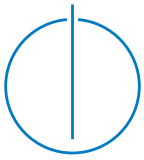
DEPARTMENT OF INFORMATICS

TECHNISCHE UNIVERSITÄT MÜNCHEN

Master's Thesis in Informatics: Games Engineering

# Exploration and Classification of State-of-the-art Visible 3D User Interfaces in Virtual and Augmented Reality Scenarios

Moritz Schwab



DEPARTMENT OF INFORMATICS

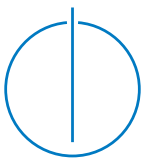
TECHNISCHE UNIVERSITÄT MÜNCHEN

Master's Thesis in Informatics: Games Engineering

# Exploration and Classification of State-of-the-art Visible 3D User Interfaces in Virtual and Augmented Reality Scenarios

# Exploration und Klassifizierung Aktueller Sichtbarer 3D Benutzeroberflächen für Anwendungen im Bereich der Virtuellen und Erweiterten Realität

| | |
|---|---|
| Author: | Moritz Schwab |
| Supervisor: | Prof. Gudrun Klinker, Ph.D. |
| Advisor: | Sven Liedtke, M.Sc. |
| Submission Date: | 15.07.2018 |



I confirm that this master's thesis in informatics: games engineering is my own work and I have documented all sources and material used.

Munich, 15.07.2018 Moritz Schwab

# Abstract

The fields of Augmented and Virtual Reality have seen a flurry of development in recent years, both in entertainment and more serious areas of research such as medical, educational or commercial applications. The technology has matured to a point where even many end-users own a Head-Mounted Device and frequently use it for both AR and VR software, while web cams make AR applications available to anyone with access to a desktop computer. While hardware and software seem to have entered a stable phase of only incremental improvements, the new, three-dimensional user interfaces needed to interact with fully three-dimensional applications are still immature compared to the decades of research and development that have lead to modern two-dimensional user interfaces.

Tasks such as navigating a three-dimensional space, working with a three-dimensional model or even just interacting with a list of data points are often hampered by a rudimentary, unergonomic 3D interface or a two-dimensional interface projected into three dimensions. While many attempts have been made to create new and better interfaces, these are mostly independent efforts with little connection to each other and often not documented publicly if at all. This thesis aims to begin to rectify this situation by documenting the current state of the art in visible 3D User Interfaces. The documentation will consist of a collection and classification of existing interface solutions and the tasks they were created to achieve, and an evaluation of how well-suited they are to achieving their specific task.

# Contents

# 1 Introduction

Augmented and Virtual Reality have become a popular topic of research and speculation in the last few years, as the technological possibilities finally catch up with the grand ideas researchers and futurologists had in the last decades. The success of games like *Pokémon GO*, which was downloaded over 100 million times just on Android devices [Dog17], proves that there is an enormous opportunity for smartphone Augmented Reality applications to flourish on the entertainment market, while museums, schools and university are beginning to use Augmented Reality for the purpose of education. While these smartphone applications are popular and often easy to use, their user interfaces mostly consist of traditional two-dimensional menus and buttons, which limit the potential of these applications.

Almost all applications, not just smartphone apps, are defined primarily through their user interface. They are usually the first thing a user experiences and interacts with and their quality greatly influences the quality of the entire interaction with an application. Many decades of research have gone into optimizing user interfaces for traditional screen display, settling on a Window-Mouse-and-Pointer (WIMP) interface for desktop computers and simplified menus and buttons, enhanced by gestures, for mobile applications. In the context of Augmented and Virtual Reality applications however, these two-dimensional user interfaces are often limiting both the interaction with the application, which has become three-dimensional, but also the methods and types of data that can be displayed.

As more and more companies develop Augmented and Virtual Reality headsets and similar head-mounted devices there is a growing demand for good three-dimensional interface solutions. The following work aims to classify the different tasks of a three-dimensional user interface and document existing solutions that aim to fulfill these tasks and evaluate their effectiveness in doing so. Due to the broadness of the topic this work will focus only on one aspect of 3D user interfaces: the graphical aspects, such as visualization of data, such as 3D models, data produced by the application or entered by the user, status updates and similar. This also includes visual aspects of the user's interaction such as markers aiding in selection, which may be arrows, halos, or distortions, travel markers, wayfinding aids and similar representations of possible or current interactions.

This section introduces the core concepts of Augmented and Virtual Reality and gives

a brief overview of the history and types of user interfaces.

## 1.1 Augmented Reality

While many see Augmented Reality as a subset of Virtual Reality, or the same field, or declare them both to be entirely separate fields that share many attributes, one of the most common definitions is found in *Milgram et al.*'s work on Augmented Reality [Mil+95] in which they propose the existence of a Reality-Virtuality Continuum (See figure 1.1). In this continuum reality forms one endpoint with complete virtuality, including many forms of Virtual Reality, forming the other. Augmented Reality falls into the space between the two ends, which is called Mixed Reality, though it is placed closer to reality on the continuum to emphasize that it is used only to augment or enhance reality instead of replacing it.

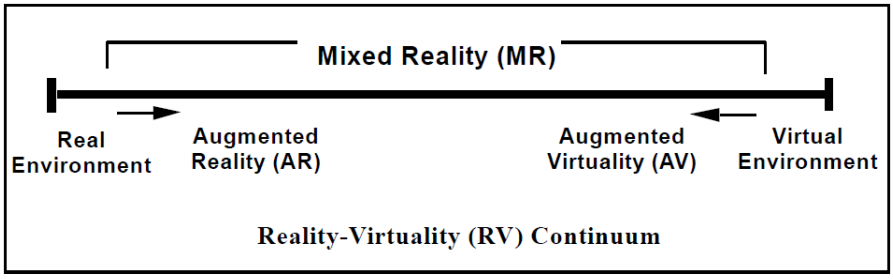



Figure 1.1: *Milgram et al.*'s Reality-Virtuality Continuum [Mil+95]

*Milgram et al.*'s conclusions work well in differentiating Augmented and virtual reality but do not clearly define where Augmented Reality begins and ends on the continuum and its differences from a simple overlay. In his work *A Survey of Augmented Reality* [Azu97] *R. Azuma*, suggests an alternative definition through three defining characteristics an application must show to be considered Augmented Reality:

1. It combines real and virtual
2. It is interactive in real time
3. It is registered in 3-D

These characteristics were intended by *Azuma* to include the Augmented Reality applications found in his survey of the then-current state-of-the-art as well as any possible future applications while also excluding borderline scenarios such as interactive two-dimensional overlays on live video. The broadness of this definition is likely due to the wide spectrum of applications included in *Azuma*'s survey which also serve

to showcase the potential for Augmented Reality in general. One such application focused on in the survey is medical visualization such as rendering three-dimensional data from ultrasounds and computer tomography scans and overlaying it on a patients body, replacing the need for a separate manual in surgery or guiding surgeons on where to begin their surgery. Overlaying important information or instruction manuals over objects is also useful in other fields such as maintenance of complex machines, replacing the head-up display in planes or simply annotating various objects such as library books or museum exhibits with additional information.

## 1.2 Virtual Reality

Figure 1.2: The chaperone system of the HTC Vive showing a boundary and outlines of real objects [Bro17]

The popularity surrounding the arrival of affordable Virtual Reality setups on the consumer market ensures that today even laymen will have an understanding of what the term Virtual Reality stands for: immersive Virtual environments created by head-mounted devices (HMDs), rooms setup with screens on all walls or similar setups that fully replace the real world with a virtual one. The history of Virtual Reality however starts with a different term. *Artificial Reality*, a term coined by *Myron Krueger*, was already used to describe Virtual Reality in the 1970s and later documented in the year 1983 in his book *Artificial Reality* [Kru83]. In the second edition of his book [Kru91], he defines Artificial Reality as something that perceives a users actions in relation

to a graphic world and reacts in a way that makes the user believe these actions are taking place in that world. Virtual Reality's first documented appearance as a term for immersive virtual environments was published in 1982, in the science-fiction novel *The Judas Mandala* [Bro82], though it was likely inspired by the use of the term *Virtuality* in the article *Interactive Systems and the Design of Virtuality* by *Theodor Nelson* [Nel80] which was published two years before.

*Milgram et al.*'s Reality-Virtuality Continuum (see figure 1.1) places Virtual Reality at the far-right end of the continuum, outside the area defined as Mixed Reality. Modern roomscale setups, such as the *HTC Vive* often fall into the spectrum of Mixed Reality, as their inclusion of outlines of real-world objects such as furniture to avoid collisions [Gep17] (see figure 1.2) would make them Augmented Virtuality applications which are still distinct from Augmented Reality as defined by *Azuma* and are commonly considered a sub-field of Virtual Reality. This is due to the outlines still being virtual and thus not fulfilling the first of the three characteristics of Augmented Reality.

Another way to define Virtual Reality is found in *Understanding Virtual Reality* [SC03] by *Sherman et al.*, a book intended as a foundational work on the topic, providing a high level overview over all components of a Virtual Reality system. They identify four key elements that make up Virtual Reality:

1. A virtual world
2. Immersion
3. Sensory feedback
4. Interactivity

A virtual world is, according to the authors, simply any imaginary space, which may be manifested through a medium or a description of a collection of objects and the rules governing and relationships between those objects. Immersion is defined as a sense of presence, or being mentally immersed into a virtual world, while sensory feedback may describe any kind of feedback to the users actions, such as head tracking allowing the user to look around giving visual feedback, or the resistance a haptic interface uses to simulate an object giving haptic feedback. Finally, the virtual world must respond to user actions, making it interactive. From these elements they derive a definition of Virtual Reality as a medium composed of interactive computer simulations, which are capable of sensing the users actions and/or position as well as augmenting or replacing the feedback to one or more senses to immerse the user in a virtual world.

## 1.3 3D User Interfaces

The book *3D User Interfaces: Theory and Practice* [LaV+17] by *LaViola et al.* defines 3D user interfaces through the definitions of the components of a classical UI: A UI translates a user's inputs into a computer-interpretable representation, as well as translating the computer's outputs into a human-interpretable representation. The user interacts with the computer using a set of interaction techniques which are defined as a method that allows a user to accomplish a task. They include both hardware and software components and input and output. These definitions are then expanded to include 3D and result in a definition for 3D interaction: Human-Computer Interaction (HCI) in which the user acts in a 3D spatial context. This includes both 3D input devices and 2D input devices such as mouses that can be mapped to provide 3D dimensional input. A 3D user interface is then simply defined as a UI that involves 3D interaction.

A historical overview on how 3D user interfaces came to develop, such as the one found in *LaViola et al.*'s book, may provide further insight as to the differences between 2D and 3D interfaces Before the 1980s HCI was mostly limited to command-line tools. The 1980s however saw the development of several new technologies that would allow user interfaces to become graphical and interactive: affordable personal computers, inexpensive raster graphics displays and the computer mouse. The rise of graphical user interfaces in turn created a new age of HCI research, as making graphical user interfaces (GUIs) more usable affected anyone who was using computers.

The book draws parallels between the technological progress that fueled the advent of GUIs and the progress that now draws researchers to 3D User Interfaces. Advanced 3D graphics, Augmented Reality, Virtual Reality and better 3D tracking technologies not only enabled new applications but also new, previously unseen, ways to interact with them, creating new challenges and usability issues.

One of the first serious takes on the idea of three-dimensional HCI was formulated in 1965 by *Ivan Sutherland* [Sut65]. Only three years later he developed a prototype HMD, capable of Augmented and Virtual Reality which used precise, mechanical means to track the user's head movements and determine the viewing angle and head position, making head tracking the first documented 3D HCI technique. While ahead of his time, *Sutherland* did lay the foundation for a new wave of Virtual Reality applications in the late 1980s and early 1990s, inspired by new technological progress at the time. According to *LaViola et al.* this progress consisted of three-dimensional stereoscopic computer graphics, miniaturized CRT displays, new position tracking systems and new interaction devices such as datagloves. They postulate that although most prototypes used only head-tracking at the time, input devices like the data glove may have shaped public perception of Virtual Reality more so than any other new technology.

Unlike the highly interdisciplinary field of 2D HCI, 3D interaction seemed to be

dominated by computer scientists and software engineers. Most applications used simple techniques to realize concepts such as scientific visualization, virtual architectural walkthroughs and Virtual Reality games. While the challenges posed by these applications greatly advanced several areas of research such as latency reduction, graphics and tracking, the simple nature of the interaction, mostly limited to navigating the environment, provided few opportunities to further improve 3D HCI. As technology advanced further, applications slowly grew more complex and the range of possible interactions broadened. *LaViola et al.* offer the example of an architectural walkthrough being expanded with further interactions, such as the ability to record audio annotations linked to certain designs, change materials, manipulate details of the building or hide certain features to allow for different views. While all these features were easily implementable from a technological standpoint, there was no established way to design these interactions in a way that would make them intuitively usable without further research into 3D HCI.

Fortunately many decades of HCI research were just as applicable to three-dimensional space as they were to two-dimensional space, such as research into how humans process information, design guidelines and principles for the design and development process. In many areas however, traditional HCI principles failed to capture the entirety of the task at hand. *LaViola et al.* suggest that in the fairly simple case of an application visualizing three-dimensional medical data, using a 3D input device, traditional HCI guidelines only specify that direct manipulation (represented in the example by the dataset rotating if the input device is rotated) is an efficient and intuitive solution. However, these guidelines do not offer a solution for the variety of challenges and problems arising with this approach, affecting software and hardware design. Some examples include how to scale the rotation so as not to tire the user out, how to design the interaction device so no wires hinder the interaction but also graphical aspects of the design such as how the user selects a particular location to annotate, how such a selection might be highlighted and other similar challenges.

Additionally, many 3D User Interfaces still need to be designed around technological limitations such as cumbersome devices tiring users out if an interface encourages repetitive straining actions, the size of 3D workspace limiting the UI or the UI having to work around dropouts or latency in tracking. This is compounded by the large design space offered by 3D interactions, as the many different varieties of input devices and the broad range of possible interpretations of input create infinite possibilities for 3D UIs, which makes choosing the "right" interaction, or even a highly usable one a daunting task. Many of these possible interaction methods are highly task-specific, with a gesture controlled Augmented Reality UI not translating well to a controller-based Virtual Reality application. To address these tasks this and other works catalog and evaluate existing and possible solutions to a variety of tasks.

# 2 Related Work

When exploring new territory in any kind of research it is important to be aware of what others have achieved in the same field. This section lists and analyzes important works from the fields of Augmented and Virtual Reality as well as 3D user interfaces, focusing on those that provide high-level overviews of the field, explore the current state-of-the-art or feature classifications of those fields.

## 2.1 Important Works On Augmented And Virtual Reality

*Augmented Reality: Principles and Practice* by *Schmalstieg et al.* [SH16] was written, as the title suggests, both as an educational textbook on basic principles, to be used for self-study or university courses, or as a guidebook and reference for developers working on practical Augmented Reality applications. It also serves as an overview of the current state-of-the-art of Augmented Reality and related technologies. After an introduction to the definitions and history of the field, its second and third chapter discuss the hardware needed for Augmented Reality: displays and other sensory feedback methods as well as tracking methods and sensors. The next three chapters deal with the software aspects of Augmented Reality. The chapter on computer vision contains algorithms and solutions for problems related to visual tracking, using case studies to explain them as well as discussing which of these are implemented in commonly used libraries and which are usually re-implemented. The chapter on calibration and registration explains how the cameras used for visual input are calibrated and how the physical and digital parts of the image are aligned. Visual coherence focuses on techniques that seamlessly blend the real and virtual world, such as occlusion or shadowing between real and virtual objects. The next five chapters directly deal with the subject of this work, user interfaces, data visualization and interaction methods. In the seventh chapter the authors discuss different methods of situated visualization, both 2D and 3D and how to design them in an easily understandable way. This chapter is followed by a chapter on interaction, which discusses different interaction methods and both the input and output aspects of a variety of user interfaces. The chapter on modeling and annotation describes in detail the challenges and opportunities, both in design and implementation of using Augmented Reality to add new geometry and annotations to the existing world. Discussing the topic of authoring content for Augmented Reality

applications, the authors suggest the existence of a need for an authoring system decoupled from the implementation itself and propose a system requiring minimal programming knowledge and recent advances towards such systems. In the last of the chapters directly relating to this work, *Schmalstieg et al.* discuss navigation in Augmented Reality, such as using Augmented Reality as a navigational tool, but also techniques paralleling *Laviola et al.*'s definition of wayfinding [LaV+17] such as guiding the users view to certain objects or key features in the environment. The final three chapters then discuss the possibilities and challenges of using Augmented Reality for both local and remote collaboration, discuss the software architecture and architectural patterns best suited to an Augmented Reality implementation, with the final chapter debating possible options for Augmented Reality research in the future, examining possible obstacles of these options and predicting future trends for the whole field.

