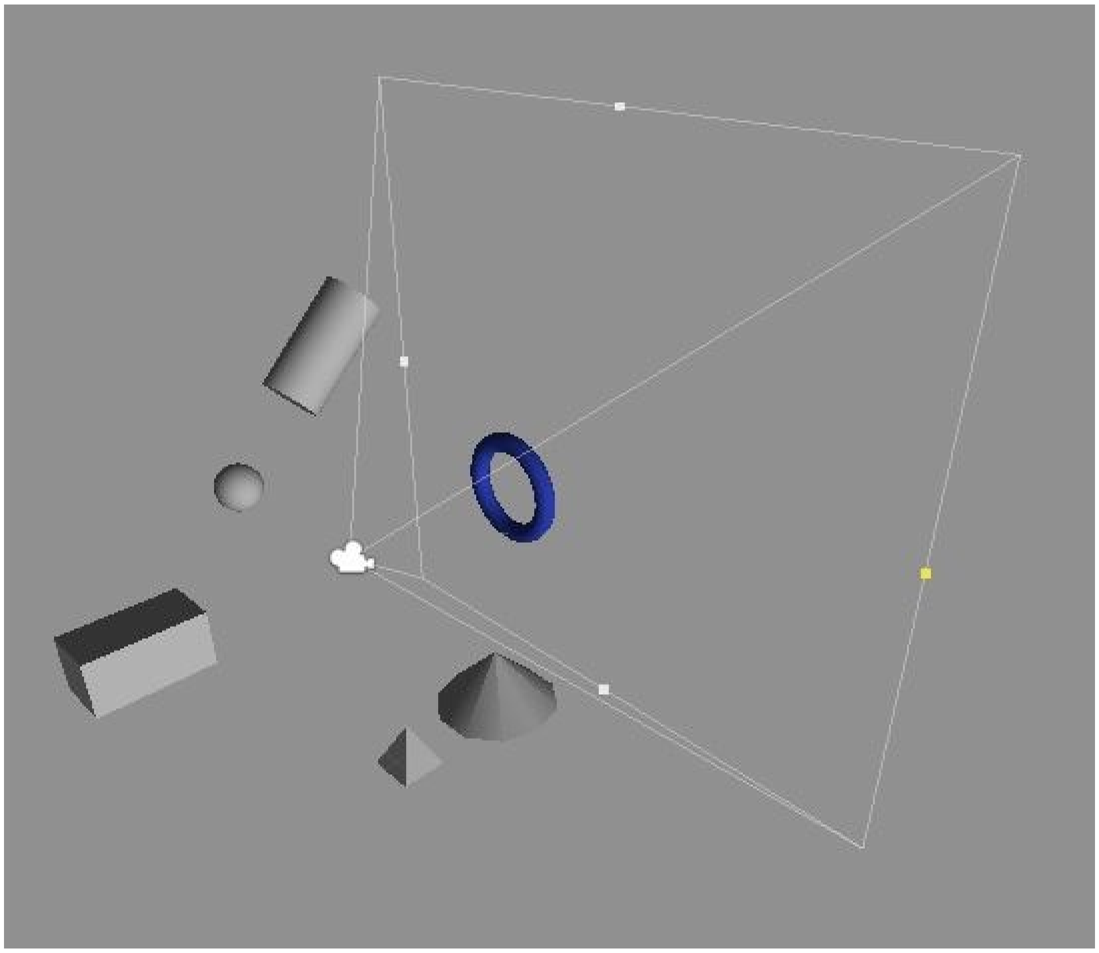


Figure 2-11. Illustration of scene geometry. The camera is pointing towards the positive x-axis, torus highlighted in blue

The cubemap is updated in real time, every time the camera viewpoint or view direction changes. The cubemap encodes the distances from the camera to all pixels (and thus to all visible objects). More specifically, the distance values are normalized in relation to the near and far planes. A visualization of these distances in world space coordinate system is shown in Figure 2-12. These values are stored in the alpha channel of the each pixel where an object is visible. The color channels for each pixel represent the direction of each of the ray through each pixel that intersect with a visible object and where red, green, blue are the X, Y and Z components of the direction vector and the range [-1, +1] is mapped to [0, 255] in each channel. In the images a grey color indicates that the ray in that direction intersected with no objects. This allows the cubemap to encode the visual depth/distance information of the environment at every point and at any given time without the need of additional preprocessing, a very desirable feature in the case of dynamic scenes.

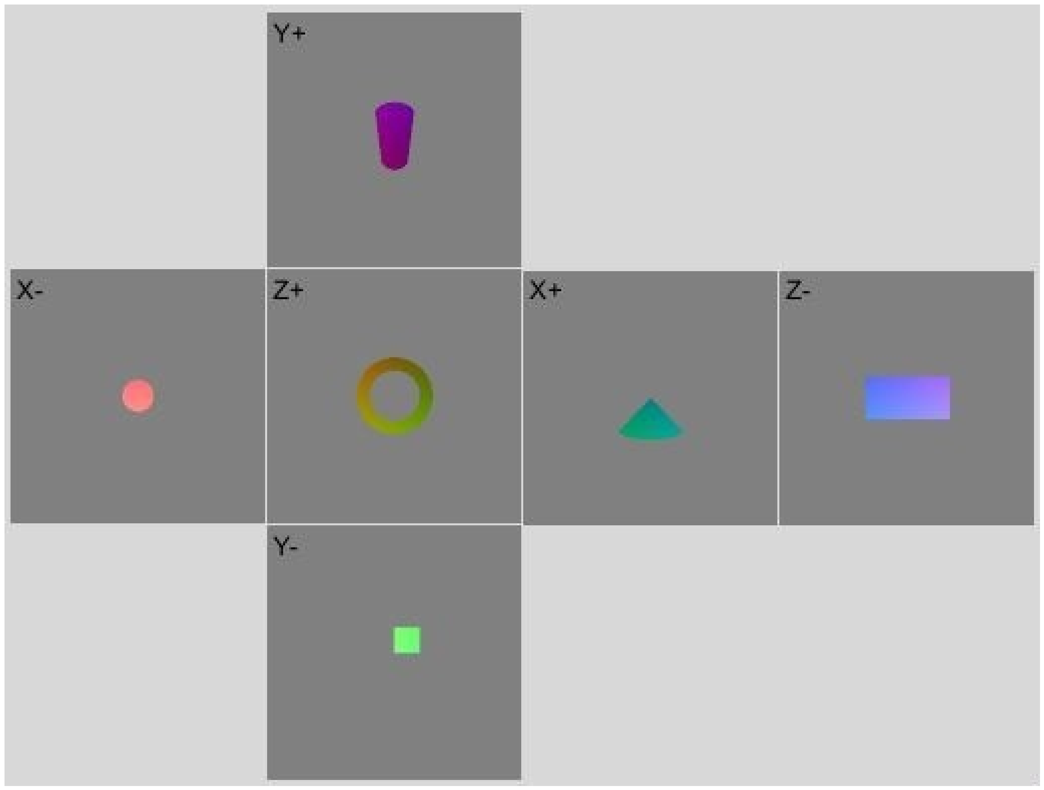


Figure 2-12. Depth information rendered to world space coordinate system cubemap of Figure 2-1

There is a cost associated with updating the cubemap as it requires six additional rendering steps, which can impact performance. Since McCrae's application of the cubemap estimates the distances to the environment through sampling, a lower resolution can ameliorate any performance loss. In their work a resolution of 64 x 64 was sufficient to reach the level of precision that typical users needed.

These distance values are then used to compute the average displacement vector as shown in the Equation 1bellow:

$$\frac{1}{6N_xN_y}\sum_{x,y,i} w\big(dist(x,y,i)\big) \cdot norm(pos(x,y,i) - eye)$$

Equation 1. McCrae's original average displacement vector equation

In the equation above N_x and N_y represent the horizontal and vertical resolutions and i is an integer with values between 1 and 6 inclusive representing the six sides of the cubemap. The soft penalty function *w()* is defined as:

$$w(dist) = sign(dist)e^{\frac{-\min(|dist|-\delta,0)^2}{2\sigma^2}}$$

Equation 2.The original soft penalty function as proposed by McCrae

The softness parameter σ presented in the above equation defines the transition function, but had not suggested value. The bound radius δ can be modified based on the current scene size or scale. As the soft penalty function is multiplied in the final calculation a penalty of zero expresses no influence of the environment on the user's speed, while a value of one expresses maximum influence. Together the cubemap and the equations compute a vector that displaces the camera in a way that adjusts speed, similar to the distance-dependent speed control presented by Ware and Fleet [75]. Through the weighting by distance, the direction of the vector adjusts the travel direction to avoid collisions.

Trindade et al. [72] improve two well-known navigation techniques in order to assist and facilitate navigating in a multiscale virtual environment. In their flying technique they include collision avoidance and automatic navigation speed adjustment with respect to the scale of the environment. One of the issues identified when flying close to geometry is that speed control via the global minimum can unnecessarily slow down the user. For example, in the case where the user is flying through a tunnel that has no occluding geometry straight ahead (i.e., sky at the end of the tunnel), nearby walls reduce the speed greatly, and therefore the user would fly very slowly. One of their proposed methods for this problem is to use the distance along a ray in the view direction to detect situations where the viewer can speed up. Using an exponentially weighted average between the distance along the view direction ray and the global minimum distance, they smooth out the rough speed changes. Despite using an exponentially weighted average between these two speeds, a speed computed for a distance of infinity or something equivalent will be so large that it will overwhelm any smaller speeds calculated using the minimum distance. This may cause user to move at huge speeds very close to geometry, which is undesirable. Moreover, the discrete nature of using a sampling ray can cause abrupt changes in speed, if said ray fall on/off geometry. In the examine technique they use a point-of-interest technique with automatic pivot point based on the construction and maintenance of a cubemap.

Argelaguet [2] proposes a new method of speed control that aims to keep optical flow constant. His dynamic speed technique adopts an efficient algorithm for speed adaptation to the virtual environment enhancing the effectiveness of dynamic speed in virtual visits. Argelaguet [2] also found that there is no strong difference between distance-based speed control and a method that keeps optical flow constant.

Speed can also be controlled using physical force-based devices, such as force-feedback joystick [40, 13].

2.4 Ray Casting

Ray casting is a technique for determining the distance to an object by sending out rays and determining the first objet intersected by a ray. This involves "casting" rays from the viewpoint ('camera'), through the viewing plane, and into the environment to an object.