Paralleling the famous survey of the field by *R, Azuma* [Azu97] a more recent review of the state-of-the-art of Augmented Reality was published in 2015 under the same name by *Billinghurst et al.* [BCL15]. They built upon previous surveys to create a more comprehensive overview of the field. Starting out with explaining their definition and taxonomy of the field, as well as a brief recounting of its history, the survey is divided into twelve chapters, with the first few chapters covering the technology used in tracking, displays, input and interaction as well as the tools used in the development of Augmented Reality applications. The authors then discuss guidelines and patterns for designing such applications and cover their evaluation. Finally, they present an overview of current applications, divided into three categories, education, marketing and architecture, and predict future research directions.

A fundamental introduction to Virtual Reality and its development, as well as a collection of relevant materials and references is found in the book *Understanding Virtual Reality* [SC03] by *Sherman et al.*. It is divided into nine chapters, with the first chapter serving as an introduction to Virtual reality, containing the authors' definition of VR and a history of the field. The second chapter discusses the implications of VR's characteristics as a medium on content presentation, authoring and interaction. In the fourth and fifth chapter the authors examine the influence input and display methods and their usability on the quality of the user experience. The next chapter, titled *Rendering the Virtual World*, discusses issues related to human perception in VR, such as how to visually arrange or display content, but also touches on how to design user interfaces in a way that increases intuitiveness. This topic is continued in chapter six, which covers interaction in VR, UI metaphors as well as definitions of and information on different interface tasks in VR. Chapter seven provides guidelines on authoring and implementing content for VR that increases immersion in the virtual world and how to avoid common pitfalls. The foundations from earlier chapters are then built upon in chapter eight, which presents best practices for the design of a VR experience. The final

chapter is used to document the then-current state of the field along with the authors' predictions for the future of the field.

## 2.2 Important Works On User Interfaces

Human-Computer Interaction has been an active area of research for many decades now, while 3D HCI has had much less time to develop as a field. Still, many guidelines, principles and aspects of human psychology apply just as well to three-dimensional space as they do to two-dimensional space.

### 2.2.1 2D User Interfaces

While this work will focus on observational methods in researching user interfaces (see section 3.2), the experimental perspective is extremely important in the development of UIs. Recognizing the need for an introductory work on the subject, *I. S. MacKenzie* created *Human- Computer Interaction: An Empirical Research Perspective* [Mac13] which serves as an introductory guide to and practical manual for empirical research in HCI. Divided into eight chapters, the book begins with an accounting of the history of the field of HCI while the second chapter introduces human cognitive, sensory and motor capabilities. The next chapter covers the core elements of human-computer interaction and action-response pairings. Chapter four introduces the scientific foundations of HCI research, providing a guide on how to craft quantifiable hypotheses, chapter five then discusses how these hypotheses may be tested in experiments and user studies. The math needed to interpret the statistical data gained from these experiments and to evaluate the research hypotheses based on that data is the subject of chapter six. In chapter seven the author presents two common approaches to modeling in HCI, descriptive modeling and predictive modeling. The book ends with a guide on combining its lessons into and publishing an HCI research paper.

To understand what separates unusable from usable UIs and how certain mistakes are made it is important to know the principles of the UI design process. In *Designing for Usability: Key Principles and What Designers Think* by *Gould et al.* [GL85] the authors describe design principles they believe must be followed and a case study as proof of their effectiveness. The three principles are as follows: (1) Early focus on users and tasks, meaning that the UI needs to be designed with typical tasks and user behavior in mind. Developers should be in contact with users and understand their needs before starting system design. (2) There should be empirical measurements of user performance early in the process. (3) Feedback from user testing must be incorporated into the design which results in iterative development of the UI. The included case study documents the development of the IBM Audio Distribution System, a phone-based messaging system

intended to be used by an audience not familiar with computers. After a prototype revealed that different features than those initially predicted to be important would need to be emphasized, the authors found that user contact is necessary even before the initial design. Still, feedback from early user tests led to sweeping changes in design which would have been impossible later in the process. Additionally, user testing also revealed several improvements which quantifiably increased user performance but contradicted design guidelines. The authors conclude that the system was greatly improved through the use of their guidelines.

### 2.2.2 3D User Interfaces

One book about 3D user interfaces that is held in especially high regard is *3D User Interfaces: Theory and Practice* by *Laviola et al.* [LaV+05]. Originally adapted from a series of talks given by the authors it has remained one of the most renowned collections of knowledge on the subject and has served as the basis for many university courses and talks about the subject since its publication. With new technologies further advancing the field the authors have recently released an updated second edition [LaV+17] to address the changed state of 3D HCI research. The book is divided into six major parts, covering the foundations of 3D user interfaces, the basics of Human-Computer Interaction and Human Factors, the hardware and technology required for 3D user interfaces, possible 3D interaction techniques, the design and development of 3D user interfaces and the future of 3D User Interfaces. In the first part they introduce a definition of 3D User Interfaces by first defining 3D interaction techniques and classifying user interfaces that make use of them as 3D user interfaces, while defining their study as the research of 3D HCI. They then outline the motivation of their research and briefly introduce the history and applications of 3D user interfaces segueing into the chapter on Human Computer Interaction which in turn focuses on all areas where user interfaces work around human capabilities and properties. These include how humans process information, perceptive and cognitive abilities and ergonomic limitations. It also includes guidelines on how to understand, design and engineer for an improved user experience. The chapter on hardware includes an overview of possible visual, auditory and haptic output devices and a variety of traditional, spatial, complementary and special input devices as well as their advantages and disadvantages. The 3D Interaction techniques the book presents are grouped by task, with one chapter each discussing the tasks of selection and manipulation, travel, wayfinding, and system control. The last major section of the book provides guidelines and strategies useful in the development and design of 3D UIs, as well as the methods and metrics with which they are evaluated.

In contrast to *LaViola et al.*'s focus on a high-level overview of available technologies

and best practices, guidelines and general techniques, the book *Interaction Design for 3D User Interfaces* by *Ortega et al.* focuses on the practical aspect of developing hardware and software for 3D user interfaces, especially providing as much information as possible on input methods and technologies. The book is split into four parts, focusing in turn on theory, advanced topics, hands-on projects and a case study using speech as input. Part one starts out with a brief introduction to the field in general and an overview of input and output technologies and devices. The authors then briefly touch upon general principles of 3D user interfaces, which they split into 3D interaction, a topic which includes tasks such as selection and manipulation, and 3D navigation, which covers tasks such as travel and wayfinding. The authors then introduce input models and focus on the technologies, math and theoretical foundations needed to develop for input devices using multi-touch or brain-computer interfaces. The section on advanced topics included in the book begins with an introduction to the topics of 3D math for input devices, digital signal processing, 3D rotations and various sensor types and the math needed to use their data. This knowledge is then applied in the hands-on section to several projects using new technologies such as inertial sensors for input, or developing for the Oculus Rift, Leap Motion or Kinect. The book finishes with a case study on using speech as an input in the context of conversational interfaces.

# 3 Evaluation of 3D User Interfaces

While cataloging and classifying User Interfaces or aspects thereof provides a useful reference for others to use in and of itself, evaluating one's findings in respect to their usability or ability to fulfill the tasks they were designed to fulfill, provides additional, very important, context. According to *Improving a Human-Computer Dialogue* [MN90], a good user interface should adhere to nine simple guidelines and almost all problems a user interface might exhibit are violating one of these guidelines. These guidelines proclaim that a UI, here called a dialogue, split into smaller dialogues, should:

**Have simple and natural dialogue:** It should not contain irrelevant or rarely used information, as irrelevant information will compete for visibility with information that is actually relevant.

**Speak the user's language:** Words, phrases, concepts and icons used in the UI should be familiar to the user instead of using system-specific terminology.

**Minimize the user's memory load:** The user should not have to remember information between different parts of the UI. Complex interactions should be simplified and relevant information easily retrievable or visible.

**Be consistent:** A user should not be confused by usage of similar terminology, imagery or actions that lead to similar results, instead one particular user action should lead to one system action and vice-versa.

**Provide feedback:** The UI should provide the user with appropriate feedback on success, failure or progress of the execution of the user's commands.

**Provide clearly marked exits:** Exiting a system dialogue or canceling a multi-step action should be possible through either a clearly marked or intuitive "emergency exit" action.

**Provide shortcuts:** While clearly marked actions, verbose dialogue or extended subdialogues make a UI easier to learn, they may slow down experienced users. The system should provide subtle shortcuts to bypass such aids, allowing experienced users to work more efficiently.

**Provide good error messages:** Error messages should be defensive, precise and constructive. Defensive error messages blame system deficiencies instead of the user, precise messages provide the exact problem, constructive error messages provide suggestions on how to avoid the problem.

**Have good error prevention:** Errors can and should be prevented by good design.

This section provides an overview of methods created to evaluate user interfaces, originally created for two-dimensional user interfaces, which can be applied to 3D user interfaces. Additionally, it will introduce the methods used to evaluate the user interfaces cataloged in chapter 5.

## 3.1 Methods To Evaluate User Interfaces

*LaViola et al.* [LaV+17] provide an overview of different methods used in the evaluation of two-dimensional user interfaces that have been used for three-dimensional ones as well.

### 3.1.1 Heuristic Evaluation

Heuristic evaluation, also known as guidelines-based expert evaluation was proposed by *Nielsen et al.* in *Heuristic Evaluation of User Interfaces* [NM90]. This method consists of several usability experts separately examining a user interface, using general UI design guidelines and specific heuristics and guidelines tailored to the UI at hand. The results of these examinations are then ranked and used to improve the UI and identify usability issues and improve the UI. This method is based on *Nielsen et al.*'s discovery that while a single expert could only discover between 20% and 51% of usability issues in an interface, a group of three experts could find 42% to 81% of issues between them and a group of ten experts even being able to identify 71% to 97% of issues in an interface. From these findings they conclude that when evaluating an interface heuristically, one should not solely rely on one persons feedback, but instead suggest that three to five persons independently examine the interface. They also note that this method often identifies usability problems without suggesting solutions for them and rarely leads to new breakthroughs in the evaluated designs.

### 3.1.2 Cognitive Walkthrough

The cognitive walkthrough was developed by *Polson et al.* [Pol+92] based on cognitive theory, evaluating the suitability of a UI to be discovered through exploratory learning. This method is useful when designing or evaluating systems whose users are unlikely

to receive formal training or spend a lot of time learning the system or if one wants to evaluate the usability for a first-time or infrequent user. For a cognitive walkthrough session, one creates a detailed design description of the user interface, a task scenario, a detailed set of predictions concerning users and the context of use and finally a step-by-step sequence of user actions needed to fulfill the task. Then, the user actions are stepped through, evaluating at each step how the interface may affect the user and which actions might be unintuitive or difficult to choose or perform. *Polson et al.* base their model of user behavior on four steps they assume every user to perform: The user sets a goal to accomplish with the system, then searches the interface for currently available actions and selects the one most likely to progress towards the goal. Finally, the user performs the action and decides whether it made progress towards their goal based on system feedback. As this process greatly depends on how salient the correct next step appears to the user, *Riemann et al.* derive five further heuristics [RFR95] to assess the salience of the correct path: (1) Users will try labeled actions first, before they attempt to directly manipulate unlabeled objects. (2) A well-labeled action is especially salient. (3) If labeling is impossible or no metric for the quality of a label can be established, providing few actions to choose from greatly narrows the search. (4) Set effects may keep the user from trying untypical actions. (5) Users are reluctant to extend their search beyond readily available or frequently used menus and controls.

### 3.1.3 Formative Evaluation

The term formative evaluation (along with summative evaluation, see section 3.1.4) was originally coined by *M. Scriven* in 1967, as a method for evaluating educational curricula [Scr67]. Its use in interface design was documented by *Hix et al.* in their book *Developing User Interfaces: Ensuring Usability Through Product & Process* [HH93]. They note that this evaluation method can be used formally and informally and that it is applied during every step of the design process in order to quickly address usability issues. Interfaces are evaluated by assigning users, which are representative of the final user base, to scenarios based on single tasks. This can lead to qualitative results such as problematic design and user feedback on the interface as well as more quantitative results such as the number of errors being made or the time taken to fulfill a task. These results are then taken into account in further design and redesign work and alternating between evaluation and design leads to a more refined and usable UI.

### 3.1.4 Summative Evaluation

While the term summative evaluation was originally coined by *M. Scriven* as the opposite of formative evaluation, it was also described by *Hix et al.* (also known as

comparative evaluation) as a method to either compare different UI designs, components, or techniques to each other or to measure their usability against target usability values. In summative evaluation users perform task-based scenarios while data (both quantitative and qualitative) is collected by the evaluators. If the task scenarios are consistent between tests these results can then be compared with the results of another design, allowing direct comparisons (on a task-by-task basis) between different interfaces or interface components.

### 3.1.5 Questionnaires

*Hix et al.* define a questionnaire as a written set of questions which are posed to subjects before or after they have used the UI that is being evaluated. They can be used to collect demographic information or qualitative and subjective data such as user feedback, comments and difficulties a user may have faced.

### 3.1.6 Interviews

An interview is defined by *Hix et al.* as gathering information from a user by talking to them directly. Interviews can be both structured, with a predetermined sequence of questions or open-ended with the interviewer reacting to the answers of the user and adding additional questions to further explore new ideas a user may have inspired. Both open-ended and structured interviews are well-suited to collecting subjective feedback from users and are often able to elicit more detailed feedback than a questionnaire. Usually, interviews are not used as the sole method of evaluation, instead supplementing other methods such as formative or summative evaluation [BH97].

## 3.2 Evaluation Methods Used In This Work

To evaluate the interfaces cataloged in this work, an evaluation method had to be chosen. The previous section introduced several possible ways to evaluate 3D user interfaces and their components. However, not all of them are suitable for use in this work.

The fact that the interfaces are already finished designs makes it impossible to evaluate them during their design process, ruling out formative evaluation, which shapes them during the design process. The number of interfaces included in this work, as well as their scale, would require a very large group of users and either a long time or large number of evaluators, if one were to use questionnaires, interviews or even summative evaluation. This leaves the cognitive walkthrough and heuristics-based evaluation as the only suitable choices for evaluation in this work. In addition to the

application of these evaluation methods, any documented evaluation of the interfaces discussed in this work, whether by the original developers of the interface or a third party, will also be included in this work.

# 4 Classification Of User Interfaces

Aspects of or even whole user interfaces are usually defined through the method in which they present data or the interaction technique used. However, Augmented and Virtual Reality offer an uncountably large space of possible presentation methods and interaction techniques and many aspects of user interfaces require discussion of both elements. These facts make it necessary to decompose the space off possible UI functions into groups and subgroups. This chapter will explore prior work on classifying user interfaces in Augmented and Virtual reality and derive a classification system from these works that will be used to catalog the user interfaces and interface elements found in the catalog of UIs (see chapter 5).

## 4.1 Prior Work On Classification Of User Interfaces

### 4.1.1 Classification Through Interaction Tasks

#### Classification According to *3D User Interfaces: Theory and Practice*

Many modern works on 3D user interfaces use *LaViola et al.*'s classification of interaction techniques, (see for example *Ortega et al.*'s work [Ort+16]) which is based on the concept of grouping them by their associated user interaction task [LaV+05][LaV+17]. They define three main tasks intended to cover almost all possible interaction techniques: selection and manipulation, travel, and system control and document different possible classification methods for each of them.