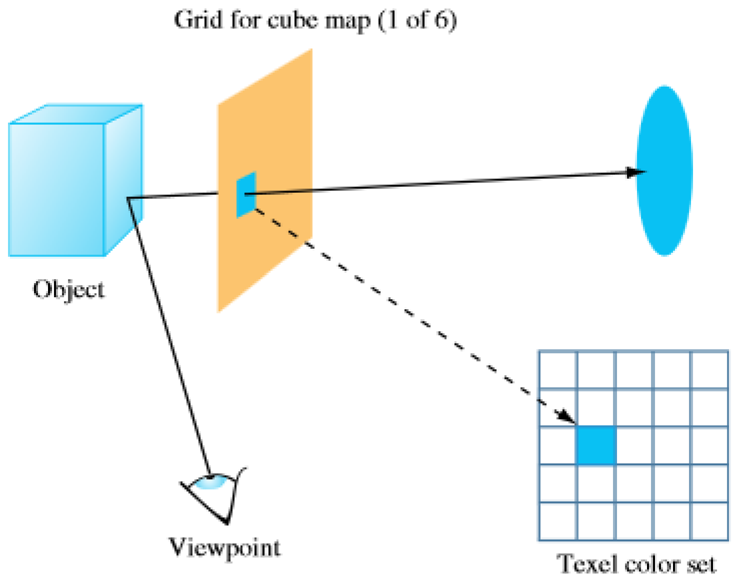


Figure 2-13 Creating a cubemap using ray casting.

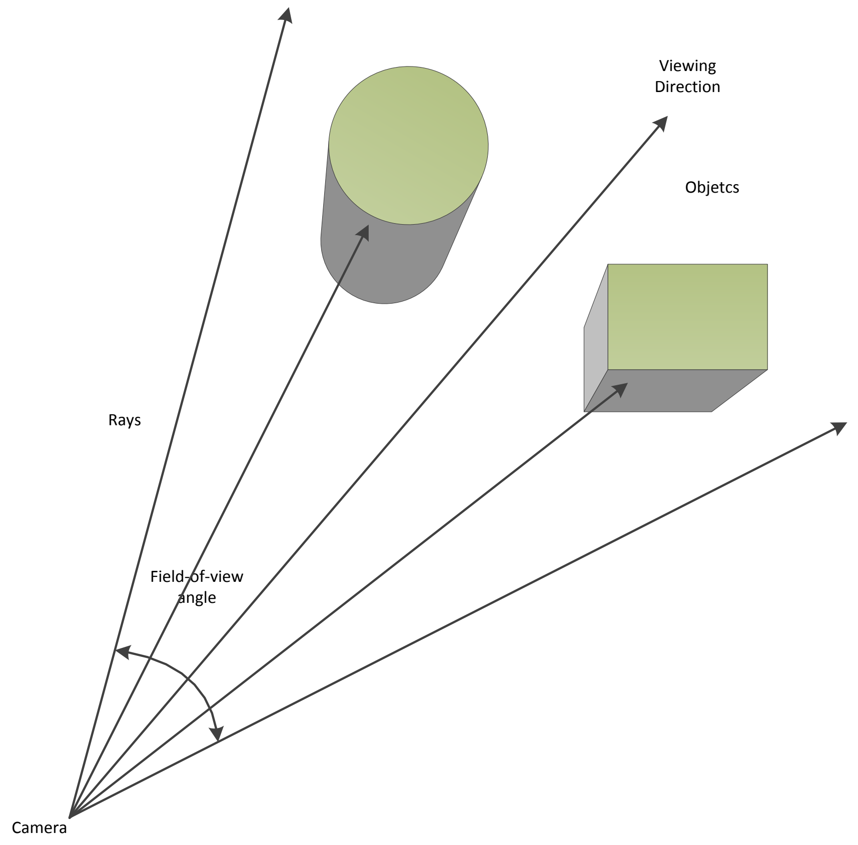


Figure 2-14. The basic ray casting model involves a camera or viewpoint (eye), a line casted from the viewpoint to the objects in the scene (ray).

Ray casting can be used to construct cubemaps. Rays are cast from the center of the cube, through texel grids positioned as the six faces of the cubemap, creating images for each face. The pixels for each image are computed by mapping a grid onto the cube face, corresponding to the desired texture resolution and then casting a ray from the center of the cube through each grid square color to a specific texel in the cubemap.

Chapter 3
Examining Automatic Speed Control for Navigation in 3D VE

In this chapter, my proposed method for speed control is discussed. Firstly, I will explain the automatic speed control technique. Then, I will present the experiments that I conducted to evaluate my novel method.

Similar to McCrae et al. [46], I am computing the distances to all objects around the viewer by generating a cube map of all objects. Instead of using a world space aligned cube map I am using a view aligned cube map that was introduced by Trindade et al. [72]. The reason for this is because a world space aligned cube does not "know" what geometry is where relative to the viewer. Therefore, by looking at a world space aligned cube map it is harder to tell where an object is relative to the viewer.

I am proposing an improvement to the equation by McCrae et al. [46] by scaling it with a smoothing function. The equation below computes the average displacement vector from the cube map over all pixels:

$$\frac{1}{6N_xN_y}\sum_{x,y,i} w\big(dist(x,y,i)\big)\cdot norm(pos(x,y,i)-eye)$$

In the above equation i is an integer value between [1, 6] and represents one side of the cube map. The horizontal and vertical resolutions are represented by N_x and N_y. McCrae also presented a soft penalty function as follows:

$$w(dist) = sign(dist)e^{-\frac{\min(|dist|-\delta,0)^2}{2\sigma^2}}$$

This penalty function gives a larger weight to geometry closer to the viewer. In this soft penalty function, σ is a softness parameter variable which has no explicit suggested values. A simpler alternative to the above exponential function is to use a smooth-step or an improved version of the smooth-step function which has zero 1st and 2nd order derivatives at *t=0* and *t=1* to determine how nearby geometry influences the viewer:

$$smoothstep(t) = 6t^5 - 15t^4 + 10t^3$$

$$w(dist) = \begin{cases} 1, & if\left(\frac{\min(dist,\delta)}{\delta} < \alpha\right), else \\ 1 - smoothstep\left(2 * \frac{\min(dist,\delta)}{\delta} - 1\right) & \end{cases}$$

Vs.

$$w(dist) = a\, exp\left(-\frac{(dist-\delta)^2}{2\sigma^2}\right) + d$$

Where δ represents the bound radius within which objects should affect the user, σ is a softness parameter and α is a dynamic penalty control variable with values between [0, 1]. As δ is constant across samples, the viewer's collision boundary is then a sphere with radius δ. The bound radius δ can be modulated by scale estimate which is the minimum distance from the cubemap. The value I have choose for the dynamic penalty control variable α was *0.5*.

As mentioned above in the review of previous work Trindade et al. [72] observed that the speed may be perceived to be too low when navigating in long narrow tunnels. Yet, their approach with a ray introduces a binary response. To address this issue in a better way, I propose to multiply the inner terms with a second weighting term, which scales the contribution of geometry close to the view direction with a smooth fall off for geometry orthogonal or behind the viewer. My first idea here is to use a squared cosine of the angle relative to the view direction.

$$w_2(dist) = \max(\cos^2(\theta), 0)$$

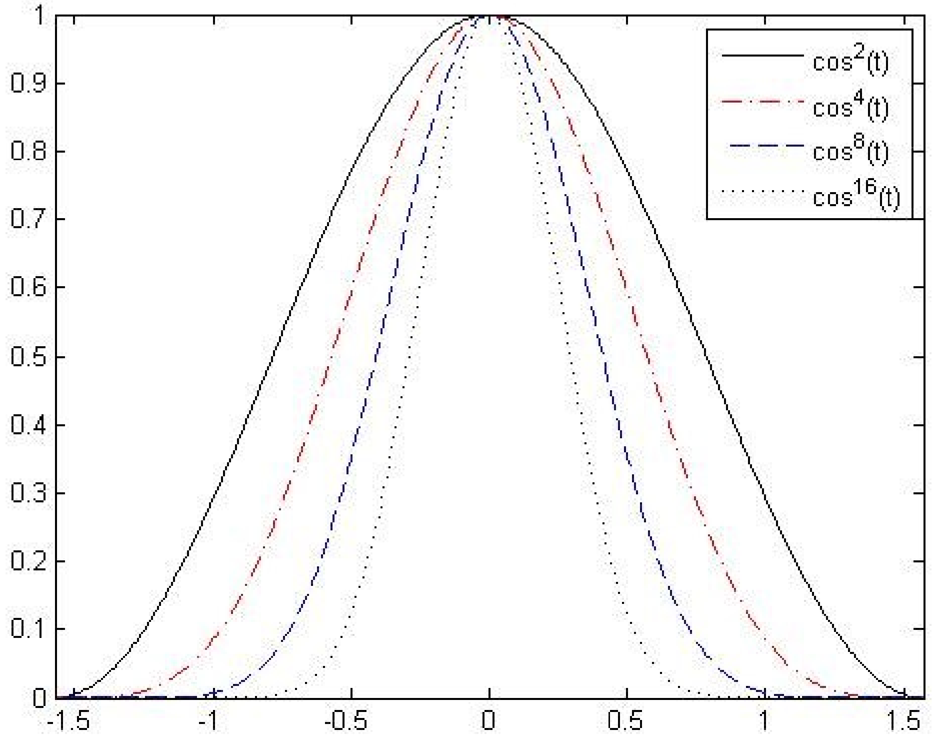


Figure 3-1 Plot of powers of cosine, it shows $\cos^{16}$ will narrow better the view direction

After testing the squared cosine implementation, I found that the surrounding environment had a great drag effect on speed, it was slowing down the user unnecessarily in close spaces. Increasing the power to 4, 8, and 16 resulted in improved speed, best results achieved with power of 16. The experiments presented later in this thesis were conducted with a cosine power of 16.

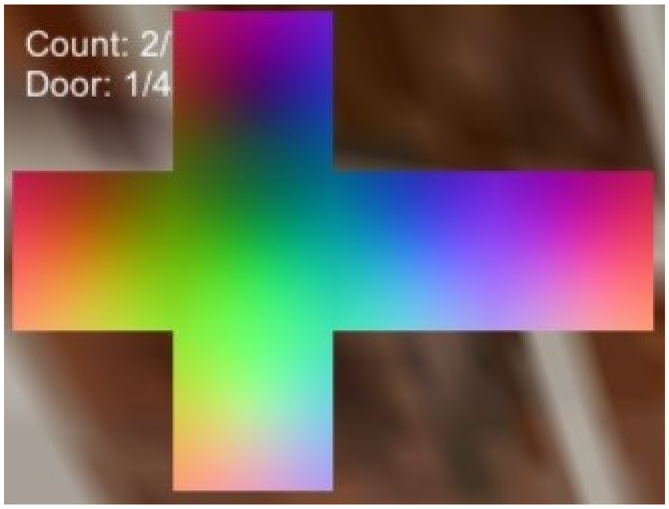


Figure 3-2 Illustration of depth buffer without second term (cosine power)

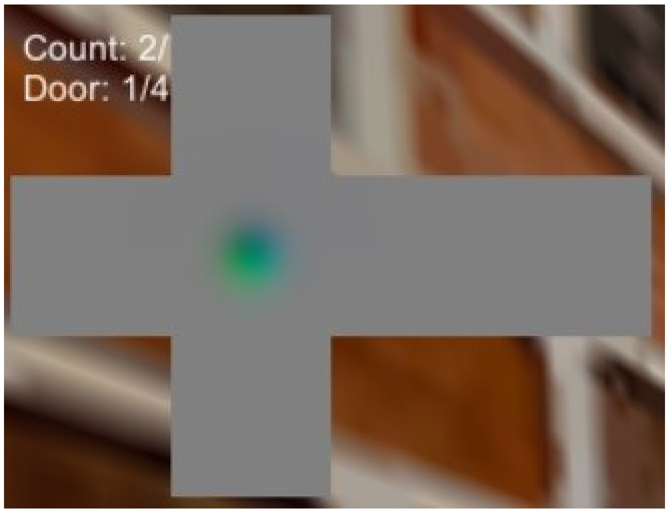


Figure 3-3 Illustration of depth buffer with second term ($\cos^{16}$) applied

As shown in Figure 3-3 applying the second term *w_2(dist)* reduces the influence of the surrounding objects on speed as opposed to no weighing term Figure 3-2.

3.1 Experiment 1

The objective of this study was to evaluate my proposed automatic speed control (Technique C) and to compare it with the speed control via the global minimum described by McCrae et al. [46] (Technique A) as well as the automatic speed adjustment developed by Trindade et al. [72] (Technique B).

3.1.1 Participants

I recruited 14 participants (11 male, 3 female) for this study aged from 23 to 45 (mean age 31.8 years, *SD* 8.35). One participant found the task too difficult in the practice session and declined to continue and one felt dizzy in the practice session and was instructed to not participate. All participants had used VEs before, played FPS games or 3D race car games.

3.1.2 Setup

The experiments were conducted with a 24" wide screen monitor (HP ZR24w) with a resolution of 1920 x 1200 pixels, with Microsoft IntelliMouse Optical mouse as the only input device. The distance between the user and the monitor was approximately 50 cm.

To evaluate my new navigation technique, I was inspired by Argelaguet's experimental design and the virtual environment they used in their work [2]. The virtual environment that I built for my experiment consisted of two different scene configurations. The first scene is a uniform section configured as a maze-like environment. The second scene is configured as a non-uniform section filled with geometrical objects (see Figure 3-4). Furthermore, I used three different levels of scale (1:1, 1:2 and 1:10).

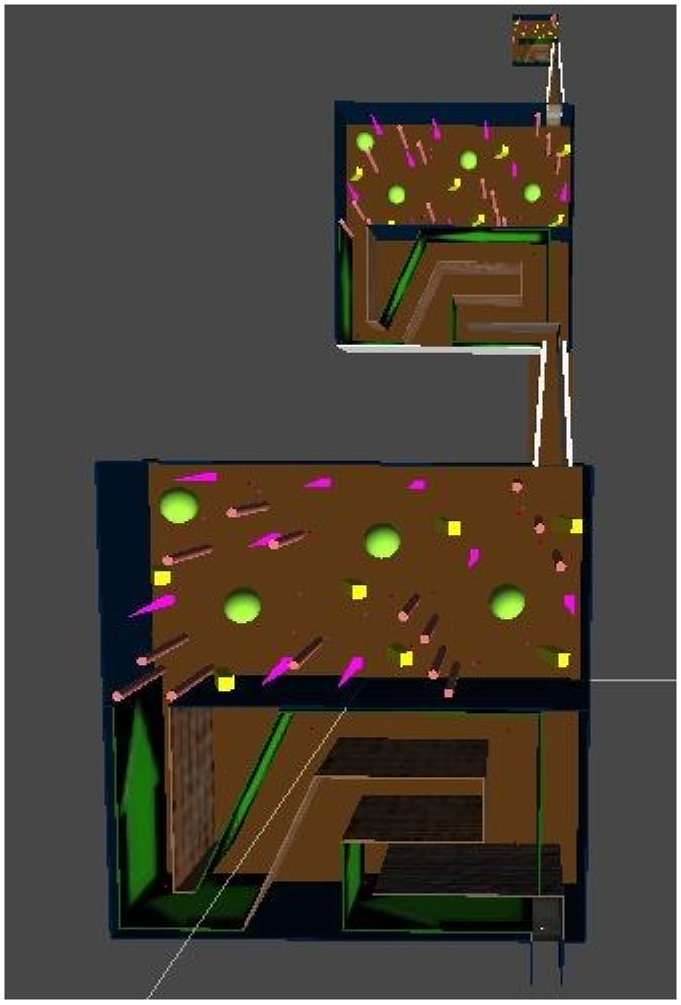


Figure 3-4: Virtual environment used for the navigation task. The VE has 2 different sections and three different levels of scale (1:1, 1:2 and 1:10).

To guide users along the way I painted arrows on the maze walls to show the direction of the path to be followed in case the user turns around. For depth perception I used different textured walls (brick/stone). For enticing users to follow a certain path I added pick-up rotating red cubes (see Figure 3-5). Once the user collides with the cubes the cubes were removed from the environment and the pick-up was accompanied by acoustic sound. In the geometry section of the environment these pick-up cubes were connected through visible rays so the user would know which cube is next. The rays were removed when the target cube is reached (see Figure 3-6).

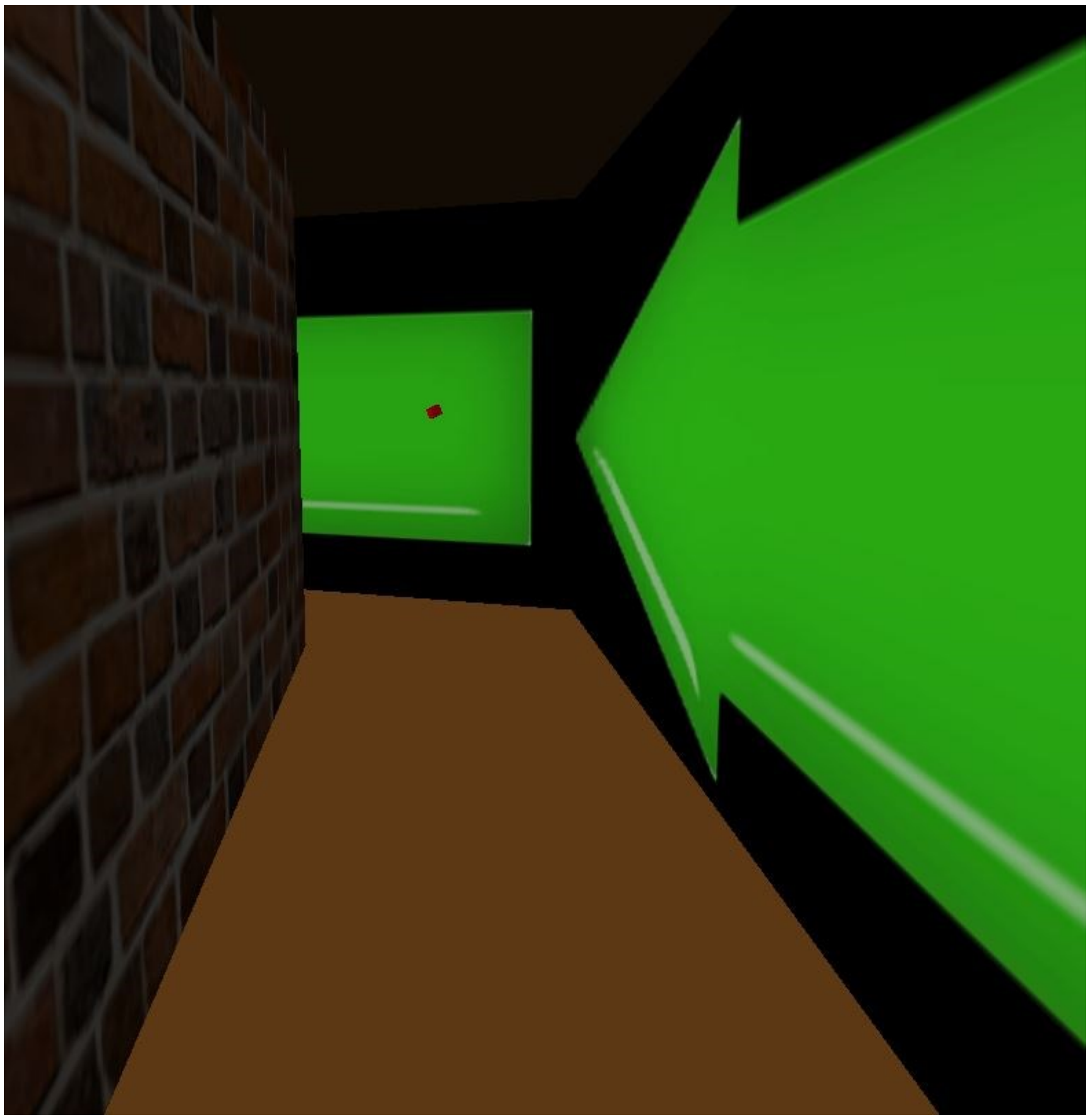


Figure 3-5: First section of the environment, the maze, with directional arrows pointing towards pick-up objects and textured walls.

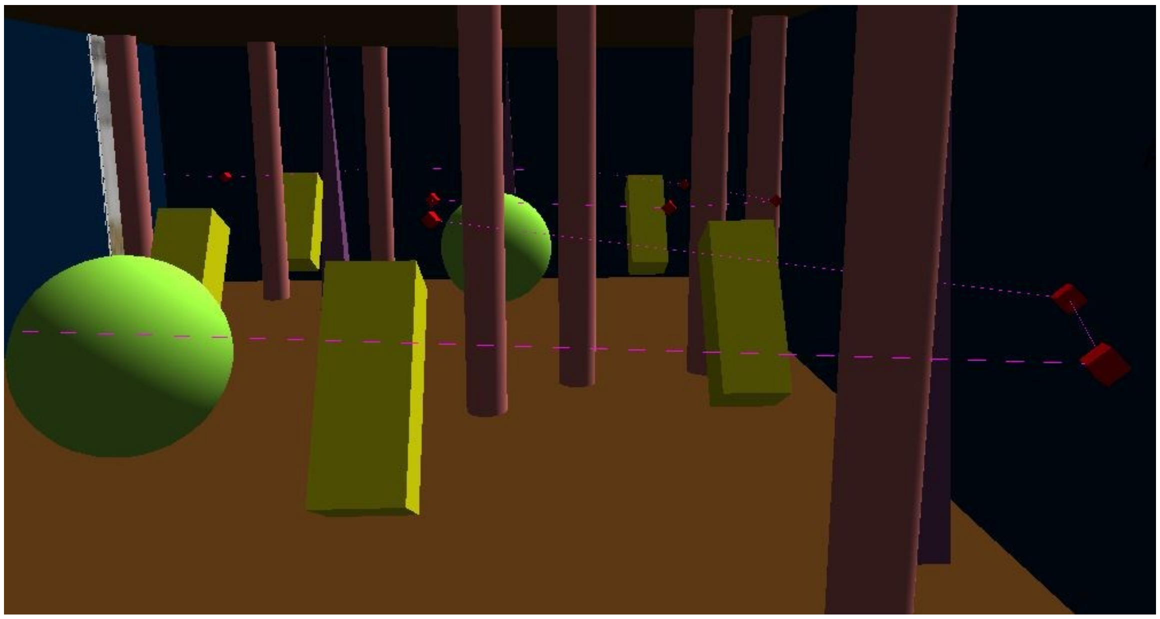


Figure 3-6: Second section of the environment with geometry objects showing pick-up objects connected through visible rays.