The most fundamental interaction tasks, selection and manipulation, form the base for most application specific tasks and the quality of the techniques used for them strongly affects the user experience of the entire UI. The authors present three methods to further classify selection and manipulation techniques, the first of which is to differentiate between isomorphic and non-isomorphic techniques. Isomorphic techniques impose a strict one-to-one correspondence between hand motions in the physical and virtual world, which has the advantage of feeling more intuitive but may be limited by tracking hardware or range limitations such as the length of the human arm. Conversely, non-isomorphic methods are not based in reality and often use "magic" tools such as stretchable arms [Pou+96], voodoo dolls [PSP99] or ray-casting [BH97] for selection.

Another method to classify selection and manipulation techniques is the decomposition of these techniques into smaller subtasks [BH99]. A limited number of these subtasks can be used to construct almost all selection and manipulation techniques. *LaViola et al.* use the example of a selection technique being divided into the subtasks of indicating an object, confirming the selection and feedback and then further decompose these into smaller subtasks, which may also be subdivided.

The third and most extensively discussed classification method is based on classifying techniques through metaphors, defining metaphors as a fundamental mental model of a technique. Six common metaphors and the possibility of hybrid approaches are documented by the authors, with the first metaphor being based on grasping, which is based on users reaching out and grasping objects with their virtual hand. Two approaches are identified: In hand-based grasping the user's hand is represented by a single-point effector while finger-based grasping simulates the fingers of the virtual hand separately (the position of the virtual fingers is typically based on the position of the user's actual fingers). The second metaphor is based on pointing, which allows for the users to select objects by simply pointing at them, even if they are out of their immediate reach. *LaViola et al.* identified the type of selection calculation as the best base for a decomposition of pointing metaphors and classify techniques as either vector-based or volume-based. Vector-based techniques simply use vectors to calculate where the user is pointing which allows for very simple implementations, but usually limits selection to single objects at a time. Conversely, volume-based techniques use a vector and a volume to select objects (singular or plural) that intersect the volume. Surface metaphors are the third class of metaphors and are used in interaction techniques for multi-touch surfaces. While usually used in 2D contexts, there are several possible applications for three-dimensional interaction. For example, pinching is often used as a technique that modifies the absolute scale of an object both in 2D and 3D, but in 3D it can also be used to adjust distance from the (virtual) camera. Unlike the first three metaphors, all of which are based on directly manipulating virtual objects, indirect metaphors are interaction techniques in which objects are manipulated indirectly. These metaphors can be grouped into three approaches, control spaces, proxies and widgets. The control-space approaches physically separate the space in which the user performs an action from the primary display method and then map the users actions onto the virtual environment. A proxy method supplies the user with proxy objects representing the entire world or objects in the world and maps changes in orientation and location resulting from manipulating these proxies back onto the actual objects. Widget techniques attach widgets to objects or place them in the environment. These widgets, if interacted with, affect the objects in the environment or the environment itself in a consistent mapping of the users actions. As opposed to almost all other metaphors, which are usually implemented in such a way as to use

only one hand, bimanual metaphors require the user to use both hands. Bimanual techniques can be further divided into symmetric techniques in which both hands perform the same actions and asymmetric techniques in which both hands perform different actions. Finally, hybrid metaphors combine techniques from one or several metaphors to create a more powerful UI. This can be achieved by either aggregating techniques, by letting the user or the system choose from different options, or through integration of techniques where the interface switches between suitable techniques based on context.

The second set of fundamental tasks is made up of travel and wayfinding. Travel is defined by *LaViola et al.* as the act of moving from the current location in either a set direction or to a target location. The definition of wayfinding is taken from the work of *Golledge* [Gol99] who describes it as the process of determining and following a path or route between an origin and a destination. Though they are discussed in the same chapter, the authors use different methods to classify travel and wayfinding techniques. Four methods to classify travel techniques are presented: distinguishing between active and passive techniques, or between physical and virtual techniques, through task decomposition, or by metaphor.

The first of the presented methods distinguishes travel techniques based on whether they are considered active or passive. A technique is considered active if the user directly controls the movement of the viewpoint, it is considered passive if the movement is controlled by the system. *Bowman et al.* include an additional option in this classification scheme, route planning, which is considered both active and passive as the user plans a route through the environment which is then executed by the system [Bow+99].

The second method distinguishes techniques depending on whether they use physical travel or virtual travel. In physical travel, the movement of the user's body is mapped to the viewpoint's movement, translating or rotating the viewpoint through its own movement and rotation. Virtual travel is used to describe techniques in which the user's body remains stationary while the viewpoint moves. The authors note that this classification method is orthogonal to the active/passive classification, thus forming a 2-by-2 design space.

In the same way as selection and manipulation, travel tasks can be classified through decomposition into smaller subtasks. Three subtasks can be identified [BH97]: direction or target selection, in which the user either selects a target location or a direction to move in, velocity or acceleration selection, in which users control their speed, and conditions of input, which defines how travel is initiated, continued or terminated. These subtasks can then be further decomposed into more fine-grained subtasks based on the techniques analyzed. *LaViola et al.* include a second task decomposition, which is based on chronological phases of travel tasks [Bow+99]. In this decomposition, four subtasks exist: First the user starts to move, indicates a position, then an orientation,

then stops moving. The authors note that some techniques may order these subtasks in different ways, for example having the user indicate the target position before starting to move. The indication of the target position is further decomposed into specification of position, velocity and acceleration. Three possible metaphors for the specification of the position are presented by the authors: discrete target specification, one time route specification and continuous specification of position.

The last classification method suggested for travel tasks is classifying them by the overall interaction metaphor. The authors note that this classification is not as useful as classifying by subtask, as it does not allow decomposition into subtasks directly, however this method is much easier to understand. *LaViola et al.* identify four common metaphors used in travel techniques: Walking, steering, target/route selection, and travel by manipulation. Walking metaphors are all based on walking, however due to technological and space limitations not all of them can simply map the actual action of walking to virtual movement. Thus, the authors differentiate between three different categories of walking metaphors, full-gait techniques which map the entirety of the human gait cycle, partial gait techniques which map only a part of the cycle, and gait negation techniques which negate the user's movement. While natural techniques, such as walking metaphors, are able to make use of the intuitive nature of walking, virtual techniques are far more common in most applications, with steering metaphors being the most common. In steering metaphors the user constantly specifies absolute or relative travel direction either through spatial interactions, in which control is exerted through the manipulation of a tracking device such as a head tracker, or through physical steering props such as a steering wheel. The next metaphor, selection-based travel, is based on the idea that the user selects either a target to move to or a path to travel, simplifying the usage by having the travel technique handle the actual movement. While target-based travel techniques are based on the user only choosing the target, teleportation through space is not the only possible choice in implementation. The application may also move the user through space continuously, quickly or slowly and using linear or complex paths. Route-planning on the other hand allows the user to exactly define the path the viewpoint will take to its destination before it is executed. Finally, manipulation-based travel tasks use hand-based manipulation techniques to either manipulate the viewpoint or the world to move the viewpoint to the desired position in the world.

Wayfinding in 3D environments is largely dependent on the number and quality of cues and aids provided to users through the UI and environment. These cues can be divided into user-centered wayfinding cues, which make use of the known characteristics of human senses to offset the fact that output devices do not fully match the human sensory capabilities, and environment-centered cues, for which the environment is designed in such a way that it aids the user in their wayfinding tasks.

The final interaction task presented by *LaViola et al.* represents one of the biggest challenges of 3D user interfaces: system control and symbolic input. The authors define system control as a task in which commands are issued to either request the system to perform a particular function, change the mode of interaction, or change the system state. They further note that while for other tasks the user specifies what should be done and how something should be done, system tasks usually leave the action to the system, only specifying the result. While issuing commands to a system is for the most part a solved problem in 2D, many 2D widgets and interfaces cannot be used or are extremely cumbersome to use in 3D. Thus, new ways to use old techniques and completely new techniques had to be found. To be able to gain an overview over all possible techniques, the authors introduced a classification system for system control techniques based on the main interaction styles. As the choice of style is often based on available input devices, most of the styles are based on certain input devices. Under this classification scheme, system control techniques can be based on physical controllers, graphical menus, voice commands, gestural commands, tools and multimodal techniques. Physical controllers can use buttons or switches and graphical menus can use adapted 2D menus, 1-degree-of-freedom menus, or 3D widgets. Gestural commands are decomposed further into mimic gestures, symbolic gestures, sweeping, sign language, speech connected hand gestures and whole body interaction. Finally, tools are divided into physical tools, virtual tools and tangibles.

#### Classification According To *Understanding Virtual Reality*

In their book *Understanding Virtual Reality* [SC03] *Sherman et al.* suggest a different set of fundamental interaction tasks, defining them as manipulation, navigation and communication. Compared to *LaViola et al.*'s approach, which treats them as connected tasks, they have decided to treat selection and manipulation as well as travel and wayfinding as elements of larger tasks and combine them into the tasks of manipulation and navigation, respectively. Additionally, they do not consider system control a distinct fundamental interaction task and instead add a third task named communication which they define as the act of communicating with either other users or agents within the virtual world.

*Sherman et al.* document four methods of manipulation: Direct control which has the user interact with objects as they would in the real world, physical control which has the user manipulate real objects such as controllers or cockpits, virtual control which has the user interact with virtual objects which are often mimicking physical controls and agent control which has the user issue commands to an agent which then executes those commands. As there can be no manipulation of objects without a selected object, three different types of selection are defined by the authors, direction selection which

has the user indicate a direction or vector in some way which is then used for selection of objects in that direction or as a travel direction, item selection which has the user choose items from an enumerated list often through methods of direction selection and finally symbolic input in which selection is performed through input of alphanumeric symbols.

Just like manipulation is split between selection and manipulation, navigation is split into the tasks of travel and wayfinding. Wayfinding refers to methods used to determine one's position in space or time and to determining routes to different locations in the environment. Several types of methods to aid the user in wayfinding tasks are documented by the authors. Marking a path through the environment, both through environment design or by displaying path markers in the UI is considered the easiest of the methods to implement. Maps can be used both as a wayfinding tool to help the user orient themselves and a navigation tool by combining wayfinding and travel. Similar to real landmarks, virtual landmarks placed in the environment can help the user determine their position in relation to one or several landmarks. In larger environments, memorable placenames are helpful for users to distinguish different parts of the environment and can be used in selection-based travel. Leaving breadcrumbs is used by the authors as a metaphor for leaving a trail through the environment which allows the user to retrace their steps. Compasses and other indicators of orientation allow the user of an application to easily ascertain their direction of travel. Similarly, instrument guidance can be represented through virtual representations of real instruments such as those used in aviation but also through UI markers such as large arrows guiding the user in a certain direction or even sound markers. Shifting the user's view from an egocentric to an exocentric viewpoint such as a world-in-miniature including a user avatar or a simple third-person perspective can be helpful to users in determining their own location. Displaying the user's coordinates or the distance of and direction to the nearest landmark helps the user in determining their current location and provides them with enough information to later find it again. Finally, one of the simplest methods to reduce the wayfinding needed is to constrain the user's movement or possible destinations in the virtual world. Travel, too, can be decomposed into different implementations. *Sherman et al.* define several methods of travel such as physical locomotion, in which the user's movements are tracked and mapped to movement around the environment, the "ride-along" in which the user is moved along a pre-determined path and can only look around, the towrope metaphor in which the user is towed along a path but retains enough freedom to veer from the path a little and the fly-through which allows the user to set a direction to move in and a speed to move at through an interface. Similarly, the pilot-through allows the user to move around the world by piloting a simulated vehicle. If a manipulation-based travel method is desired, a move-the-world method provides the user with the ability to move the world around

them to reach a different location, rather than moving their own viewpoint while a scale-the-world approach provides the user with a scaled-down model of the world which can be manipulated to move the viewpoint. The simplest way to implement travel, however, has the user select a target location which they are then moved to either instantaneously or on a path set by the application. Finally, the authors mention a technique for moving between different perspectives of objects called orbital viewing in which an object orbits around the user always staying in their field of view but retaining its orientation, thus allowing the user to view it from all its sides simply by turning their head.

Communication, or interacting with others in the virtual world is the third set of tasks defined by *Sherman et al.*. Communication becomes necessary if an experience in VR is shared, either by other users in the virtual world or by onlookers able to interact with or perceive the virtual world. If the objective of a shared experience is the solving of a task, the authors refer to it as collaborative. Several types of shared experiences are identified: one immersed person with one or several onlookers, two or more immersed persons, open displays allowing anyone to see the virtual world and a transferable tracking device, and finally a multi-person cockpit which uses screens through which anyone inside can see the virtual world. These methods strongly shape the interfaces needed for applications based on their implementation. In the virtual world, the authors differentiate between synchronous communication methods, which have no delay between one user sending and another receiving a message, and asynchronous communication methods, for which there is a delay between sending and receiving a message. Synchronous methods include real-time voice or video transmission as well as transmitted body language while asynchronous methods may be stored messages, playbacks of a users actions or changes to the virtual world such as annotations, which may take the form of text, sounds, gestures or pictures.

While system control (referred to as metacommands by the authors) is not considered one of the fundamental tasks by *Sherman et al.*, they do note its importance in fully utilizing virtual worlds. However, no further decomposition of this task is provided in the book.

### 4.1.2 Classifying 3D Menus

If one hears the words user interface, the first word many will think of is menus. While menus for two-dimensional environments have been extensively studied and are often quite similar to each other, system control in three-dimensional virtual environments has very different requirements while also opening up new options for menu shapes, hierarchies and interaction methods. A survey of available options has been conducted by *Dachselt et al.* in 2006, who then went on to use the results to create a taxonomy of

3D menus [DH07]. They identify a number of categories of criteria for classification.

The intention of use category is defined through two main properties, the number of displayed items, which may be limited to a small number by the shape of the menu, or by reduced usability if too many items are added, and the hierarchical nature of a menu, which is divided into four types: Temporary option menus, which appear only for a short time, allow the selection between a small number of items (usually options) and vanish after a selection is made. Single menus may appear for longer stretches of time or are even displayed continuously and their number of selectable items may be higher than the first type and can consist of arbitrary items. Menu systems extend single menus with one submenu for each item, which makes them menu hierarchies with a depth of 2. Menu hierarchies display an arbitrary number of items which may be arranged in an arbitrary number of submenus (of a depth of at least 3).

The appearance and structure category is made up of four main properties. The geometric structure describes the underlying geometry of a menu, such as a flat list, a ring, or a cylinder. The structural layout property, which has a strong impact on memorability and interaction speed, describes how the menu items are arranged either within space or on the supporting geometry. The possible structures are the acyclic list, the cyclic list, the matrix, free arrangements, and layouts following the geometric structure. The type of displayed data describes what a menu option appears as and includes: 3D objects, text entries, images, a combination of images and text, and a combination of 3D objects and text. The size and spacing of menu items strongly affects usability and is strongly dependent on the other properties.

The placement of a menu is defined through three properties. The reference of a menu describes in relation to which point-of-reference it is placed. To describe an object's reference the authors use the placement options described by *Bowman et al.* [LaV+05]. According to *Bowman et al.* a menu may be world-referenced (placed in the world, object-referenced (attached to an object), head or body-referenced (attached to head or body) or device-referenced (attached to a fixed point on the display device, for example the edge of a screen). The properties of orientation describes a menu's orientation and whether it is fixed in relation to the user or world while the property of repositioning is simply defined by the user's ability or inability to reposition a menu.

The invocation and availability category is made up of four different properties and describes how a user may invoke menu. The visibility of a menu can be of three different lengths: always visible, visible for the duration of the selection and visible for as long as the user wants it to be. The invocation of a menu can be achieved through four different means: selecting an icon or miniature, a context-dependent activation (which may relate to another menu, an object or a specific background), a free activation at an arbitrary point, or no action if the menu is always visible. The property of animation, which describes how a menu is animated becomes much more

important in three-dimensional space than in two-dimensional space as the number of possible animations vastly increases. Animation techniques also often affect the collapsibility of a menu, a property which describes how the menu or parts of it can be collapsed to temporarily reduce the space taken up by it.