3.1.3 Procedure

First, each participant was given a brief questionnaire about his/her background. The questionnaire recorded gender, age, and previous experience with 3D virtual environments. Then, the participant was instructed to use the VE and was encouraged to practice until him/her felt comfortable. All users used the mouse as the only means to navigate the environment. Left mouse button was forward, right mouse button was used for backward movement. With no button pressed the users could orient themselves in the virtual environment. The order of the techniques was counterbalanced with a Latin square (see Figure 3-12) design across all participants in order to avoid learning effects (see Figure 3-13). The order of the sections (Maze, Geometry) and the scale factor was fixed due to the design of the virtual environment (see Figure 3-4). All three navigation techniques were having the same settings: same radius, far plane, smoothness of the collision factor. Once the participants were comfortable with the VE, they were instructed to traverse the VE as fast as possible following the path marked by red pick-up cubes. The participant was instructed to try to pick-up these target cubes as quickly and accurately as possible but not to be concerned by the fact that the pick-up was not successful. Overall, the study took about half an hour per participant.

3.1.4 Results

Data was first filtered for participant errors, such as disorientation in the VE, deviating from the path or pausing in the middle of the navigation to focus his/her attention elsewhere. Then, these errors and outliers were removed (i.e., results with more than three standard deviations from the mean, amounted to a 3% loss of total data collected).

3.1.5 Average Speed

To analyze the average speed across all three scales I have multiplied the speeds from the half scale environment by 2 and the speeds from the 1:10 scale by 10. With this adjustments the data was normally distributed.

The one-way ANOVA of technique versus speed showed a main effect on technique ($F_{2,22} = 3.39$, $p < .05$). See Figure 3-7 for average speed.

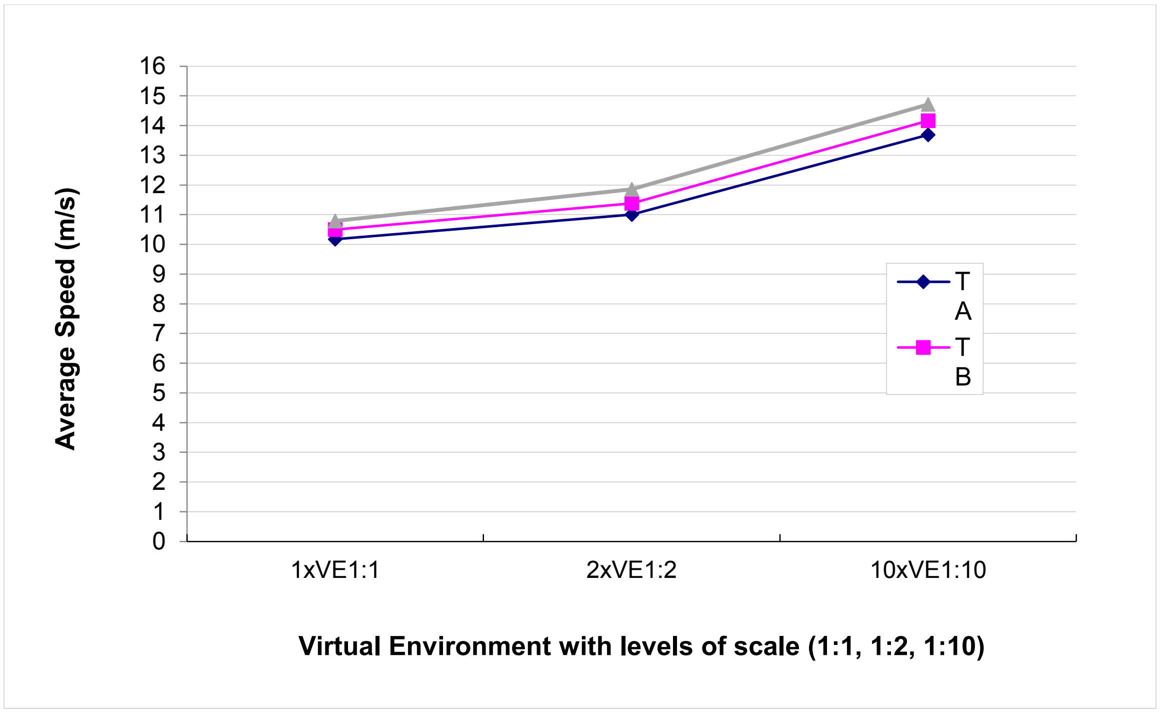


Figure 3-7 Average speed in m/s for each scale and technique.

Post hoc test showed that the mean speed values for technique C were higher (M = 12.45 m/s; SD = 2.03 m/s) than technique B (M = 12.01 m/s; SD = 1.91 m/s) and mean speed values for technique A (M = 11.62 m/s; SD = 1.84 m/s) were slower than technique B (see Figure 3-8), where technique A is McCrae's method, technique B is Trindade's method and technique C is my proposed method.

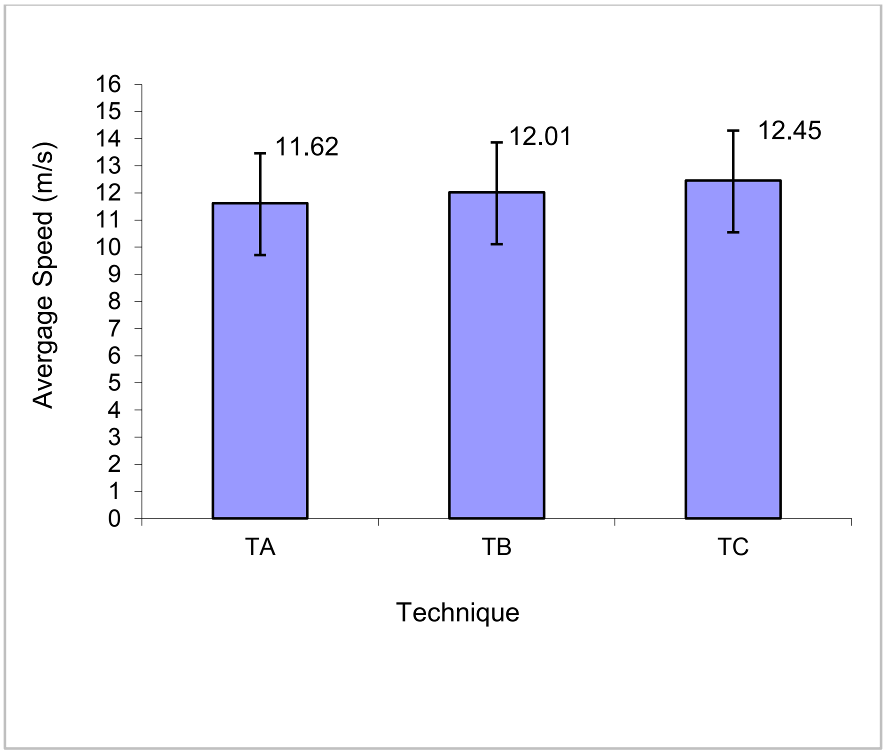


Figure 3-8 Mean speed across all scales

The graph shows the average speed in m/s across all participants. Overall technique C has higher speeds than technique B or A.

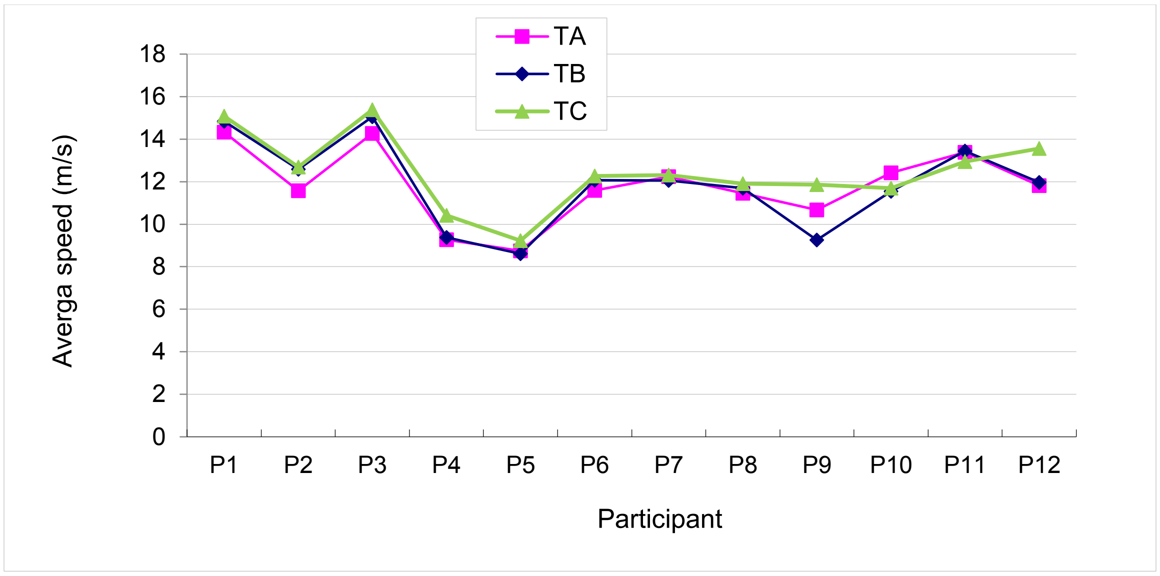


Figure 3-9 Average speed for each participant

3.1.6 Task Completion Time

The data for task completion time was not normally distributed. Levene's test for homogeneity revealed that the data did not have equal variances [Appendix A:]. The Aligned Rank Transform for nonparametric factorial data analysis [Appendix B:] has been used and a repeated measures parametric ANOVA has been performed on the transformed data.

There is a significant effect of completion time on technique ($F_{2,22} = 8.14$, $p < .01$). See Figure 3-10 for task completion times for each scale of the environment.

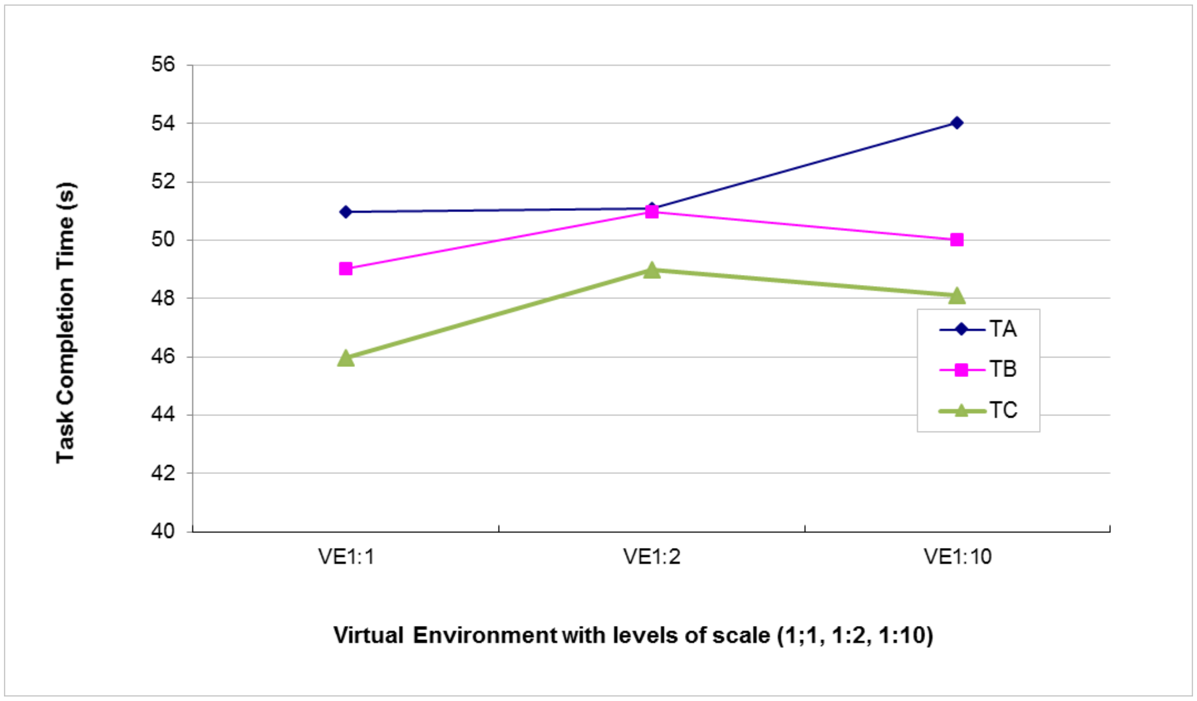


Figure 3-10: Graph depicting the task completion time (s) for each scale of environment.

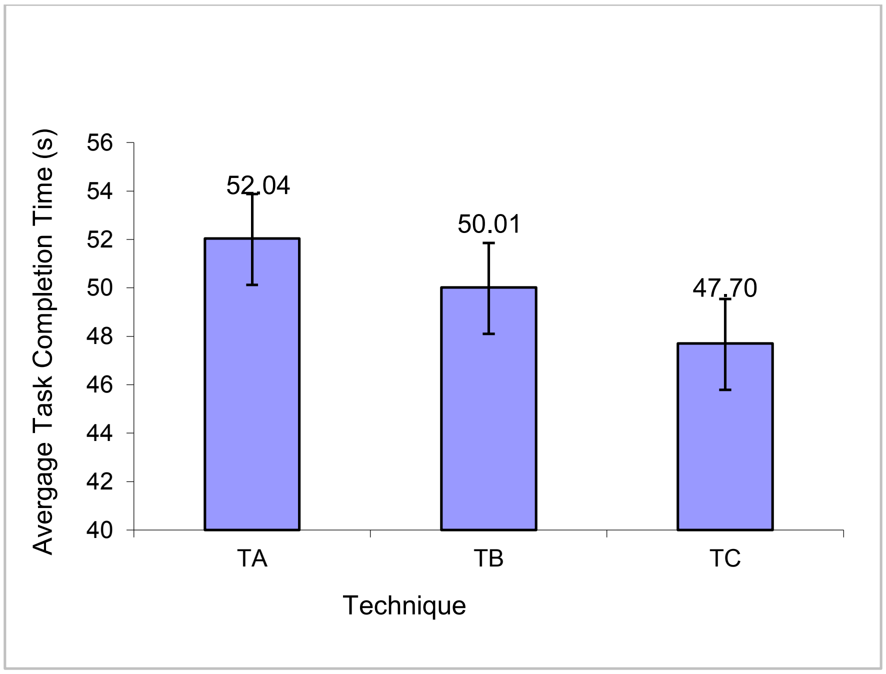


Figure 3-11: Graph depicting the average task completion time for each technique.

3.1.7 Learning

Over all participants, no significant learning effects could be detected, but this does not mean that users did not learn.

A	B	C
B	C	A
C	A	B

Figure 3-12. 3 x 3 Latin Square

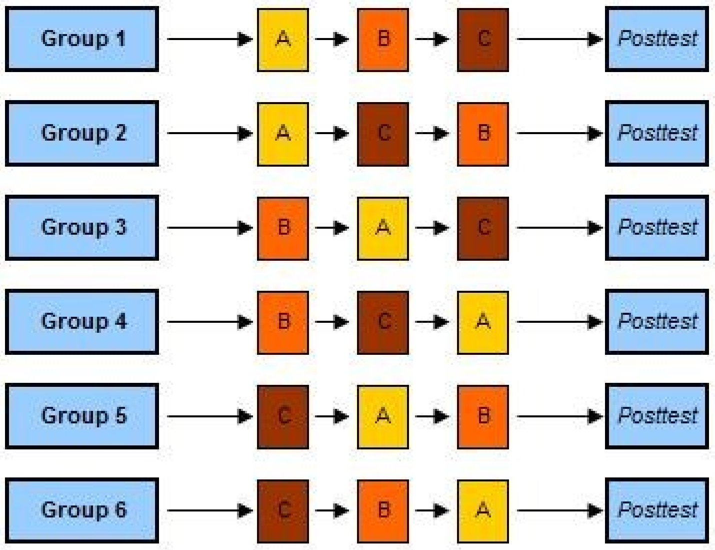


Figure 3-13. Counterbalanced measures design with 3 condition and 6 groups

3.1.8 Other Results

The data has also been analyzed to see if counterbalancing had worked correctly. The ANOVA test on group effect was not significant ($F_{5,6} = 0.652$). A non-significant group effect means counterbalancing worked. Thus, any learning that may have taken place was balanced out.

3.1.9 Discussion

The overall conclusion from this study is that my implementation allowed for a smooth navigation with a performance comparable with state of the art navigation approaches. In comparison to two previously presented methods (McCrae and Trindade) my method had an improved speed allowing the users to achieve the goal in less time.

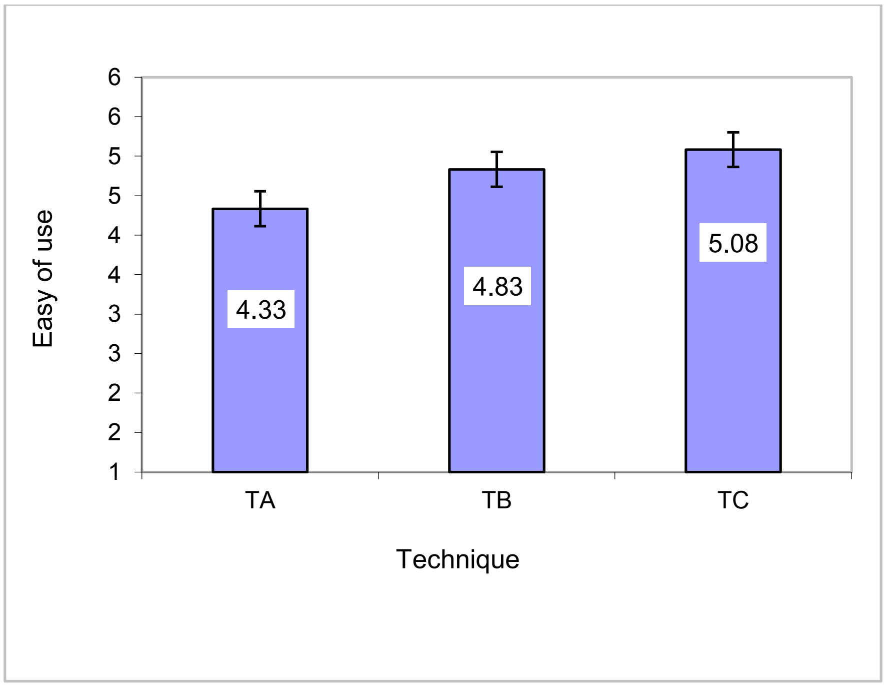


Figure 3-14. Graph showing user feedback regarding ease of use of each technique

Observing the participants and their comments during the experiment I found that they felt that the technique C was more comfortable leading to better user satisfaction. The data collected in the user questionnaire regarding ease of use and smoothness confirms my finding (see Figure 3-14 and Figure 3-15). However implementing a system that allows automatic speed change for a dynamic range of scales reveals specific issues which need to be accounted.