One of the largest influences on menu design, the interaction and I/O setting is defined through five properties. The interaction device strongly shapes how a menu is operated and its accuracy and usability strongly affect the usability of certain types of menus. This is true for both input devices, which shape the users interaction and output devices which affect the user's perception of a menu. The application type and setting may be either Virtual Reality, desktop-VR, Augmented Reality, or a 3D mobile application. The dimensionality of a menu requires at least the same dimensionality in the input device, such as a 3D cursor being needed to select one of several floating menus. The degrees-of-freedom needed to operate the menu may be congruent to the DOF offered by the input device or current task, such as a controller which is able to input rotation being used to cycle through a ring menu, but it may also constrain that input, such as a tracked controller being used for pointing at a floating menu instead of having to be positioned to actually touch it. Another important property of a menu is how it highlights selected options and provides feedback on the selection to the user. The authors note that this is closely related to the last property, the visualization of the selection path in menu hierarchies, which is common in 2D menus but very rare in 3D.

The usability of menu is described using two properties. The evaluation criteria are used to formally evaluate a menu and usually include selection speed or average selection time, error rate, efficiency, user comfort, ease of use, and ease of learning. The combinability of a menu describes whether or not a menu solution can be combined with another solution to form a menu system or hierarchies, and if it can, with which other solutions it may be combined.

From these categories of classification criteria a taxonomy for 3D menus was created by *Dachselt et al.*, which combines the criteria hierarchical nature and structural layout. The choice of these two criteria as the base of the taxonomy is reasoned by the authors as making the taxonomy more useful to 3D menu developers, as these would first decide on a hierarchy for the menu depending on the use case. The structural layout then serves as a useful further division to the taxonomy in order to improve clarity. This results in a taxonomy which first divides menus into temporary option menus, single menus, menu systems, and menu hierarchies and then further divides each of these according to their structural layout into lists, rings, matrices, geometric structures, and free layouts.

## 4.2 Classification Of 3D User Interfaces In This Work

As the classification schemes presented in section 4.1 each fail to capture the entirety of the possible 3D user interfaces, there is a necessity to either develop a new scheme or expand one of the current schemes using knowledge gained from other works and our own results. As the work in *3D User Interfaces: Theory and Practice* [LaV+17] is still considered the standard work on the subject, their overall classification scheme serves as the base for the expanded scheme. Thus, the overall classification of user interfaces will be by basic interface tasks. *LaViola et al.* identify three sets of basic tasks, *Selection and Manipulation*, *Travel and Wayfinding* and *System Control*. However, these tasks fail to capture all possible tasks documented in prior literature. To accommodate the additional tasks found by *Sherman et al.*, this work adds the tasks *Communication*, and *Collaboration* and as they are related very closely, discusses them in a single chapter. The following section will give an overview of how the classifications discussed in this chapter are applied to the overall tasks defined.

### 4.2.1 Selection And Manipulation

Selection and manipulation techniques are grouped in their chapter according to the task metaphors found by *LaViola et al.*. However, as there are many graphical aids in the selection and manipulation process which are not specific to a certain metaphor, they are discussed separately from the techniques in their own section.

### 4.2.2 Travel And Wayfinding

Travel metaphors are grouped into their respective metaphors, however only brief discussion will be given to metaphors such as walking which contain little to no visual elements. Wayfinding aids are grouped into the categories defined for them by *Sherman et al.*, though categories which did not appear in their work but were defined by *LaViola et al.* will also be discussed. While it would be possible to divide these categories into user-centric and environment-centric aids, this distinction was not made by *Sherman et al.* and for the sake of congruency will not be made in this work either.

### 4.2.3 System Control And Menus

As this work is focused on the visible aspects of user interaction, voice and gesture commands are discussed only if they are used in conjunction with a notable visual component, such as a menu, thus reducing system control to the two classes of menus and tools. As symbolic input which is not inputted using a keyboard is inputted using

graphical menus, surveying and classifying graphical menus according to the taxonomy developed by *Dachselt et al.* covers both possible symbolic input methods and menus.

### 4.2.4 Communication And Collaboration

Communication methods are split based on synchronicity and then classified into the categories developed by *Sherman et al.* while the visual aspects of shared environments are classified based on the type of shared environment they are used in.

# 5 A Survey Of Visible 3-Dimensional User Interfaces

The vast field of 3D user interfaces provides almost infinitely many solutions to any given user interface task. As this makes exploring all possible solutions an impossible task, this work focuses on only one aspect of user interfaces: the visual solutions to tasks and how their different approaches affect their usability. In this chapter the classifications developed in the previous chapter are used to classify existing solutions from the current state-of-the-art in the field of 3D user interfaces. Each of the possible solutions that was found for an interface task is sorted according to the methods and metaphors used and examples of its usage are provided. Then, possible advantages and disadvantages of the technique are discussed and, if possible, a conclusion about its usability is drawn.

## 5.1 Selection And Manipulation

Whether the user intends to interact with the environment or menus, selection and manipulation are the two most fundamental tasks in a 3D environment and often underlie many of the other tasks. Tackling a manipulation task usually begins with a selection technique since the user needs to choose which object is to be manipulated. Additionally, the method used to manipulate an object is derived from the metaphor used in its selection. This section deals with several techniques for selection, grouped by the metaphor used, and finally discusses visual aids used in the selection and the manipulation process.

### 5.1.1 Selection And Manipulation Techniques

#### Grasping Techniques

Grasping an object, as one would do in the real world, is the most natural of all selection and manipulation techniques. According to *LaViola et al.* [LaV+17] many first-time users will attempt to grasp objects independent of whether the application supports selection through grasping. There are two kinds of grasping techniques, hand-based grasping and finger-based grasping.

**Hand-Based Grasping** As most three-dimensional user interfaces provide a tracked controller or tracking device to determine the position and orientation of the user's hand, but usually do not track the user's individual fingers, hand-based grasping techniques represent the user's hand as a single object, usually a virtual hand. The position and orientation of this virtual hand corresponds to the position and orientation of the user's tracked hand. *Zhai et al.* note that rendering this hand semi-transparently both improves depth perception of virtual objects but also reduces occlusion of the environment as objects remain partially visible behind the cursor [ZBM94].

Figure 5.1: The player reaching out to grasp a moving disk in the game *Echo Arena* [Lak17]

A simple implementation of this technique is found in the game *Echo Arena* [Rea18]. The objective of the game is to grab a disk and throw it through the enemy team's goal in an environment in which 6-DOF movement is possible. To select and manipulate the disk, the user is holding a tracked controller in each hand whose positions and orientation is tracked and mapped to the position and orientation of the virtual hands of the player's avatar. To grab a ledge, handle, lever, or the disk the user simply needs to move their virtual hand to the position of the desired object (see figure 5.1), moving near it if it is out of reach, and press a button on their controller to grasp it. If a manipulatable object is grabbed, the user can then hold the button and move their hand to move and rotate the object until the button is released, at which point the virtual hand releases the object, which retains any momentum it gained in the interaction. This can be used to throw objects, such as the disk, making the grasping action essential to scoring in the game. While this simple technique has the advantage of being extremely

easy to understand and execute due to its similarity to the real world action, its biggest disadvantage is the limitation of its selection range to the range of the user's arms. An interaction which alleviates this problem, the Go-Go technique, was designed by *Poupyrev et al.* [Pou+96]. The Go-Go technique uses a simple virtual hand, but gives the user the means to extend the length of their virtual arm to reach far-away objects. This is achieved by setting a threshold value for the distance of the user's hand from their body: below this threshold the hands movement is mapped linearly to the virtual hand, beyond this threshold the mapping is done non-linearly and the virtual arm is extended towards the user's target. The Go-Go technique has the advantage of being extremely similar to the real world movement of reaching out for distant objects, while also seamlessly combining manipulation and selection for nearby and far-away objects. The authors do note however, that ray-casting is more efficient at simply selecting distant objects. Additionally, while the user's reach is greatly extended, it is still limited by the length of their arms and the further away an object the user is trying to reach is, the harder will it be to precisely grasp it, as any small movements they make are amplified by the non-linear mapping.

**Finger-Based Grasping** While hand-based grasping tracks only the position and orientation of the user's hand, finger-based grasping achieves much greater precision and much more control by tracking each of the user's fingers. This allows for much more complex interactions such as moving objects around between virtual fingers, holding multiple objects between fingers or manipulating only parts of objects, such as small handles or buttons, using single digits. *LaViola et al.* [LaV+17] point out two disadvantages common to all finger-based techniques: the lack of haptic feedback on almost all tracking devices (which are usually computer vision-based) making it hard for users to move their fingers in such a way as to manipulate objects in the desired way, and the requirement that the virtual fingers do not penetrate virtual objects combined with the lack of haptic feedback leading to situations where the virtual fingers' position does not match the user's real fingers. *LaViola et al.* also differentiate between three common techniques for finger-based grasping, rigid-body fingers, soft-body fingers, and god fingers.

A simple implementation of rigid-body-based fingers was developed by *Borst et al.* [BI05] using bend-sensing gloves. The gloves track the position and orientation of the user's fingers, which are then mapped onto a virtual hand and fingers, modeled as rigid bodies.The authors used a system of spring-dampers to create realistic physical interactions between hands and objects by having the spring system dynamically affect the mapping between the user's real hand, referred to as the tracked hand by the authors, and the virtual hand, referred to as the spring hand. When not interacting

with any objects the position and orientation of the spring hand usually matches the tracked hand, with a difference usually appearing only if an object is grasped, at which point the spring system will prevent the spring-hand from penetrating into the object if the tracked hand is moved inside of it. Additionally, the spring system maps this penetration into a force applied to the object, enabling the user to pick up and manipulate objects. The main disadvantages of this technique are two-fold: a certain visual stiffness of the fingers resulting from them not quite grasping an object like real fingers would due to rigid bodies not deforming and the forces resulting from the spring system making it hard to precisely release objects without the help of an heuristic-based release function due to the virtual fingers sticking to objects [PB12].

Soft-body fingers were only recently made possible by advances in the field of fast deformation algorithms such as the development of the FasLSM algorithm [RJ07] in 2007, which is used in the soft-body fingers implementation developed by *Jacobs et al.* [JF11] in 2011. Their approach was to use the FastLSM algorithm to be able to model soft body fingers attached to a rigid body hand skeleton. When coming into contact with a virtual object, the pads of the virtual fingers would deform. If the user simply touched an object, very little deformation would take place and there would be few points of contact, but whenever the user's real (tracked) fingers penetrated into a virtual object, the deformation would be more pronounced leading to an increased number of points of contact and increased friction between the object and the fingers, thus making the user able to firmly grasp it. The main advantages of this technique are its ability to model the increased friction and firmer hold that results from a firmer grip as well as the fingers visibly deforming to accommodate virtual objects, leading to a more realistic appearance of the virtual hand.

While soft-body fingers lead to a more realistic interaction and more realistic visuals, they are still computationally expensive compared to rigid-body physics [TML13]. An alternative are god objects [ZS95] which are virtual points that adhere to the rigid-body physics of the environment and thus never penetrate into virtual objects but are mapped to the tracked positions of the user's fingers and can thus be used to calculate and apply forces when the user's real fingers penetrate into a virtual object. The god-fingers technique developed by *Talvas et al.* uses this concept to simulate the interactions of soft-body fingers and virtual objects. First the contact area of the virtual finger pads and an object are calculated as if they were flat, the contact area is deformed to fit the object in the god object's force direction and an angular threshold between the face normals of the virtual object and the contact normals is applied to filter out odd deformations. The main advantage of this technique is successfully replicating the interaction between virtual objects and soft-body fingers, including increased friction and the ability to balance spherical objects on top of the virtual fingers (see figure 5.2). It does not however lead to any visual deformation of the fingers, which may lead to

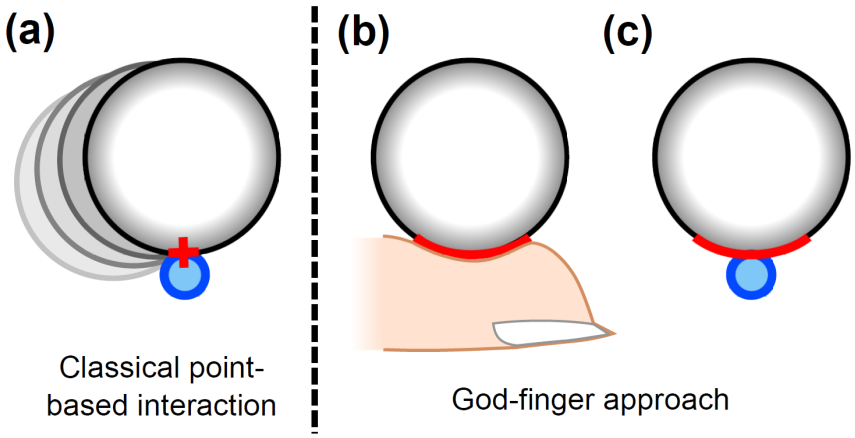



Figure 5.2: A picture from *Talvas et al.*'s work showing the differences in object interaction between (a) rigid-body fingers of which a spherical object would roll off and (b) real fingers on which it would balance and (c) how the god-finger approach simulates the interaction of real fingers using only one point of contact [TML13]

the same visual stiffness seen in the rigid-body fingers technique.

### Pointing Techniques

Pointing implementations are one of the simplest and most intuitive selection methods for objects which are outside of the user's immediate reach. Pointing techniques are based around a vector or volume originating from a tracked hand, headset, or controller intersecting an object and the user selecting the intersected object by issuing a trigger, such as a button press. The selected object is then attached to the pointing vector and can be manipulated simply by moving or rotating the origin of the ray. While quantitative analysis has shown pointing to be more effective at selecting objects than grasping [I+98], it generally does not work as well grasping if one wants to move an object as the distance between the user and the object usually can not be changed easily and rotation can usually only be performed efficiently around one axis.

**Vector-Based Pointing** Vector-based pointing techniques use only a vector for selection, however implementations may differ from its simplest form, which is ray-casting. Ray-casting uses a a virtual ray attached to a tracked controller (usually represented by a virtual hand or a virtual model of the controller in the environment) which the user

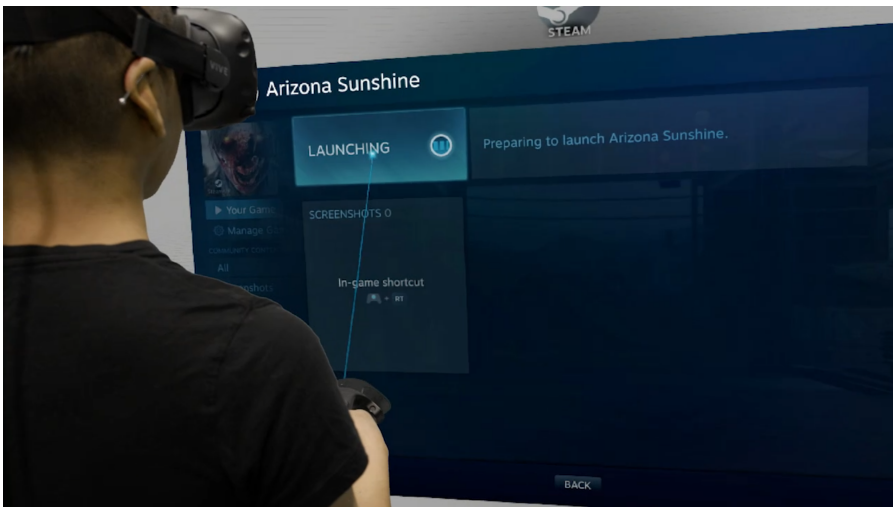



Figure 5.3: A screenshot of a composite video showing ray-casting being used to select menu items in the SteamVR application [Val18]

can point at any object to select it, usually by pressing a button on their controller. In rare cases, the ray does not originate from the user's tracked hand but instead from their tracked head, letting the user select objects by looking at them. One example of this is used in the game *Eve Valkyrie* in which the user is able to press a button and look at an enemy until a visual indicator confirms selection [CCP17]. Releasing the button after selection is confirmed fires missiles at the selected enemy. Usually, if the ray originates from the user's hand, a short segment of the ray or the entire ray up to its intersection with an object (or the horizon) is rendered. Rendering only a small segment has the advantage of only occluding very little of the user's field-of-view, but may make it harder to see if the ray is currently intersecting an object especially if that object is far away. Displaying the entire ray has the advantage of the user being able to see immediately if the ray intersects an object or not and makes it easier to move the ray towards a selection target but may occlude more of the user's view and reduce immersion. While ray-casting is extremely efficient at selecting objects, it is difficult to use when selecting distant or small objects as these require high accuracy to select [I+98], which is difficult to achieve as the ray amplifies small movements and rotations of its origin into large movements of it's endpoint.