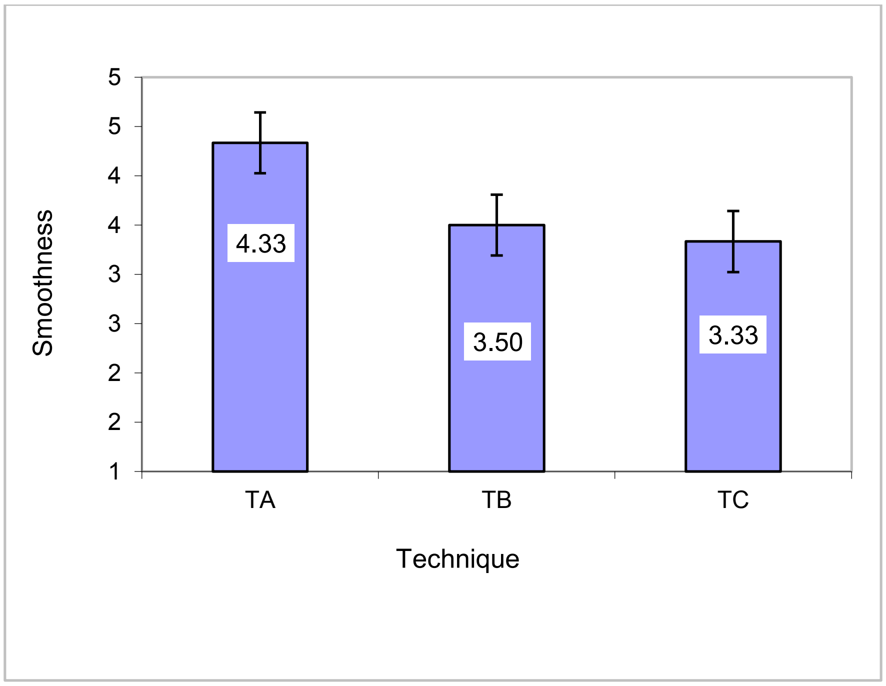


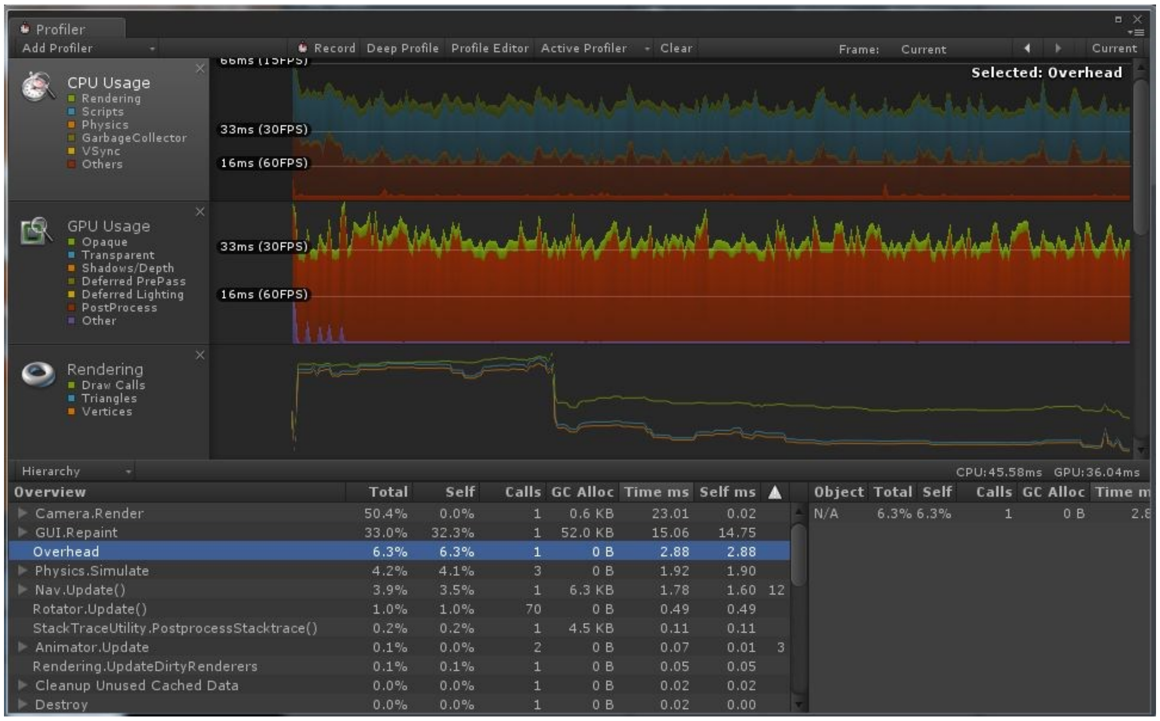
Figure 3-15. Graph showing user feedback regarding speed smoothness of each technique (less is better)

One of the problems that I have faced while navigating between scales was the lack of floating point number precision in the graphics hardware for the speed calculation and for the push back vector output of Equation 1, especially in zoomed in scenes. The effect on the user was a drift away from the close-up object, also if the user was too close to the object the object appeared to "jitter".

To address this issue, I have scaled the max scene scale to 10:1 and the user's zoom was also limited to avoid this "jitter" effect.

Even though I have used a resolution of 1024x1024, for this particular case of navigating through closed spaces (i.e. rooms, tunnels) the resolution could have been reduced with no significant effect on the speed outcome.

I have performed two GPU benchmarking one with the automatic speed control and one without (manual speed).



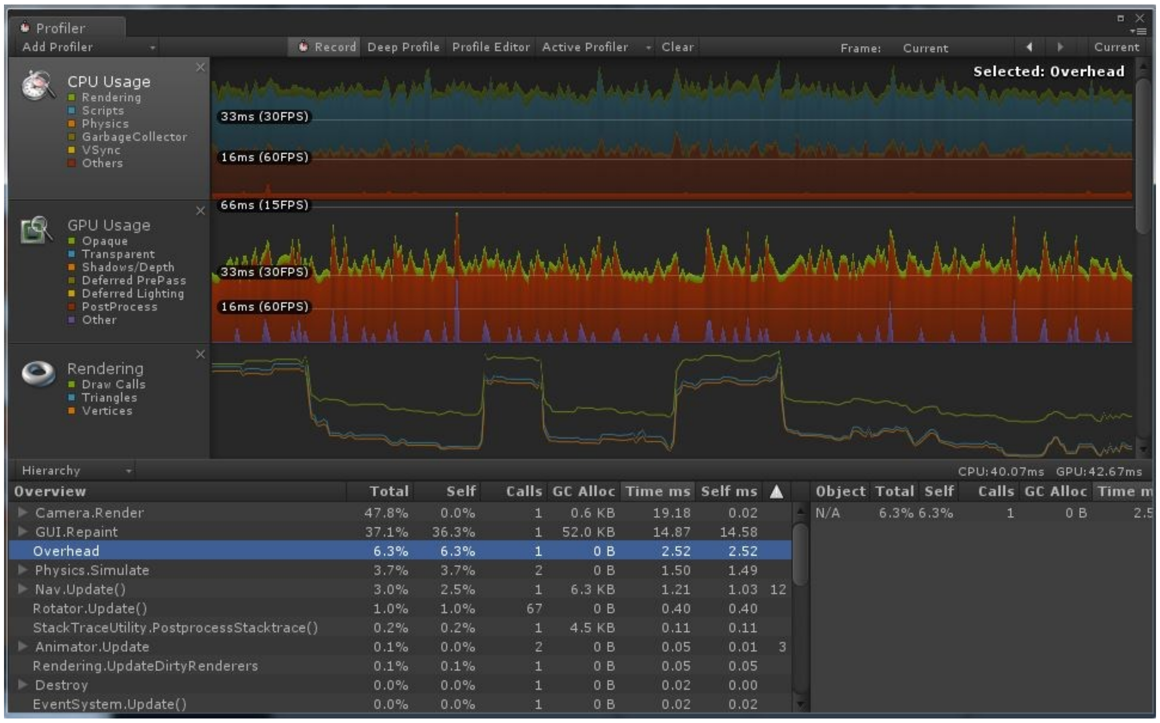
CPU:45.58ms GPU:36.04ms

Overview	Total	Self	Calls	GC Alloc	Time ms	Self ms	▲
Camera.Render	50.4%	0.0%	1	0.6 KB	23.01	0.02	
GUI.Repaint	33.0%	32.3%	1	52.0 KB	15.06	14.75	
Overhead	6.3%	6.3%	1	0 B	2.88	2.88	
Physics.Simulate	4.2%	4.1%	3	0 B	1.92	1.90	
Nav.Update()	3.9%	3.5%	1	6.3 KB	1.78	1.60	12
Rotator.Update()	1.0%	1.0%	70	0 B	0.49	0.49	
StackTraceUtility.PostprocessStacktrace()	0.2%	0.2%	1	4.5 KB	0.11	0.11	
Animator.Update	0.1%	0.0%	2	0 B	0.07	0.01	3
Rendering.UpdateDirtyRenderers	0.1%	0.1%	1	0 B	0.05	0.05	
Cleanup Unused Cached Data	0.0%	0.0%	1	0 B	0.02	0.02	
Destroy	0.0%	0.0%	1	0 B	0.02	0.00	

Object	Total	Self	Calls	GC Alloc	Time m
N/A	6.3%	6.3%	1	0 B	2.8

Figure 3-16. GPU profiling with automatic speed control

The overall overhead for the version with automatic speed control was 2.88 ms (see Figure 3-16) compared with 2.52 ms (see Figure 3-17) for the version without automatic speed control, resulting in a GPU overhead of 0.36 ms per frame.



CPU:40.07ms GPU:42.67ms

Overview	Total	Self	Calls	GC Alloc	Time ms	Self ms	▲
▶ Camera.Render	47.8%	0.0%	1	0.6 KB	19.18	0.02	
▶ GUI.Repaint	37.1%	36.3%	1	52.0 KB	14.87	14.58	
Overhead	6.3%	6.3%	1	0 B	2.52	2.52	
▶ Physics.Simulate	3.7%	3.7%	2	0 B	1.50	1.49	
▶ Nav.Update()	3.0%	2.5%	1	6.3 KB	1.21	1.03	12
Rotator.Update()	1.0%	1.0%	67	0 B	0.40	0.40	
StackTraceUtility.PostprocessStacktrace()	0.2%	0.2%	1	4.5 KB	0.11	0.11	
▶ Animator.Update	0.1%	0.0%	2	0 B	0.05	0.01	3
Rendering.UpdateDirtyRenderers	0.1%	0.1%	1	0 B	0.05	0.05	
▶ Destroy	0.0%	0.0%	1	0 B	0.02	0.00	
EventSystem.Update()	0.0%	0.0%	1	0 B	0.02	0.02	

Object	Total	Self	Calls	GC Alloc	Time m
N/A	6.3%	6.3%	1	0 B	2.5

Figure 3-17. GPU profiling without automatic speed control

Chapter 4
Overall Discussion

In general, I can state that my new solution solved the problem that excessive slowdowns occur with previously presented methods for automatic speed control when the viewer slides along walls or navigates through tunnels. In my system automatic speed controls takes all the geometry that surrounds the user into account instead of just what is visible in the camera view, such as (McCrae [46], Trindade [72]). For example, in a virtual environment for a star system, if the user is on the surface of a planet and looks up into the vast empty space between planets the system will launch the user into space at startling high speeds. The logic behind this is that the system does not detect any geometry in front of the user, and calculates the speed accordingly, assuming user wants to navigate with extremely high speed. In my system I take into account the invisible geometry behind the user and start the speed at slower values increasing it exponentially allowing the user to just lift off the ground, as if that was the real intent. The speed will increase really fast if the user keeps moving towards the outer space. Similarly, when the user approaches an object, such as a faraway planet, with high speeds, the system will slow the user down even if the user is not aiming directly at the planet, which will avoid overshooting the target. However, as noted by Trindade [72], this strategy can slow the user down too much in certain scenarios. For example, if the user navigates through a tunnel or labyrinth/maze the system will reduce the speed drastically based on the close proximity to the walls, which could be frustrating. My approach addresses this issues by weighting the influence of geometry behind or beside the viewer less than the influence of geometry directly ahead, with a smooth interpolation in between to guarantee smooth transitions. In essence, this allows the user to navigate parallel to an existing plane at a higher speed then would be achievable if navigating perpendicular towards the plane. By combining the surround geometry distance with the forward distance the navigation speed is adequate when navigating away from nearby geometry and results in a smooth "takeoff" behavior.

The system presented here is designed to help navigation on touch systems with only one hand. Because the technique requires only a 2D input, it can be easily used on different 2D devices, such as touch, pen screen or mouse. A single finger drag controls the look direction, two fingers touch and hold will move the user forward based on the current user's view for the duration of the hold. Steering is achieved by dragging both fingers in the desired direction. Backward movement is achieved with 3 fingers touch and hold. Similarly, on a desktop environment a mouse move will control the user's view direction, holding the left mouse button down will move the viewer forward with mouse movements mapped to steering. The right mouse button is mapped to backward navigation.

Finally, note that for scenarios where objects need to be picked up and moved around the scene there needs to be a decoupling mechanism to distinguish between object manipulation and viewer movement.

4.1 Limitations

As presented in previous section the output of Equation 1 is a single 3-space vector that pushes the user away from the nearest geometry. This vector will move the viewer towards the center area of any closed virtual environment and will push the user out of rooms with openings, as mentioned by McCrae [46]. As the environment scale is decreased the vector will increase and will start moving the user away from nearby geometry even without any action on the users' part. This can lead to confusion and frustration in some situations, such as the viewer being extremely close to a surface while looking parallel to it. Then the algorithm will modify the user's actions substantially and create a local "drift" effect even though there is a clear front path in the view. This may be counteracted by reducing the radius δ at which objects affecting the movement in Equation 2 by some function of scale or other dynamically calculated value adapted to the desired use case, rather than constant value. In my experiment I only explored different scales of the world of 1:1, 2:1 and 10:1 with the speed decreasing by half and 10 times compared with the 1:1 situation. If I would proceed even further down on the scale, i.e., explore scale differences of (say) 1000000:1, the speed would in theory adjust itself to this scale. Yet, because of the limitations of floating point number precision in the graphics hardware the result will be zero speed resulting in no movement at all.