A possible way to make it easier for users to select and manipulate objects at longer distances are the image plane techniques first devised by *Pierce et al.* [Pie+97] which reduce the degrees-of-freedom of the interaction to two dimensions by projecting them onto a two-dimensional image plane right in front of the user. One of the simplest

ways for the user to select an object is the sticky finger method in which the user simply reaches out to touch an object with their finger or a hand-held tool. A ray is cast from the position of the user's head through the finger to calculate which object is selected, resulting in the illusion that any object, no matter how far away can simply be "touched" to select it. The selected object is then scaled down and moved to the image plane, where it can be manipulated and rotated. Once the user is finished, the object is scaled up again placed at its original distance from the user.

One of the problems affecting almost all pointing techniques is the inability to change the distance of the selected object from the user, making it hard to position it if the intended new location is closer to or further away from the user than the original position. The fishing reel technique developed by *Bowman et al.* [BH97] gives the user a separate means to control the length of the ray an object is attached to, using a pair of buttons attached to the controller. Pressing one button brings the object closer to the user, pressing the other button moves the object further away.

**Volume-Based Pointing** As opposed to vector-based pointing, which requires only a vector, volume-based pointing uses a vector and a volume, usually relating the volume to the vector in some way. The simplest such technique is the flashlight selection technique developed by *Liang et al.* [LG94], which gets its name because it uses a cone-shaped volume, similar to a flashlight's beam, around the pointing vector to select objects. Instead of having to accurately point a vector at small objects the user now has a much larger margin of error due to the object only having to intersect a volume instead of a vector. However, this may lead to difficulties when multiple objects intersect the cone at once. *Liang et al.* suggest two rules to choose which object to select: If two or more objects are inside the cone, the object which is closest to the pointing vector is selected. If two or more objects are at the same angle between them and the pointing vector, the closest object is selected. However, if many objects are grouped tightly together and partially occlude each other the flashlight technique may make it difficult or even impossible for the user to select the desired object. One way to alleviate this drawback is the aperture technique developed by *Forsberg et al.* [FHZ96]. It uses a gaze-directed variant of the flashlight technique in which the origin of the cone is the user's viewpoint, which is determined using a tracked headset, and the center vector of the cone is determined by the vector between its origin and the tip of a tracked hand-held wand, represented in the virtual environment by a circle and a crosshair inside the cone. If the cursor is moved closer or further away from the user, the cone expands or narrows thus allowing the user to control the size of the cone allowing them to accurately select objects which are clustered in groups by decreasing cone size or small or distant objects by increasing the cone size. To disambiguate selection if several

objects are grouped extremely close or attached to each other, two small, parallel plates are displayed as part of the aperture geometry, which can be rotated by rotating the controller. If all objects present in the cone have a similar orientation the flashlight technique's rules for object selection are applied.

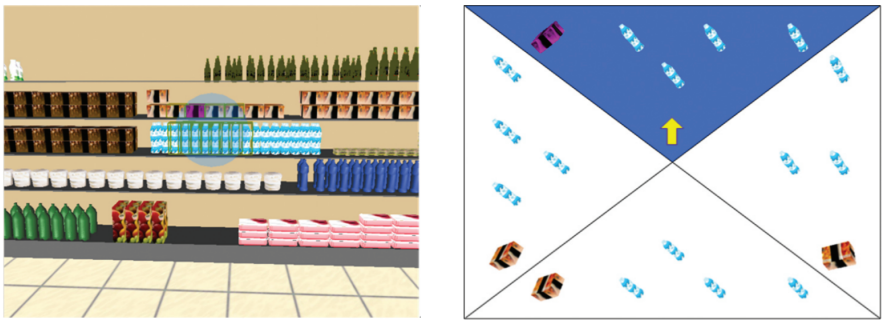


Figure 5.4: Screenshots showing a sphere being cast (left) and the items in the sphere being available for selection in a QUAD-menu (right) [KBB11]

A different approach to using volume-based pointing is the sphere-casting technique developed by *Kopper et al.* [KBB11]. This technique uses ray-casting to let the user determine a general area for selection. All objects in a sphere of predetermined size around the intersection between the ray and an object or surface are able to be selected through a menu. *Kopper et al.* use a QUAD menu for this purpose, which divides all selectable objects between four quadrants, having the user choose one quadrant and dividing all its contents between the four quadrants until only the desired object is left. The largest disadvantage of this method is the requirement for the desired object to be visually distinct from all other objects in the sphere as otherwise it cannot be distinguished from other items in the QUAD-menu (see figure 5.4).

### Surface Techniques

While surface-based techniques require a multi-touch device in addition to the virtual or augmented reality display, using multi-touch capable devices as input devices or controllers has become more common, not in small part due to their familiarity to those who have used a smartphone or tablet before. While many two-dimensional input techniques on a surface provide little visual feedback aside from the selected object moving or rotating, they are essential to understanding three-dimensional surface-based manipulation techniques and are often used in three-dimensional interaction techniques as well and will thus be covered in the following section.

**Surface-Based 2D Interaction Techniques** Many of the two-dimensional manipulation methods using a multi-touch surface can be compared to moving pieces of paper around on a flat surface without lifting them up. Selecting an object surface is usually achieved simply by touching it, deselecting it is usually done simply by ceasing to touch the object. To move an object, the user touches it with one or more fingers (usually using one finger [HC11]), and slides the fingers to the desired position. The object will then be moved from the initial position to the last position the user touched before lifting their fingers from the surface, usually along the path the user moved their fingers, but may also simply be teleported to the new position. If one wants to enable the user to rotate an object, several approaches exist. *Hancock et al.* [Han+06] describe the most common approach as a rotation around the center of the object. The user triggers a rotation by selecting the object through touching a specific area, usually the corners, and drags this area, which will rotate the area around the object's center in the direction of the user's finger's movement. Another method suggested by *Hancock et al.* is a simultaneous rotation and translation using two fingers. The first finger is used to select the object and as a center for the rotation. The object can be dragged around using two fingers and its rotation is determined by the angle between the vector between the first finger's and the second finger's starting position and the vector formed by their current or final positions. Finally, if the user needs to scale an object up or down, one of the simplest methods is pinching. *Wobbrock et al.* [WMW09] describe pinching as the user selecting an object using two fingers and the system increasing the object's size if the distance between the two fingers increases and decreasing it if the distance is decreased.

**Surface-Based 3D Interaction techniques** While manipulation on planes parallel to the surface can be achieved simply by using two-dimensional interaction techniques, manipulating the depth of an item in a three-dimensional environment using gestures on a two-dimensional surface has no real-world equivalent to be used as an intuitive metaphor. One method, which, while not using a real-world metaphor, is familiar to most users was suggested by *Giesler et al.* [GVH14]. They suggest adapting the two-dimensional pinching method in such a way that instead of increasing or decreasing the absolute size of an object in relation to the distance between the user's fingers, the depth of an object is decreased or increased, having the same effect on its visual size but giving the user the means to manipulate an objects depth. However, while this method is easy for users to learn due to its familiarity and low complexity, it is unable to manipulate objects which are occluded by other objects in the foreground requiring the user to move those objects out of the way first. To solve this problem, *Giesler et al.* developed a manipulation method based on the concept of void shadows [GVH14]. In an application

using the void shadows method, each object casts a shadow onto the touch surface, which is calculated using an invisible shadow plane located slightly above the display's position in the virtual environment. This results in stacked objects of the same size casting nested shadows allowing each object to be selected and manipulated separately simply by dragging, rotating or pinching the object's shadow.

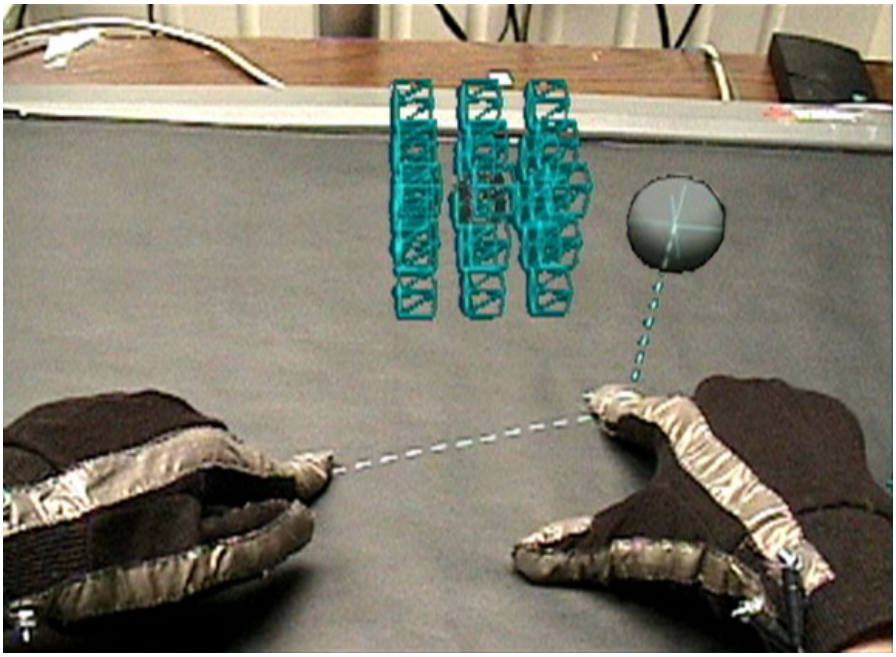


Figure 5.5: A screenshot showing the balloon selection technique being used in Augmented Reality [BF07]

An alternative to the void shadows method which still allows the selection of objects at various depths is the balloon selection method developed by *Benko et al.* [BF07]. It's name is derived from the metaphor it is based on, controlling the height of a balloon which is anchored to a point by changing the length of the rope anchoring it. The technique uses two contact points to control a spherical 3D cursor. The first contact point defines the position of the 3D cursor in the horizontal plane (representing the point it is anchored to) while the second contact point simulates the user grabbing the string which means that increasing the distance between the contact points decreases the cursor's height (pulling on the string) while decreasing the distance increases its height (releasing more string). A third contact point may be used to control the size of the spherical selection volume to be able to both easily select small objects and accurately select objects from closely clustered groups. *Strothoff et al.* [SVH11]

also suggested that a two-finger rotation technique, as described in the section on 2D surface-based techniques, could be used on the anchor point to rotate a selected object around the vertical axis. Compared to both a keyboard and a virtual-hand-based grasping technique, *Benko et al.* found balloon selection to be faster than using a keyboard for positioning and more accurate than a grasping technique. Expanding upon the balloon selection technique, *Daiber et al.* developed the corkscrew widget. The corkscrew widget uses a spherical 3D cursor on top of a circular anchor point, similar to the balloon selection technique.The user can move this cursor on the horizontal plane by touching and dragging it. Unlike the balloon cursor which uses a second contact point to adjust the height of the cursor, the height of the corkscrew widget's selection volume can be adjusted by performing a rotation gesture on the edge of its anchor point: a clockwise rotation decreases the selection volume's height while a counterclockwise rotation increases it. Comparing the corkscrew widget to the balloon selection technique, *Daiber et al.* determined that the corkscrew widget allowed for faster depth-based manipulations but also lead to more errors than the balloon selection technique. Thus, they recommend the corkscrew widget to be used wherever fast interactions are more important than precise manipulations.

Another technique, developed explicitly to avoid occlusion and accuracy problems of other surface-based 3D interaction techniques, is the triangle cursor developed by *Strothoff et al.* [SVH11]. The user chooses the triangle cursor's position on the horizontal plane using the thumb and index finger, the midpoint between the two contact points with the surface defining its position. The height of the selection volume is defined by the distance between the two contact points. If the distance is decreased to zero the cursor will touch the surface, while increasing the distance to the maximum distance the human hand is capable of moves the cursor to its maximum height in the virtual environment. The cursor itself is a spherical selection volume displayed at the top of a triangle formed by itself and the two contact points, connected to the midpoint between the two contact points by a vertical line (see figure 5.6). To give users the ability to rotate an object around the vertical axis, *Strothoff et al.* adapted the two-finger rotation technique, described in the section on 2D surface-based techniques. Instead of rotating the object around one of the contact points, the object is rotated around the cursor's position. Comparing the triangle cursor to the balloon selection technique, they found that it lead to fewer positional and rotational errors and faster task completion, speculating that these advantages may stem from the triangle cursor's user only needing to use one hand, instead of the two hands needed for the balloon selection technique.

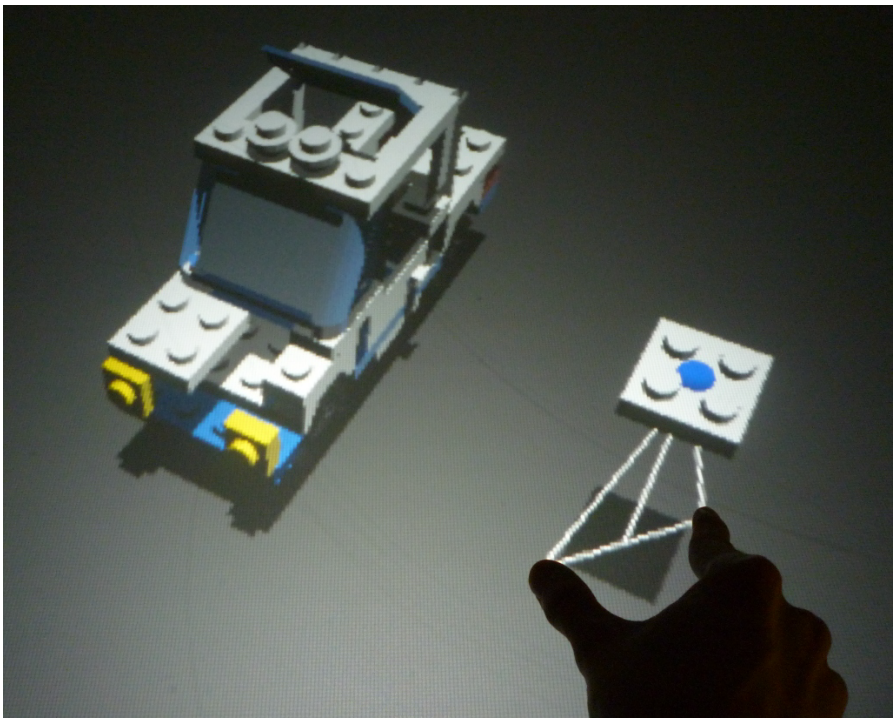


Figure 5.6: A screenshot showing the the triangle cursor being used to manipulate an object [SVH11]

### Indirect Techniques

In many situations, some aspects of the environment prevent the user from directly manipulating an object, whether it is simply located too far away to reach without travel or obscured by other objects, walls, or even the user's own body parts. There are three different approaches to indirect manipulation: indirect control-space techniques, proxy techniques and widget techniques.

**Control-Space Techniques** Control-space techniques separate the space in which the user controls the interaction from the display space or the space in which the virtual environment is displayed. The user's interactions are then mapped back to the virtual environment.

One of the most common ways to implement separation between display and control space is using a stereoscopic screen to display the three-dimensional virtual world while using a multi-touch surface for interaction. *A. L. Simeone* refers to this technique as indirect touch [Sim16]. Compared to direct touch techniques there are two important

differences. The first difference is how the touch surface is mapped to the display. The touch surface can be mapped absolutely, meaning the cursor is always on the same point on the display that the user is touching on the surface, or relatively, meaning the cursors stays where it was even if the surface is touched but if the contacting finger is moved it is moved in the same direction. The second difference is whether techniques from direct touch surfaces are used or new, indirect techniques such as those of *Simeone* are used. Additionally, while direct touch techniques often don't need a visible cursor, the cursor is usually always visible on indirect touch techniques. The main advantage of indirect touch over direct touch techniques is improved ergonomics, due to the use of a vertical display avoiding neck straining and a horizontal touch surface being less tiring than a vertical one. Indirect touch techniques also avoid the occluding of objects on the display by the user's hands which is common in direct touch techniques. *Simeone* compared indirect touch to two different direct touch techniques (DS3, triangle cursor [Sim16]) and did not find any significant performance decreases while noting that indirect touch did make the viewing experience more comfortable. *Knoedel et al.*, however, found that indirect touch required significantly more time to manipulate objects than direct touch techniques but noted that it increased accuracy when moving objects along optimal trajectories and when placing objects [KH11].

**Proxy Techniques** Proxy methods make use of local proxies of remote objects to circumvent common usability issues and allow for more natural selection metaphors such as grasping to be used on remote objects by transferring changes made to the proxy's position, scale and orientation back to the remote object.

One of the most well-known proxy methods used for remote manipulation is the world in miniature technique developed by *Stoakley et al.* in 1995 [SCP95]. In this approach a virtual miniature model of the environment they are located in is given to the user, represented in initial study by a physical prop with a tracker on it. The user is then able to interact with any object in the environment simply by grasping and manipulating its copy in the miniature. As the world in miniature is an exact copy of the environment, walls and other objects may obscure parts of the model. Backface culling, in which only faces facing the camera are rendered, is used to hide the walls in such a way that the user is always able to see into the room (see figure 5.7), making sure that the user is always able to see and interact with every part of the environment. The obvious advantages of the world in miniature approach are the ability to quickly select and manipulate any object in the environment, whether or not it was initially located within the users reach. It is also possible to use this technique for both travel and wayfinding by adding a manipulatable user avatar to the miniature. The main disadvantage of this technique is its limited scale. As environments become

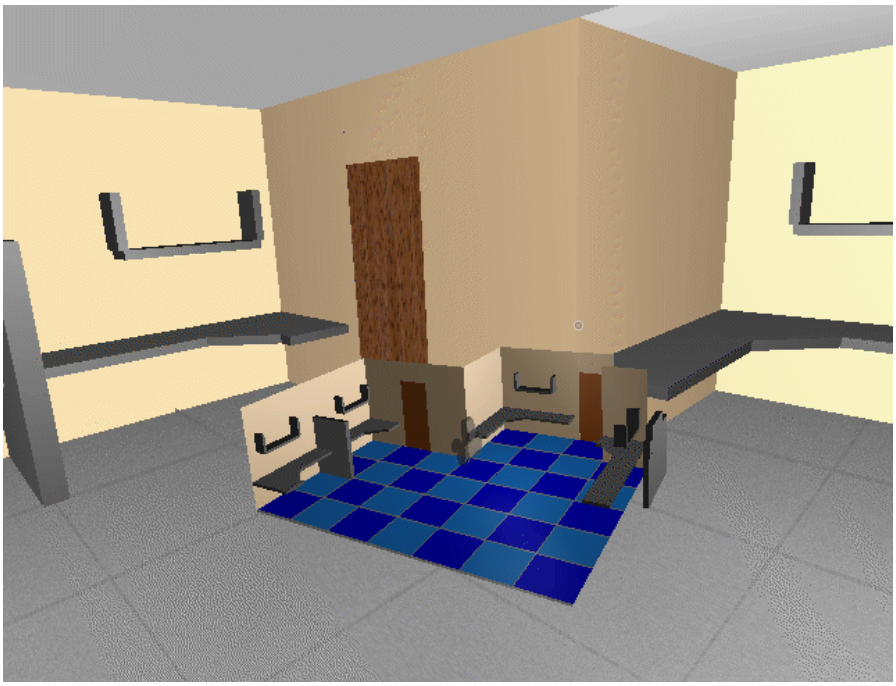


Figure 5.7: The virtual environment and its miniature replica [SCP95]

bigger, the miniature world either grows, negating its use for remote manipulation by moving parts of it out of reach or the proxies of objects become too small to naturally interact with, reducing its usability. More recently however, several solutions aiming to eliminate this drawback have been developed, such as *Wingrave et al.*'s work [WHB06] which suggests the use of scaling and scrolling might counteract the drawbacks of large environments. However, complex environments containing several rooms adjacent to each other both horizontally and vertically still pose problems as backface culling and even newer, more advanced algorithmic solutions still fail to properly address the problem of how to access rooms occluded on all sides by other rooms.

**Widget Techniques** In the context of indirect interaction, widgets are objects placed within the virtual world that if directly interacted with affect other objects in the world.

By far the most common type of widgets, often defined so broadly as to be considered a supertype of other widgets, are 3D widgets, first proposed by *s. Houde* in 1992 [Hou92] and further improved by *Conner et al.* [Con+92]. *Conner et al.* define a widget as "an encapsulation of geometry and behavior used to control or display information about application objects". 3D widgets differ from 2D widgets by the number of degrees-of-freedom. While 2D widgets offer only three DOF, 3D widgets are able to provide

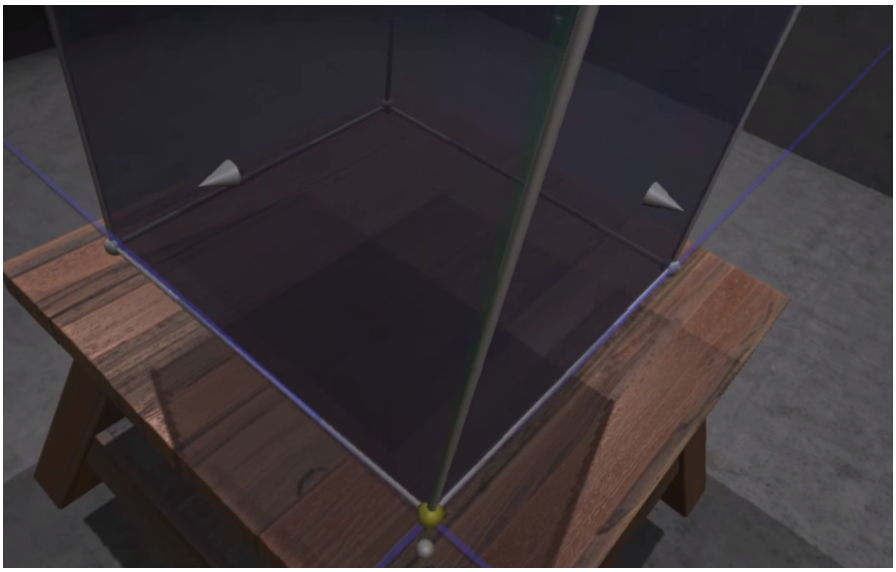


Figure 5.8: A screenshot of arrow-shaped widgets used to move a model in the *VRTX* application [BLA16]

up to 6 DOF, not all of which have to be used at once by the same widget, however. The most basic approach to manipulation employed by *Conner et al.* is adding widgets representing handles to a virtual object, allowing the user to move an object by selecting one of its handles and moving it. If one wants to give the user the ability to rotate an object, the simplest widget would be a sphere, that if selected maps the movement of the cursor or controller between selection and deselection onto a rotation in the movement direction. They also propose a color picker widget and a widget that allows snapping movement of one object to another. All these basic widgets can be combined into more complex widgets fulfilling more specialized tasks such as moving or rotating only part of an object, effectively deforming it. The main advantages of 3D widgets are their intuitive nature due to their shape usually being representative of the task they fulfill (see for example figure 5.8) and the ease of switching between different manipulation tasks by simply selecting a different widget. The main disadvantages of 3D widgets are the visual clutter added to a scene by their presence, often occluding important details, objects or each other, as well as an additional learning period for new users as different widgets may react differently to the same input such as a widget used for moving and a widget used for rotating an object may react to the same cursor movement with different rates of change.

### Bimanual

Unlike most of the previously discussed techniques, which require only one hand, bimanual techniques require the user to use both hands. *Ulinski et al.* identify two distinct properties of bimanual techniques: symmetry and synchronicity [Uli+09]. In a symmetric method both hands perform identical movements and a synchronous method uses both hands at the same time. Conversely, in an asymmetrical method both hands perform different movements and in an asynchronous method both hands move at different times. This results in four different classes of methods: symmetric-synchronous methods, symmetric-asynchronous methods, asymmetric-synchronous, and asymmetric-asynchronous methods. For simplicity's sake, this section will be grouped simply by symmetry, first discussing symmetric, then asymmetric methods.

**Symmetric Bimanual Techniques** The Spindle technique developed by *Mapes et al.* in 1995 is a symmetric-synchronous technique which uses two 6-DOF controllers to define a virtual spindle which reaches from one controller to the other [MM95]. The midpoint of this spindle is used to select and manipulate virtual objects, thus requiring the user to move both hands in order to move an object without affecting its rotation, as the entire spindle has to be moved. If the hands are rotated in relation to each other, the orientation of the spindle changes, which can change the yaw and roll of the selected object. If the user moves their hands closer together or further apart, the scale of the object is decreased or increased. *Schultheis et al.* conducted a comparative study and found that while the spindle required an initial learning and practice period, it outperformed both a virtual hand and a mouse-based technique for 3D manipulation [Sch+12].

The intersection-based Spatial Interaction for Two Hands (iSith) is a symmetric-synchronous technique developed by *Wyss et al.* [WBB06]. The iSith technique uses two separate 6-DOF controllers from which two rays are cast. The shortest possible line between the two rays is calculated and the midpoint of that line used as a selection and manipulation cursor, similar to the Spindle technique's midpoint of the spindle. If the controllers are rotated the position of the shortest possible line and thus its midpoint change allowing quick changes to the cursors position without large hand movements.

Finally, all selection and manipulation techniques which can be performed using only one hand are symmetric-asynchronous bimanual techniques if the user is enabled to perform them using either hand.

**Asymmetric Bimanual Techniques** Several asymmetric bimanual techniques, such as the balloon selection technique which is asymmetric-synchronous, have been discussed

in previous sections. This section, however, will focus on techniques which do not fit any other classification.

The Spindle technique was expanded in 2015 by *Cho et al.* who added a feature which allowed users to change the pitch of selected objects [CW15]. They achieved this by adding a wheel to the virtual representation of the dominant hand's controller, from which their new technique derives its name, Spindle + wheel. Rotating the controller around the wheel's axis, pitching it, changes the pitch of the selected object. The technique is considered asymmetric, unlike the original Spindle technique, which was symmetric, due to the wheel only being added to the dominant hand's controller. A user study performed by *Cho et al.* found that expanding the Spindle with a wheel resulted in faster completion times compared to one-handed techniques for all tasks which required 6-DOF manipulation and scaling. However, when no scale changes were required, accidental scale changes reduced performance compared to methods which separated 6-DOF and scale controls.

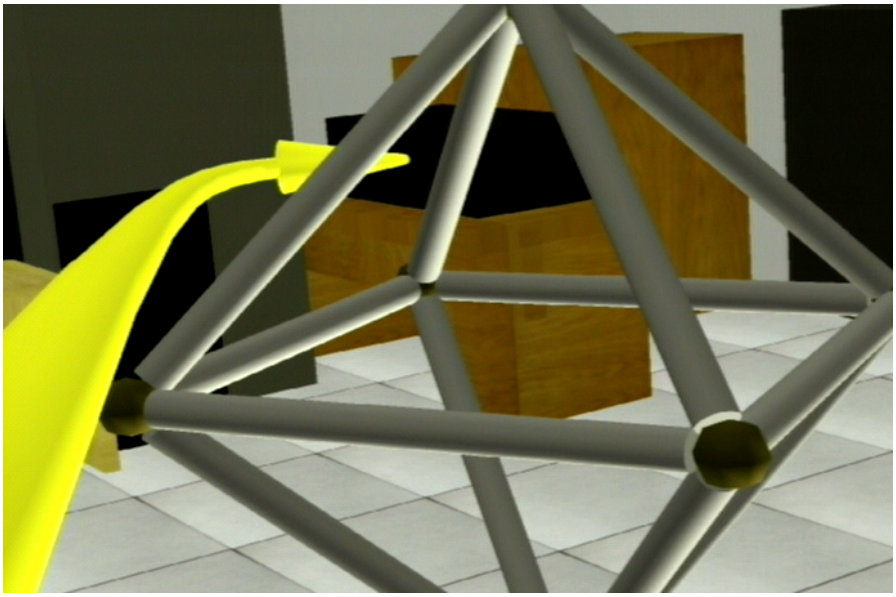


Figure 5.9: A screenshot of the flexible pointer being used to point around on obstruction [OF03]

The idea of two-handed ray-casting, an approach that is quite different from the bimanual techniques discussed before, was initially discussed by *Mine et al.* [MBS97] as a traveling technique in which the user's hands defined a ray along which the user moved. One hand's position was used as the origin of the ray while the other's position was used to define the direction of the ray, allowing the user to point in more directions

without turning. One example is the ability to easily point behind them simply by moving the hand from which the ray originates further away from them than their other hand. *Olwal et al.* used this idea in a selection technique [OF03] which uses a curved ray to select partially occluded objects. The flexible pointer is displayed as a large yellow arrow originating from the hand nearest to the user and always pointing at the axis defined by the positions of the user's hands. It is implemented as a quadratic Bézier spline whose curvature is defined by three points, the user's hands and a control point which moves closer to the hand whose orientation deviates the furthest from the direction of the axis. This means the user can affect the pointer's curvature simply by bending their hands.

#### Hybrid

As so far no manipulation method has proved to be superior in every situation or task, combining techniques to compensate for their weaknesses is an oft-discussed direction of research. *LaViola et al.* [LaV+17] have identified two approaches, technique aggregation and technique integration. Technique aggregation simply means implementing several techniques and providing the user with either a manual or an automatic mechanism to switch between these options. One of the most common mechanism for switching between techniques are 3D menus, which are discussed in section 5.3.1. Technique integration, on the other hand, automatically and transparently switches between different techniques depending on the current task. *LaViola et al.* based this approach on their observation that manipulation is based on a repeating task sequence: objects have to be selected before they can be manipulated, thus the techniques for selection and manipulation may differ and an automatic switch between the different parts of the process is possible.

*Bowman et al.* developed an integrated technique called HOMER, which stands for hand-centered object manipulation extending ray-casting and uses both ray-casting and a virtual hand [BH97]. The user selects an object using ray-casting. Upon selection, the user's virtual hand is immediately teleported to the object and the user can then use a simple virtual hand technique for manipulation. The manipulation range of the virtual hand is scaled based on the position of the object and the user's real hand at the moment of selection. This means that the user can easily reposition the object anywhere between its original distance and themselves no matter how far away it initially was, but that the maximum distance the object can be moved to is limited by the scale which was defined during its selection. This results in problems when the user moves a distant object closer to them and wants to move it back to its original position. Due to the scale being set during selection, the user would have to either reselect and move the object many times or use another manipulation technique to move it.

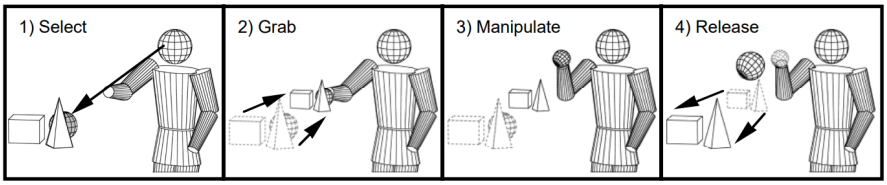



Figure 5.10: A series of pictures demonstrating the scaled-world grab for manipulation as described by *Mine et al.* [MBS97]

The scaled-world grab, another integrated hybrid technique designed to combine a long reach selection technique with grasping manipulation, was developed by *Mine et al.* [MBS97]. The user can select an object using an image plane technique (as discussed in the section on pointing-based techniques) but instead of the object's absolute size being scaled down as it is moved closer to the user, the entire virtual environment is scaled down to move the object within arm's reach of the user. The user can then use a simple virtual hand technique to manipulate the object until they release the selection at which point the environment is scaled back to its original dimensions. One advantage of this technique is that if the center of the scaling operation is located at the midway point between the user's eyes, the user will not notice the world has been scaled as long as they do not move their head, as small head movements may now enable the user to see around objects which were previously too big or may result in small movements moving seemingly larger distances than they should in an environment of normal scale. While this technique may seem even more natural for the user than HOMER, it shares the same weakness of making it hard for the user to move a nearby object further away from them.