Chapter 5
Conclusion

To summarize, the major contributions of this thesis revolve around automatic speed control for 3D navigation in VEs. I have proposed a new and efficient way to automatically adapt a user's speed. This new method takes geometry in both the surrounding environment and the area that is directly in front of the user into account. By using multipass shaders, this technique can be substantially accelerated relative to CPU based techniques. This enables us to take several orders of magnitude, collect more samples and do more computations.

While we have brought our study to fruition, research is a continuous process. Future work can be focused on investigation of techniques for navigating vast empty spaces, such as a star system. Specifically, if the distance between objects increases beyond what can be represented on graphics hardware within the limitations of floating point number precision. As future work, it is worthwhile to explore setting clipping planes at the bound radius of influence instead of just the scene far clip plane reduced by a fixed amount. After all, anything rendered beyond the bound radius will yield zero contribution due to the weighting function. Thus, a much closer far plane can be used for distance computations, which in turn will speed up rendering of the cubemap.

Porting this system to the mobile/tablet world requires heavy optimization on the computational side both on the GPU and the CPU. For the user study I have used an

Android device with one of the latest Qualcomm Snapdragon 805 chipsets with Adreno 420 graphics. Out of the box, the system was rendering the virtual environment at 4 fps resulting in an unacceptably slow navigation behaviour. As far as I could tell, the main slowdown was caused by internal limitations of the Unity3D compiler or the graphics chipset. Thus, I have performed several computational optimizations and also reduced the size of the depth textures to less than 1024x1024, to significantly reduce bandwidth requirements. In the future, other options would be to port the code to the latest Unity3D version 5 and use an even newer Qualcomm Snapdragon processor the Snapdragon 820 with Adreno 530 GPU, which now supports OpenGL ES 3.1.

Bibliography

1. Anthes, Christoph, Paul Heinzlreiter, Gerhard Kurka, and Jens Volkert. "Navigation models for a flexible, multi-mode VR navigation framework." In Proceedings of the 2004 ACM SIGGRAPH international conference on Virtual Reality continuum and its applications in industry, pp. 476-479. ACM, 2004.
2. Argelaguet-Sanz, Ferran. "Adaptive Navigation for Virtual Environments." In IEEE Symposium on 3D User Interfaces 2014, pp. 91-94.
3. Bakker, Niels H., Peter J. Werkhoven, and Peter O. Passenier. "Aiding orientation performance in virtual environments with proprioceptive feedback." In Virtual Reality Annual International Symposium, 1998. Proceedings., IEEE 1998, pp. 28-33. IEEE, 1998.
4. Beckhaus, Steffi, Kristopher J. Blom, and Matthias Haringer. "ChairIO–the chair-based Interface." Concepts and technologies for pervasive games: a reader for pervasive gaming research 1 (2007): 231-264.
5. Bowman, Doug A., and Larry F. Hodges. "An evaluation of techniques for grabbing and manipulating remote objects in immersive virtual environments." In Proceedings of the 1997 symposium on Interactive 3D graphics, pp. 35-ff. ACM, 1997.
6. Bowman, Doug A., D. Johnson, and L. Hodges. "Testbed environment of virtual environment interaction." ACM VRST'99 Symposium on Virtual Reality Software and Technologies. 1999.

7. Bowman, Doug A., David Koller, and Larry Hodges (1998). A Methodology for the Evaluation of Travel Techniques for Immersive Virtual Environments. Virtual Reality: Research, Development, and Applications 3: 120–131.
8. Bowman, Doug A., David Koller, and Larry F. Hodges. "Travel in immersive virtual environments: An evaluation of viewpoint motion control techniques." In Virtual Reality Annual International Symposium, 1997., IEEE 1997, pp. 45-52. IEEE, 1997.
9. Bowman, Doug A., E. Davis, A. Badre, and L. Hodges (1999). Maintaining Spatial Orientation during Travel in an Immersive Virtual Environment. Presence: Teleoperators and Virtual Environments 8(6): 618–631.
10. Bowman, Doug A., Ernst Kruijff, Joseph J. LaViola Jr, and Ivan Poupyrev. 3D user interfaces: theory and practice. Addison-Wesley, 2004.
11. Bowman, Doug A., Jean Wineman, Larry F. Hodges, and Don Allison. "Designing animal habitats within an immersive VE." IEEE Computer Graphics and Applications 5 (1998): 9-13.
12. Bowman, Doug A., C. Wingrave, J. Campbell, and V. Ly (2001). Using Pinch™ Gloves for Both Natural and Abstract Interaction Techniques in Virtual Environments. HCI International, New Orleans, Louisiana.
13. Brienza, David M., and Jennifer Angelo. "A force feedback joystick and control algorithm for wheelchair obstacle avoidance." Disability and Rehabilitation 18, no. 3 (1996): 123-129.

14. Brogan, David C., Ronald A. Metoyer, and Jessica K. Hodgins. "Dynamically simulated characters in virtual environments." Computer Graphics and Applications, IEEE 18, no. 5 (1998): 58-69.

15. Brooks Jr, Frederick P. "What's real about virtual reality?." Computer Graphics and Applications, IEEE 19, no. 6 (1999): 16-27.

16. Brooks Jr, Frederick P. "Walkthrough—a dynamic graphics system for simulating virtual buildings." In Proceedings of the 1986 workshop on Interactive 3D graphics, pp. 9-21. ACM, 1987.

17. Butterworth, Jeff, Andrew Davidson, Stephen Hench, and Marc T. Olano. "3DM: A three dimensional modeler using a head-mounted display." In Proceedings of the 1992 symposium on Interactive 3D graphics, pp. 135-138. ACM, 1992.

18. Chance, Sarah S., Florence Gaunet, Andrew C. Beall, and Jack M. Loomis. "Locomotion mode affects the updating of objects encountered during travel: The contribution of vestibular and proprioceptive inputs to path integration." Presence 7, no. 2 (1998): 168-178.

19. Chen, Michael, S. Joy Mountford, and Abigail Sellen. "A study in interactive 3-D rotation using 2-D control devices." In ACM SIGGRAPH Computer Graphics, vol. 22, no. 4, pp. 121-129. ACM, 1988.

20. Chung, James C. "A comparison of head-tracked and non-head-tracked steering modes in the targeting of radiotherapy treatment beams." Proceedings of the 1992 symposium on Interactive 3D graphics. ACM, 1992.

21. Cockburn, Andy, and Bruce McKenzie. "Evaluating spatial memory in two and three dimensions." International Journal of Human-Computer Studies 61, no. 3 (2004): 359-373.

22. Darken, Rudolph P., William R. Cockayne, and David Carmein. "The omni-directional treadmill: a locomotion device for virtual worlds." In Proceedings of the 10th annual ACM symposium on User interface software and technology, pp. 213-221. ACM, 1997.

23. Darken, Rudy P., and John L. Sibert. "A toolset for navigation in virtual environments." Proceedings of the 6th annual ACM symposium on User interface software and technology. ACM, 1993.

24. Deisinger, Joachim, Ralf Breining, A. Robler, D. Ruckert, and J. J. Hofle. "Immersive ergonomic analyses of console elements in a tractor cabin." In 4th International Immersive Projection Technology Workshop Proceedings, pp. 19-20. 2000.

25. Dörr, K. U., Schiefele, J., & Kubbat, W. (2000). "Virtual Cockpit Simulation for Pilot Training", Institute for Flight Mechanics and Control Technical University Darmstad.

26. Fairchild, Kim M., Beng Hai Lee, Joel Loo, Hern Ng, and Luis Serra. "The heaven and earth virtual reality: Designing applications for novice users." *Virtual Reality Annual International Symposium*. IEEE, 1993. 47-53.

27. Foxlin, Eric. "Motion tracking requirements and technologies." Handbook of virtual environment technology 8 (2002): 163-210.

28. Galyean, Tinsley A. "Guided navigation of virtual environments." In Proceedings of the 1995 symposium on Interactive 3D graphics, pp. 103-ff. ACM, 1995.

29. Höllerer, Tobias, Steven Feiner, Tachio Terauchi, Gus Rashid, and Drexel Hallaway. "Exploring MARS: developing indoor and outdoor user interfaces to a mobile augmented reality system." Computers & Graphics 23, no. 6 (1999): 779-785.

30. Igarashi, Takeo, Rieko Kadobayashi, Kenji Mase, and Hidehiko Tanaka. "Path drawing for 3D walkthrough." In Proceedings of the 11th annual ACM symposium on User interface software and technology, pp. 173-174. ACM, 1998.

31. Iwata, Hiroo, Hiroaki Yano, and Fumitaka Nakaizumi. "Gait master: A versatile locomotion interface for uneven virtual terrain." In Virtual Reality, 2001. Proceedings. IEEE, pp. 131-137. IEEE, 2001.

32. Iwata, Hiroo. "Walking about virtual environments on an infinite floor." In Virtual Reality, 1999. Proceedings., IEEE, pp. 286-293. IEEE, 1999.

33. Iwata, Hiroo, and Takashi Fujii. "Virtual perambulator: a novel interface device for locomotion in virtual environment." In Virtual Reality Annual International Symposium, 1996., Proceedings of the IEEE 1996, pp. 60-65. IEEE, 1996.

34. Jeong, Dong Hyun, Chang G. Song, Remco Chang, and Larry Hodges. "User experimentation: an evaluation of velocity control techniques in immersive virtual environments." Virtual reality 13, no. 1 (2009): 41-50.

35. Kessler, G. Drew, Doug A. Bowman, and Larry F. Hodges. "The simple virtual environment library: an extensible framework for building VE applications." Presence: Teleoperators and virtual environments 9, no. 2 (2000): 187-208.

36. Koller, David R., Mark R. Mine, and Scott E. Hudson. "Head-tracked orbital viewing: an interaction technique for immersive virtual environments." In Proceedings of the 9th annual ACM symposium on User interface software and technology, pp. 81-82. ACM, 1996.

37. Kopper, Regis, Tao Ni, Doug A. Bowman, and Marcio Pinho. "Design and evaluation of navigation techniques for multiscale virtual environments." In Virtual Reality Conference, 2006, pp. 175-182. IEEE, 2006.

38. LaViola Jr, Joseph J., Daniel Acevedo Feliz, Daniel F. Keefe, and Robert C. Zeleznik. "Hands-free multi-scale navigation in virtual environments." In Proceedings of the 2001 symposium on Interactive 3D graphics, pp. 9-15. ACM, 2001.

39. Macedonia, Michael. "Innovative computing powers theme park adventures." Computer 2 (2001): 115-117.

40. MacKenzie, I. Scott. "Input devices and interaction techniques for advanced computing." Virtual environments and advanced interface design (1995): 437-470.

41. Mackinlay, Jock D, Stuart K Card and George G Robertson. "Rapid controlled movement through a virtual 3D workspace." *ACM SIGGRAPH Computer Graphics*. 1990. 171-176.