### 5.1.2 Visual Selection And Manipulation Aids

A topic which is rarely explicitly discussed in research but which is often extremely important for usability are ways in which the interface aids the user in making a selection or improves the user's accuracy during selection. Usually, this is achieved through visual feedback or widgets which may vary depending on the application and environment.

#### Visual Selection Aids

The most common form of visual aid in the selection process is feedback whether an object has been selected and which object has been selected. While the confirmation of a selection is usually done through highlighting the selected object, some applications,

usually those using a grasping technique, also give feedback on whether or not a selection has happened by having a marker appear on the user's virtual hand if it has selected an object. An example of an application which uses both highlighting and a selection marker is the Meta 2 AR headset [Met17b] which displays a blue circle on the user's hand when the user selects an object, as well as highlighting the selected object by rendering a faintly glowing white outline around it. Highlighting is especially useful in menus (which will be discussed in section 5.3.1) which often use glowing outlines to highlight the currently selected menu item. An example of this is the SteamVR application which uses a ray-casting technique to select menu items [Val18]. If the user points the ray at a menu item, the item slightly increases in size, a glowing blue outline is added to it and if the menu item is a button with text on it, its color is changed to a significantly brighter blue (see figure 5.3). Highlighting may also take the form of 3D widgets appearing around an object signifying its selection. An example of this is found in the original implementation of the Go-Go technique [Pou+96] which displays a boundary box around selected objects, which was intended by the authors to minimize overshoot and provide visual feedback of the selection.

#### Visual Manipulation Aids

The most common form of visual aids during manipulation are widgets. While these are often used for manipulation by themselves, there are visual improvements which can be made to them which do not directly affect their function but increase usability. One of the simplest improvements to implement is color-coding positioning widgets. Often, positioning widgets consist of simple arrows which can be used to move the object. 3D modeling applications such as the *Blender* [Ble18] suite often color code the different axes of manipulation, mapping the axes X, Y and Z to the RGB color scheme, thus coloring them red, green and blue respectively. Coloring manipulation widgets with the color of their respective axis makes it easy for the user to discern at a glance which way an object is currently orientated and which widget or part of a widget moves the object along which axis.

## 5.2 Travel And Wayfinding

Moving between different locations is necessary for the functionality of almost every virtual and augmented reality application. Knowing this, it is obvious that wayfinding, the act of orienting oneself in the virtual (or partially virtual) environment and becoming aware of objects in it is both a fundamental task and closely linked to travel.

### 5.2.1 Travel

This section presents different travel techniques sorted into groups based on the metaphor used in their implementation.

#### Walking

While walking is by far the most natural traveling technique, it is an invisible technique with few to no visual elements depending on the implementation. Thus this section will only briefly list the possible implementations and discuss their visual elements. *LaViola et al.* [LaV+17] differentiate between three categories of walking techniques based on the human gait. Full gait techniques simply have the user walk normally and map their movements to travel in the environment. Partial gait techniques only use a part of the normal walking motion and gait negation techniques eliminate the forward movement of the user's gait.

**Full Gait Techniques** The simplest full gait technique is to simply track the user's position as they walk around in the real world and move their position in the virtual environment the exact same way. This technique has no visible elements but is the most simple and natural for users to use.

If the virtual environment's size is larger then the environment the user can move in in the real world one option is to redirect the user's walking in a way that is imperceptible to them by subtly altering the direction of their travel in the virtual world so they adjust their real world movement in a way that keeps them within the boundaries of the real environment [Raz+02]. One method to accomplish this is the stop-and-go method which *Bruder et al.* [BLS15] describe having the virtual world subtly turn faster or slower then the user if the user is stationary and physically turning. This allows the technique to direct the user away from the boundaries of the real environment as whenever they try to turn towards the real boundaries they are redirected away from them. Another option they describe is continuously rotating the virtual world as the user moves to turn their forward movement into a circular trajectory. *Bruder et al.* note that if the adjustments made through rotations are small enough they will be imperceptible to the user. Studies conducted by *Steinicke et al.* [Ste+10] found that a straight path in the virtual environment could be turned into a circle with a radius of at least 22 meters without the user noticing, meaning that a 40 meters by 40 meters space in the real world can be used to have users move along an infinite straight path in the virtual environment. *Bruder et al.*, however, estimate the area to be closer to 45 meters by 45 meters [BLS15]. *Steinicke et al.* also found that if the user physically turned the environment could turn up to 49% less than or up to 20% more than they physically

turned and that they could be moved up to 26% further or up to 14% nearer than they physically moved. They did note however that if a virtual environment was modeled after a real environment familiar to the user or contained many cues which helped the user judge distances such as trees or buildings, users were better at estimating distances and thus had a higher detection threshold for these redirections. Another possible way to have the dimensions of the virtual environment exceed those of the real environment is to have different rooms of a virtual environment overlap in space, but to only display the room the user is currently inside of. This technique which is called impossible spaces was devised by *Suma et al.* [Sum+12]. They found that small rooms could overlap by up to 56% and large rooms by upt to 31% percent before users noticed that the room layout overlapped and they were inside an impossible space. *Vasylevska et al.* [Vas+13] employ a similar approach in their flexible spaces technique, which is based on the concept of user's change blindness regarding the layout of a virtual environment and impossible spaces. Their technique procedurally generates virtual corridors between rooms which direct the user away from the real environment's center when they exit a room and then back towards the center when they enter the next room, which may overlap adjacent rooms.

Another possible approach is to simply map the user's real movements to their virtual movements in such a way that one step in the real world equals several steps in the virtual environment. The simplest approach, simply scaling up all horizontal movement of the user's tracked headset was found by *Interrante et al.* [IRA07] to induce cybersickness due to small swaying movements of the user's head being amplified into large back-and-forth movements of the viewpoint. To combat this problem, they devised the seven league boots technique which amplifies only movements made in the direction of the user's movement, which is determined based on the direction of their gaze and their movement in the previous few seconds. The scaling factor is increased if the user is walking faster and decreased if they walk slower.

**Partial Gait Techniques** Partial gait techniques have the user perform a motion that is very similar to real walking but does not move them in the real world. The simplest such technique is simply walking in place, moving one's feet up and down as if one were walking but without moving forward. The user is then simply moved, as long as they simulate walking, in the direction they simulate walking in, which can be determined from their gaze or through a variety of tracking hardware determining their travel direction from the position and movement of several parts of their body [LaV+17]. Another possible way to implement a partial gait technique is to simply move the user in a certain direction if they step away from their starting position in that direction and stop moving them if they return to the starting position. This might be done through

a platform with pressure sensors in the rim, such as *Wells et al.*'s Virtual Motion Controller [WPA96] or through measuring the horizontal distance of a tracking headset from the center of the tracking space such as in *McMahan et al.*'s work [McM+12].

**Gait Negation Techniques** Gait negation techniques are based on using specialized hardware such as treadmills or low friction surfaces to negate any directional movement while also providing a sensation as close as possible to real walking to the user [LaV+17]. They do not however, have any visual effect on the simulation or any necessary visual interface components and are thus not relevant to this work.

### Steering

*LaViola et al.* [LaV+17] define steering techniques as a set of virtual techniques in which the user is constantly exerting control over their own motion by specifying either an absolute or relative direction of travel. They categorize steering techniques into two groups, those that use spatial interactions to specify the direction of travel and those that use physical props. This section discusses a variety of steering techniques but focuses on those which have visible virtual elements or for which visible aspects have been researched.

**Spatial Steering Techniques** The simplest way to steer is simply looking in the direction one wants to travel in. This technique, called gaze-directed by *M. Mine* [Min95b], simply moves the user in the direction of their gaze, which is usually determined using a tracked headset or a vector from the virtual camera to the center of the viewport. This movement is actively triggered by the user either through a button on a controller or another form of trigger, such as gestures or commands. It may happen at a preset velocity or one the user can set using a system control technique. *Chee et al.* [CH02] expanded upon this simple technique by adding strafing to it, allowing the user to not only move in the direction of their gaze but also orthogonal to it. While gaze-directed steering is simple to understand and easy to learn, it has the disadvantage of being imprecise if one wants to move on a level plane, as the user may accidentally travel slightly upwards or downwards instead of straight forward. The technique also forces the user to always look in their direction of travel which means they cannot look around to orient themselves while traveling.

Another technique *Mine* describes in his work is steering by pointing [Min95b]. This technique uses the orientation of the user's hand or a hand-held controller to define the direction of travel. Apart from how the travel direction is determined, this technique works the same way as gaze-directed steering does but outperforms it in situations where motion relative to an object is required [BKH97]. *Mine* also describes a way to

specify the direction of travel which uses both the user's gaze and their hand and which he intended to be easier to use for novice users, the crosshairs technique [Min95b]. The crosshairs technique displays a virtual crosshairs at the position of the user's real hand or hand-held controller which the user then moves until the desired location is visible within the crosshairs. The direction of travel is defined by the vector originating from the user's head and passing through the middle of the crosshairs. *Mine* notes that while this technique is simpler to use than the pointing technique it may also lead to arm fatigue faster as the user has to keep their hand in line with the target location. *Mine et al.* [MBS97] expanded upon steering by pointing by developing a technique which uses both hands to define the travel vector. The vector originates from the user's non-dominant hand and passes through their dominant hand. This allows for travel in all directions without the user having to contort their hand or having to physically turn, as even backwards movement can be accomplished simply by extending the non-dominant hand further out than the dominant one. As an additional advantage, the travel speed can be controlled by the user by adjusting the distance between their hands, increasing when the distance is increased and decreasing when it is decreased. *Bowman et al.* [Bow+99] note that steering by pointing greatly aids in wayfinding as the user may look in any direction while traveling.

Using the direction the user's torso is facing as the direction of travel is another possible way to give the user the ability to look around during travel [LaV+17]. However, unlike steering by pointing, steering using the direction of the torso also keeps the user's hands free during travel. *LaViola et al.* [LaV+17] note that this technique has the advantage of potentially being even easier to use than gaze-directed steering but has the disadvantage of only being suited to horizontal travel and requiring an additional tracking device to be attached to the user's torso. A very similar technique to torso-directed steering or the partial gait technique developed by *McMahan et al.* [McM+12] is steering by leaning in the direction of travel [LaV+17]. A very simple implementation of this technique was developed by *Von Kapri et al.* [KRF11] who called it the PenguFly technique. The PenguFly technique tracks both of the user's hands and their head. To obtain the direction of travel, the position of the user's hands and head are projected onto the horizontal plane and the vectors between the projection of each hand and the head are added together with the resulting vector being the direction of travel. The travel speed is defined as the quadratic function $v_{max} \cdot (\|\vec{d}\| / x_{max})^2$ where $v_{max}$ is the maximum velocity, $\vec{d}$ is the direction of travel and $x_{max}$ is a normalization constant which is determined individually for each user from their arm length. *Von Kapri et al.* found that this technique was more accurate than pointing due to finer control over travel speed but that it increased cybersickness. Another disadvantage of steering by leaning is that it is limited to horizontal travel due to the position of the user's hands and head having to be projected into the horizontal plane due to the human arms not

being able to freely rotate in any direction. *Kruijff et al.* [Kru+16] found that adding visual and sound cues related to walking such as head-bobbing camera motions and footstep sounds greatly increased the immersion of the user and aided in inducing vection.

**Physical Steering Techniques** Similar to gait-negation techniques, physical steering techniques are based mainly on specialized hardware being used to interact with the simulation. *LaViola et al.* [LaV+17] identify two types of physical steering props techniques: The first type are props which imitate cockpits. These may be as simple as a car's steering wheel or as complex as a replica of a plane's cockpit. The second type of props, cycles, are used in cases where physical exertion is desired but walking is unnecessary. They usually consist of a setup similar to an exercise bike or a stationary unicycle-like setup. The user pedals to initiate travel and stops pedaling to stop travel. Most setups base the speed of travel on the speed of the user pedaling, many also support changing the direction of travel by using handlebars or simply leaning in the direction of travel.

### Selection-Based travel

Selection-based travel techniques are defined by *LaViola et al.* as the set of techniques in which the user selects either a target to travel to or route to travel along [LaV+17]. These techniques greatly simplify travel, as they usually do not require any continuous input or cognitive effort on the part of the user during travel.


Figure 5.11: A screenshot from *Epic Games*' game *Bullet Train* showing the player using ray-casting to select a location to teleport to [Sci15]

**Target-Based Travel Techniques** Target-based techniques are based on the concept of the user only wanting to reach a target location instead of wanting to travel along any specific route or wanting to have any input on their travel. *Bowman et al.* [BKH97] found that while the route may be unimportant to the user, instantaneously teleporting the user to a target location may disorient the user, making continuous movement a better choice if the user's spatial orientation is important. However, *LaViola et al.* [LaV+17] note that continuous movement outside of the user's control may increase cybersickness. Based on this, they suggest the use of extremely fast movement to the target to minimize time spent moving, which, according to *Bowman et al.*'s findings, is less disorienting than teleportation.

*LaViola et al.* [LaV+17] differentiate between several methods to select a target for travel, some of which are using techniques which are discussed in detail in other chapters of this work. These methods include: Selecting an object at the target location or the location itself using one of the selection techniques discussed in chapter 5.1.1, as seen in *Epic Games*' Virtual Reality game *Bullet Train* in which the player can use ray-casting to select the first location intersecting with the ray, which is displayed as a blue laser, and teleport to it (see figure 5.11) [Sci15]. Using manipulation techniques from chapter 5.1.1 to place an object at the target location, as seen in the game *Budget Cuts* by *Neat Corporation* in which the player can use a ball gun held in their virtual hand to shoot a ball at a target location and then teleport to the ball by pushing a button [Lan16]. Selecting a target location from a list or inputting a name or coordinates of a location using one of the menu solutions discussed in section 5.3.1, as seen in the Virtual Reality reconstruction of the legion camp *Vindonissa* in which the player can use a small menu to select and teleport to locations in the camp [Chr17]. This leaves two techniques to discuss in this section, representation-based target techniques and dual target techniques.

In representation-based target techniques the user specifies their target location using either a map or a 3D model such as the world in miniature discussed in section 5.1.1. A simple implementation is found in *Bowman et al.*'s work, who use a tracked stylus and tablet to implement traveling [Bow+98]. The virtual representation of the tablet displays a map on which a red dot represents the user's current position. Placing the stylus on top of the red dot and pressing a button allows the user to drag the dot across the map to a new position. Once the user lets go of the button, they are smoothly moved to the new location of the dot. *Bowman et al.* note that this technique has several advantages: the map serves as a wayfinding tool by itself (see section 5.2.2), the user can quickly move to any target location without having to navigate the environment and, as the user is only moved once they let go of the dot, they can look around and gain spatial knowledge during travel.

*LaViola et al.* [LaV+17] define dual-target techniques as target-based techniques which

are designed to move the user between two target locations. The ZoomBack technique developed by *Zeleznik et al.* is one example of such a technique [Zel+02]. It uses pop-through buttons (which can differentiate between light and firm presses) and a tracked controller which is either hand-held or attached to the user's hand as input. The user simply targets a location to travel to using a virtual ray which is continuously and visibly emitted from the controller. Once they are pointing at the desired location, they lightly press the button on their controller and are moved about two feet in front of the location in about two seconds. If they immediately release the button, they are moved back to their original location at the same speed. If, instead, they press the button firmly before releasing it, they remain at their new location. *Zeleznik et al.* found that users had little trouble adapting to this technique and found it natural and efficient to use.

**Route-Planning Travel Techniques** If no continuous input is desired or possible during travel, but it is important for the user to choose the route taken, route-planning techniques, which allow the user to specify the route taken to a target themselves, are the best choice. Route-planning techniques allow the user to plan the route before it is executed, which allows more precise routes than other traveling techniques and gives the user time to refine the route before traveling. Due to the user being moved along the planned route automatically, they also allow the user to focus on looking around or completing other tasks, such as manipulation, during travel. *LaViola et al.* [LaV+17] identify two ways to plan a route, drawing a path and marking points along a path.