42. Mapes, Daniel P., and J. Michael Moshell. "A two-handed interface for object manipulation in virtual environments." Presence: Teleoperators & Virtual Environments 4, no. 4 (1995): 403-416.

43. Marchal, Damien, Clément Moerman, Géry Casiez, and Nicolas Roussel. "Designing intuitive multi-touch 3d navigation techniques." In Human-Computer Interaction–INTERACT 2013, pp. 19-36. Springer Berlin Heidelberg, 2013.

44. Marchal, Maud, Julien Pettr, and Anatole Lécuyer. "Joyman: A human-scale joystick for navigating in virtual worlds." In 3D User Interfaces (3DUI), 2011 IEEE Symposium on, pp. 19-26. IEEE, 2011.

45. Matthies, Denys JC, Felix M. Manke, Franz Müller, Charalampia Makri, Christoph Anthes, and Dieter Kranzlmüller. "VR-Stepper: A Do-It-Yourself Game Interface for Locomotion in Virtual Environments." arXiv preprint arXiv: 1407.3948 (2014).

46. McCrae, James, Igor Mordatch, Michael Glueck, and Azam Khan. "Multiscale 3D navigation." In *Proceedings of the 2009 symposium on Interactive 3D graphics and games*, pp. 7-14. ACM, 2009.

47. Mercurio, Philip J., Thomas Erickson, D. Diaper, D. Gilmore, G. Cockton, and B. Shackel. "Interactive scientific visualization: An assessment of a virtual reality system." In INTERACT, pp. 741-745. 1990.

48. Mine, Mark R. "ISAAC: a meta-CAD system for virtual environments." Computer-Aided Design 29.8 (1997): 547-553.

49. Mine, Mark. "Working in a virtual world: Interaction techniques used in the Chapel Hill immersive modeling program." University of North Carolina (1996).

50. Mine, Mark. "Virtual environment interaction techniques." *UNC Chapel Hill computer science technical report TR95-018* (1995): 507248-2.

51. Mine, Mark R., Frederick P. Brooks Jr, and Carlo H. Sequin. "Moving objects in space: exploiting proprioception in virtual-environment interaction." Proceedings of the 24th annual conference on Computer graphics and interactive techniques. ACM Press/Addison-Wesley Publishing Co., 1997.

52. Moerman, Clément, Damien Marchal, and Laurent Grisoni. "Drag'n Go: Simple and fast navigation in virtual environment." In 3D User Interfaces (3DUI), 2012 IEEE Symposium on, pp. 15-18. IEEE, 2012.

53. Noma, Haruo, and Tatsuro Miyasato (1998). Design for Locomotion Interface in a Large Scale Virtual Environment—ATLAS: ATR Locomotion Interface for Active Self Motion. *ASME-DSC* 64: 111–118.

54. Pausch, Randy, Jon Snoddy, Robert Taylor, Scott Watson, and Eric Haseltine. "Disney's Aladdin: first steps toward storytelling in virtual reality." In Proceedings of the 23rd annual conference on Computer graphics and interactive techniques, pp. 193-203. ACM, 1996.

55. Pausch, Randy, Tommy Burnette, Dan Brockway, and Michael E. Weiblen. "Navigation and locomotion in virtual worlds via flight into hand-held miniatures." In

Proceedings of the 22nd annual conference on Computer graphics and interactive techniques, pp. 399-400. ACM, 1995.

56. Phillips, Cary B., Norman I. Badler, and John Granieri. "Automatic viewing control for 3D direct manipulation." In Proceedings of the 1992 symposium on Interactive 3D graphics, pp. 71-74. ACM, 1992.
57. Pierce, Jeffrey S., Andrew S. Forsberg, Matthew J. Conway, Seung Hong, Robert C. Zeleznik, and Mark R. Mine. "Image plane interaction techniques in 3D immersive environments." In Proceedings of the 1997 symposium on Interactive 3D graphics, pp. 39-ff. ACM, 1997.
58. Poupyrev, Ivan, 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, pp. 79-80. ACM, 1996.
59. Razzaque, Sharif, David Swapp, Mel Slater, Mary C. Whitton, and Anthony Steed. "Redirected walking in place." In ACM International Conference Proceeding Series, vol. 23, pp. 123-130. 2002.
60. Robinett, Warren, and Richard Holloway. "Implementation of flying, scaling and grabbing in virtual worlds." Proceedings of the 1992 symposium on Interactive 3D graphics. ACM, 1992.

61. Ruddle, Roy A., Stephen J. Payne, and Dylan M. Jones. "Navigating buildings in" desk-top" virtual environments: Experimental investigations using extended navigational experience." Journal of Experimental Psychology: Applied 3, no. 2 (1997): 143.

62. Schell, J., and Joe Shochet. "Designing interactive theme park rides." Computer Graphics and Applications, IEEE 21.4 (2001): 11-13.

63. Shoemake, Ken. "ARCBALL: a user interface for specifying three-dimensional orientation using a mouse." In Graphics Interface, vol. 92, pp. 151-156. 1992.

64. Slater, Mel, Martin Usoh, and Anthony Steed. "Taking steps: the influence of a walking technique on presence in virtual reality." ACM Transactions on Computer-Human Interaction (TOCHI) 2, no. 3 (1995): 201-219.

65. Song, Deyang, and Michael Norman. "Nonlinear interactive motion control techniques for virtual space navigation." *Virtual Reality Annual International Symposium, 1993., 1993 IEEE*. IEEE, 1993.

66. Stoakley, Richard, Matthew J. Conway, and Randy Pausch. "Virtual reality on a WIM: interactive worlds in miniature." Proceedings of the SIGCHI conference on Human factors in computing systems. ACM Press/Addison-Wesley Publishing Co., 1995.

67. Stuerzlinger, Wolfgang, and Chadwick A. Wingrave. The value of constraints for 3D user interfaces. Springer Vienna, 2011.

68. Tan, Desney S., George G. Robertson, and Mary Czerwinski. "Exploring 3D navigation: combining speed-coupled flying with orbiting." In Proceedings of the

SIGCHI conference on Human factors in computing systems, pp. 418-425. ACM, 2001.

69. Teller, Seth J., and Carlo H. Séquin. "Visibility preprocessing for interactive walkthroughs." ACM SIGGRAPH Computer Graphics. Vol. 25. No. 4. ACM, 1991.
70. Templeman, James N., Patricia S. Denbrook, and Linda E. Sibert. "Virtual locomotion: Walking in place through virtual environments." Presence: teleoperators and virtual environments 8, no. 6 (1999): 598-617.
71. Trapp, Matthias, Lars Schneider, Norman Holz, and Jürgen Döllner. "Strategies for visualizing points-of-interest of 3D virtual environments on mobile devices." In 6th International Symposium on LBS and TeleCartography, Springer, Berlin, Heidelberg. 2009.
72. Trindade, Daniel R and Alberto B Raposo. "Improving 3D navigation in multiscale environments using cubemap-based techniques." *Proceedings of the 2011 ACM Symposium on Applied Computing*. ACM, 2011. 1215-1221.
73. Usoh, M., Arthur, K., Whitton, M.C., Bastos, R., Steed, A., Slater, M. and Brooks Jr, F.P., 1999, July. Walking> walking-in-place> flying, in virtual environments. In Proceedings of the 26th annual conference on Computer graphics and interactive techniques (pp. 359-364). ACM Press/Addison-Wesley Publishing Co..
74. Von Kapri, Anette, Tobias Rick, and Steven Feiner. "Comparing steering-based travel techniques for search tasks in a cave." In Virtual Reality Conference (VR), 2011 IEEE, pp. 91-94. IEEE, 2011.

75. Ware, Colin and Daniel Fleet. "Context sensitive flying interface." *Proceedings of the 1997 symposium on Interactive 3D graphics*. ACM, 1997. 127-130.

76. Ware, Colin, and Kathy Lowther. "Selection using a one-eyed cursor in a fish tank VR environment." ACM Transactions on Computer-Human Interaction (TOCHI) 4, no. 4 (1997): 309-322.

77. Ware, Colin, and Steven Osborne. "Exploration and virtual camera control in virtual three dimensional environments." In ACM SIGGRAPH Computer Graphics, vol. 24, no. 2, pp. 175-183. ACM, 1990.

78. Welch, Greg, Gary Bishop, Leandra Vicci, Stephen Brumback, Kurtis Keller, and D'nardo Colucci. "High-performance wide-area optical tracking: The hiball tracking system." presence: teleoperators and virtual environments 10, no. 1 (2001): 1-21.

79. Welch, Greg, Gary Bishop, Leandra Vicci, Stephen Brumback, and Kurtis Keller. "The HiBall tracker: High-performance wide-area tracking for virtual and augmented environments." In Proceedings of the ACM symposium on Virtual reality software and technology, pp. 1-ff. ACM, 1999.

80. Wells, M., B. Peterson, and Jason Aten. "The virtual motion controller: A sufficient-motion walking simulator." In Proceedings of VRAIS, vol. 97, pp. 1-8. 1996.

81. Wobbrock, Jacob O., Leah Findlater, Darren Gergle, and James J. Higgins. "The aligned rank transform for nonparametric factorial analyses using only anova procedures." In Proceedings of the SIGCHI Conference on Human Factors in Computing Systems, pp. 143-146. ACM, 2011.

82. Zeleznik, Robert C., Joseph J. LaViola Jr, Daniel Acevedo Feliz, and Daniel F. Keefe. "Pop through button devices for VE navigation and interaction." In Virtual Reality, 2002. Proceedings. IEEE, pp. 127-134. IEEE, 2002.

83. Zhai, Shumin. "User performance in relation to 3D input device design." ACM Siggraph Computer Graphics 32, no. 4 (1998): 50-54.

Appendix A: Levene’s Test

Levene’s test is a statistical test that determines the equality of variances for a variable calculated for two or more conditions. It is used in this work to check if the assumption of homogeneity of the variances holds before an Analysis of variance (ANOVA) test is performed.

Levene’s test checks for the null hypothesis that the population variances are equal. In my research, I rejected this null hypothesis when the *p*-value was < .05. When the null hypothesis is rejected, it indicates that one of the assumptions of the ANOVA test is violated and therefore it is unsuitable to use the ANOVA test.

Appendix B: Aligned Rank Transform

The Aligned rank transform (ART) is a data transformation that is useful when data violates the ANOVA test assumptions. It transforms data into a form that is suitable for an ANOVA test by simply ranking the data. If this transform is not performed, the results of the ANOVA would have an inflated risk of Type I error rates (an incorrect rejection of the null hypothesis).

This transform provides accurate nonparametric treatment for both main and interaction effects and is useful in the analysis of the data that I collected from the performed experiments. The performed transform has the pre-processing step of aligning the data for each effect before assigning ranks that makes nonparametric treatment of interaction effects possible. This is one of the innovations in the ART method over previous work [76].

There are five steps to this procedure. Step one is computing the residuals. Step two is computing the estimated effects for all main and interaction effects. Step three is computing the aligned response (Y'). Step four is assigning averaged ranks (Y"). Lastly, step five is preforming the ANOVA on the computed Y". These steps are described in greater detail in the work by Wobbrock et al. [81].