One of the first techniques which allowed the user to draw a path to follow in a 3D environment was developed by *Igarashi et al.* [Iga+98] who developed it for desktop 3D user interfaces. It allows the user to draw a path to follow directly into the environment they see on the screen, which is then displayed as a large black line. They can choose to draw a short path near their target position and are then moved there automatically, similar to target-based traveling, they can choose to draw a short path at their feet to turn, and they can choose to draw a long path from their location to the target location. If they draw a long path, it is projected onto the walking surface in the environment and followed until the user reaches the target location. A unique feature of the technique is its ability to automatically avoid obstacles, climb slopes, and to pass through gates and tunnels even if the route was drawn over the gate or on the roof of the tunnel. *LaViola et al* [LaV+17] note that in virtual reality drawing onto the environment from the current to the target location usually isn't possible and thus suggest the use of a map or 3D model onto which the path can be drawn.

Marking points along a path is often simpler to use than drawing a path depending on the complexity of the path and the input method. This method lets the user

specify certain key points in the environment through which the application then plots a route along which the user is moved. *Bowman et al.* [Bow+99] created a simple implementation in which the user places markers on a 3D map showing the environment. The application then simply moves the user in a straight line from one marker to the next one until they arrive at the last marker. The advantage of this technique is the ability to exert very fine grained control and create very complex routes simply by placing more control points while also allowing the user to cede control to the system for less complex routes by placing fewer markers.

#### Manipulation-Based Travel

Manipulation-based travel techniques use manipulation techniques, such as those from chapter 5.1.1, to move either the viewpoint or the world itself until the user arrives at the desired location. *LaViola et al.* recommend for these techniques to be used in scenarios in which manipulation and travel tasks are interspersed and frequent. They base this recommendation on the fact that if both manipulation and travel use the same metaphor and technique, switching between the tasks requires both less time and less cognitive effort, increasing usability and efficiency.

**Viewpoint Manipulation Techniques** One of the easiest ways to move the viewpoint is to simply attach the virtual camera to something the user can manipulate. *Ware et al.* developed an interface for a desktop virtual environment which they called the eyeball in hand technique [WO90]. The user holds in their hand a tracked controller and moves it through a defined area which represents the virtual environment which is shown on a separate display. The position and orientation of the tracker is mapped to the position and orientation of the virtual camera which allows the user to move the camera any way and anywhere they want simply by moving around the tracker. *Ware et al.* found the eyeball technique to be both easy to implement and easy to learn for users but noted that its perspective is limited by the reach of the user's arms and that it may be confusing or disorienting for some users due to the mismatch of a third-person view of the tracker and a first-person camera image.

Another option are avatar manipulation methods. An avatar manipulation method represents the user themselves with an avatar that they can move and rotate while the application maps these movements onto the viewpoint. One example is the world in miniature approach previously discussed in the section on selection and manipulation. In the original study [SCP95] an avatar representing the user was displayed in the miniature model of the world. Manipulating this avatar would either move the user's viewpoint along a route to their destination or instantly move their viewpoint to a new position. This technique has the advantage of combining a travel technique

with a wayfinding, and a manipulation technique, allowing the user to easily gain an overview of their environment and possible destinations at a glance, as well as seamlessly combining travel and object manipulation into one solution. A disadvantage unique to the use as a travel technique is the possible disorientation that occurs if the viewpoint is moved instantly to wherever the user places the avatar. While this can be counteracted by calculating or letting the user choose a route to the destination, some environments such as those containing enclosed rooms prevent these methods. *Pausch et al.* suggest that avoiding disorientation or cybersickness may be accomplished by moving the viewpoint down into the model at the avatar's new position in one continuous motion, enlarging the model until it replaces the environment and creating a new miniature [Pau+95].

*Pierce et al.*'s work on image plane techniques, previously discussed in section 5.1.1, also contains a travel technique which is an example of fixed-object manipulation [Pie+97]. *LaViola et al.* [LaV+17] define fixed-object manipulation-based travel as a set of techniques in which the user selects an object and performs the gestures of a manipulation method. Instead of moving the object, however, the object remains stationary and viewpoint is moved relative to the object. A simple example would be a user performing the hand movements to move the object closer to themselves. Instead of moving the object to them, they move closer to the object. If they performed the gestures to rotate the object, they would be rotated around the object instead. The use of image-plane techniques by *Pierce et al.* has the advantage that the object does not have to be within the user's reach, they may simply use image-plane selection to select any object visible to them and use it to travel. Fixed-object manipulation-based travel has the advantage of using the exact same set of techniques and movements for both travel and object manipulation, requiring only the push of a button or another similar trigger by the user to switch between the travel and manipulation contexts. One disadvantage of this technique is that the user may become confused as to which mode is currently active. This may be avoided either if travel mode is invoked by holding down a button and the user leaves travel mode simply by releasing it, or if a visual indicator of the currently active mode is displayed within the user's view.

**World Manipulation Techniques** An alternative to moving the viewpoint around a fixed world is moving the world around a fixed viewpoint. A simple implementation called the scene in hand technique is found in *Ware et al.*'s work [WO90]. As opposed to the eyeball in hand metaphor which moves the viewpoint when the user's hand is moved, the scene in hand metaphor moves the scene around the viewpoint when the user's hand is moved. *LaViola et al.* [LaV+17] note that the easiest way for a user to invoke this technique would be to use a grasping technique. This would only require

the application to differentiate between the user grasping an object, which would then invoke object manipulation, or the user grasping empty space, which would then invoke traveling by manipulating the world. They also recommend that only the translations but not the rotations of the user's hand are used to manipulate the world as otherwise the user might accidentally rotate the world, which might lead to disorientation. While this technique has the advantage of seamlessly combining object manipulation and travel without even needing to actively switch between contexts, it also requires many large arm movements to reach distant locations or to navigate complex environments. One way to remedy this strain on the user's dominant arm is to simply spread it between both of their arms. *Mapes et al.* [MM95] developed a technique which uses both of the user's hand to move the world, in a gesture similar to pulling a rope. The user simply performed a grab gesture in empty space then pulled their hand towards them, then repeats this process while alternating between their hands, each pull moving them forward. This technique also allows the user to change view direction while traveling by simply grabbing the world with both hands and rotating their dominant hand around their non-dominant hand, which is then mapped to a rotation of the world around the user.

### 5.2.2 Wayfinding

While travel is often a very interactive task, wayfinding usually happens in the user's head, a process that can be greatly aided through several different techniques discussed in this section.

#### Environment Legibility

*LaViola et al.* [LaV+17] note that the techniques for environment legibility which were designed by *K. Lynch* [LJ60] as urban design principles also apply to designing virtual environments. *Lynch* found that an environment was easier to understand and navigate in if its basic elements were understood. Five elements of environments were described in his work: Paths, such as actual paths or streets, are designed to support linear movement through the environment. Edges, such as walls, rivers or even large buildings, make up the borders of the environment and block any movement. Districts are uniquely identifiable areas which are marked with recognizable visual elements, such as style or color. Nodes are points where people will gather such as intersections of major paths, district entrances, central spaces such as plazas, or public buildings. Landmarks, which will be discussed separately in this section, are easily recognizable static objects which aid the user in recognizing a specific place and are often located near nodes. *N. Vinson* recommends that a virtual environment should contain all five

of *Lynch*'s basic elements as they each support wayfinding in a different way [Vin99].

### Virtual Landmarks

A landmark is an object that is easily distinguished from the rest of the environment, whether through its size or by simply being visually distinct from other objects. *LaViola et al.* [LaV+17] note that landmarks help a user maintain their spatial orientation, develop landmark and route knowledge, and serve as a foundation for distance and direction estimation. *N. Vinson* [Vin99] developed a set of guidelines to aid in the use of landmarks. His recommendations include the use of concrete objects and buildings instead of abstract constructs or shapes (such as a position marker or guide arrow) as landmarks to increase memorability, the recommendation that a landmark should be as distinctive from other landmarks, objects, and buildings as possible as well as a recommendation that all sides of a landmark can be easily distinguished from one another making it easier for a user to use it for position and direction estimation. Additionally, he recommends that landmarks are placed near major paths and at intersections, making it easier for users traveling along those paths to navigate, making the landmark easier to spot from a path, and increasing the memorability of landmarks. Landmarks can be of a global or local nature: a global landmark is visible at all times and provides the user with directional information while a local landmark is visible only if the user is close to it and aids in route navigation. *Steck et al.* [SM00] found that many users use both global and local landmarks to navigate, choosing one depending on visibility, amount of information provided and memorability. Some users however only use one type of landmark, either global or local, if both types are provided. While *Steck et al.* found that these users would navigate using their non-preferred type of landmark if the preferred type wasn't available, this shows the importance of including both types of landmarks in a virtual environment to support different the wayfinding strategies present amongst different users.

### Maps

One of the oldest and most common wayfinding aids, the map, is most often found in applications with a large virtual environment the user is free to roam through. While the concept predates digital technology, computers and virtual and augmented reality in particular nevertheless offer some new techniques not found in real maps but also add several special requirements. *LaViola et al.* [LaV+17] list several of these techniques such as the ability to dynamically change maps on the fly to account for changes in the environment, the ability to continuously display a location marker for the user, the ability to display paths to the next destination on the map, the ability to rotate the


Figure 5.12: A screenshot from the game *The Well* showing a map being used to navigate a VR environment [Ham17]. It is invoked through a menu option and features both a marker for the user's location and a marker for target locations.

map to show the direction the user is facing, or the ability to zoom in on the user's current location. In addition to the techniques noted by *LaViola et al.* it is important to note that the definition of a map can be expanded in three-dimensional environments to include 3D models serving as maps, which allows for the display of much more complex structures, such as showing several floors of a building and the area around it on the same map instead of using one two-dimensional map for each floor. *LaViola et al.* advise that clutter is to be avoided as an overly complicated map interface may lead to confusion with users. They recommend the usage of a marker for the user's location and for the map to be always up-to-date. The research of *Darken et al.* [DC99] suggests that the orientation of the map may strongly affect the performance of users in wayfinding tasks. They found that a map that rotates to always show the user's view direction as "up" on the map appear to lead to better performance in search tasks in which the target's location is shown on the map, while maps that always show north as "up" appear to lead to better performance in tasks where the location of the target is either known, but not shown on the map, or unknown.

*LaViola et al.* [LaV+17] note that it is often not feasible to keep the map displayed for the user at all times, as it may take up a large amount of the screen and thus obscure the user's view of the environment, potentially even hindering wayfinding by obstructing other wayfinding aids in the environment. It is recommended to either allow the user

to hide or display the map at will through some mechanism, place the map in a menu to be accessed by the user (such as in figure 5.12) or attach the map to the user's hand so it can be put away by simply lowering the hand it is attached to. These techniques may also be combined such as in the game *Rogue VR* [San16] in which the map is displayed if the user looks down at their hands while in the inventory screen.


Figure 5.13: A screenshot from the game *VTOL VR* which shows the virtual horizon indicator aligned with the horizon in the middle of the planes head-up-display [BV18].

### Compasses

A compass provides the same information in a virtual environment as it does in the real world, indicating the cardinal directions to the user. *LaViola et al.* [LaV+17] call it an invaluable wayfinding tool if it is used in conjunction with a map but note that some users may be not be able to effectively use only a compass and a map to navigate. *Sherman et al.*'s definition of compasses, on the other hand, includes any kind of UI element or instrument which indicates a direction [SC03], such as the horizon indicator on the virtual instrument panel of a plane in a flight simulator (see figure 5.13) or a combination of a cardinal direction and horizon indicator.

### Instrument Guidance

*Sherman et al.* [SC03] base their definition of instrument guidance on the real world instruments found in planes or ships, in which displays, dials, or indicators aid both in finding a target location by indicating its direction or a path to it, and in not straying off course by indicating orientation or adjustments needed to return to the correct orientation and path. In virtual environments these instruments are more diverse. One way is to simply display a virtual cockpit and all its instruments, a method which is commonly used in flight simulators. An example of this is found in the game *VTOL VR* [BV18] in which the user is placed in a simulated cockpit of a plane and has all instruments found in a real plane at their disposal to aid in navigation. However, according to *Sherman et al.*, virtual instruments do not need to be simulated versions of real-world instruments. They may take the form of interface elements such as an arrow appearing over the location the user is being guided to, or visual indicators when they veer off course. A simple example of this are the transparent walls and outlines of the real world displayed by the HTC Vive's chaperone system when the user nears the borders of their tracked environment, designed to remind the user to either turn around or move backwards lest they collide with objects in the real world [Gep17]. Another example is found in the game *Serious Sam VR*, in which taking damage from enemies which are outside the player's field of view results in a red arrow being displayed pointing in the direction the player needs to look to see that enemy [CVD17].

### Memorable Placenames

Assigning a name to a place or landmark makes it easier for users to memorize their location as they now have a name to associate it with. Placenames are also an important component of many maps, making it easier for a user to search for a location on the map as they only have to scan a number of placenames and find the one they are looking for instead of having to build a mental model of the location and how it would be displayed on the map and searching the map for it. They also enable selection-based travel techniques which simply use a menu displaying a list of place names the user can choose from. *Sherman et al.* suggest that an additional advantage to assigning a memorable placename to a distinguishable location is making the location itself a landmark, even if no other global landmarks are present at the location [SC03].

### Marking A Path

The easiest way to make sure the user is always aware of the correct path is to display that path, either through markers or a path in the environment, or on a map. *Sherman et al.* [SC03] make several suggestions as to how such a path may be marked in the

environment, such as signposts along the path with arrows on the signs leading to the next one or a colored line on the ground which the user needs to follow.

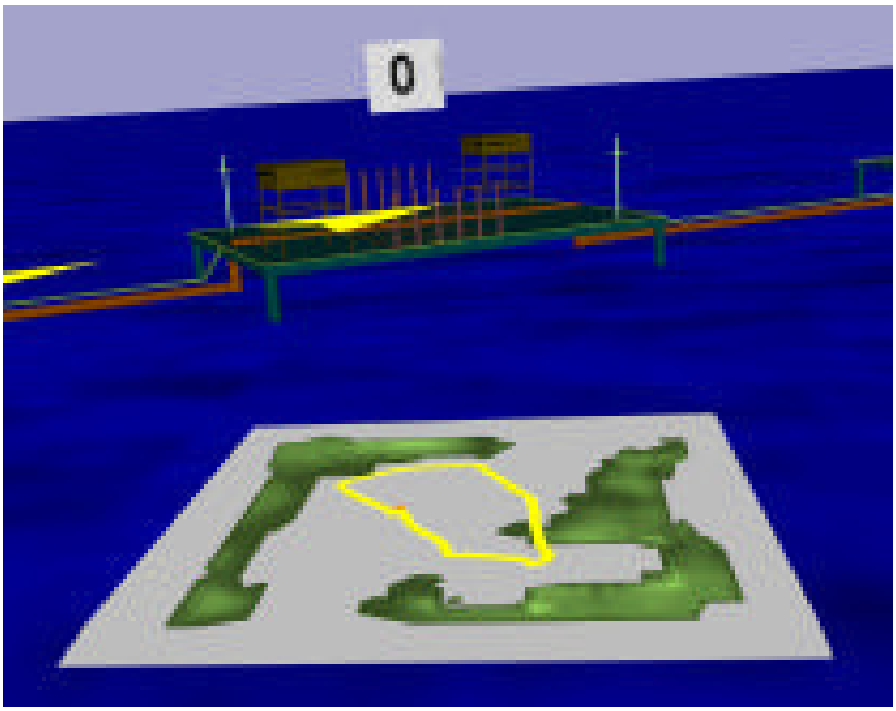



Figure 5.14: A screenshot showing a trail marking the user's path through the environment on a map [DP02]

### Leaving A Trail

The Hansel and Gretel technique [DS93] is based on the idea of the user leaving behind a visible trail or trail of markers when moving through the environment, similar to the breadcrumbs left by Hansel and Gretel in the famous fairy tale. Marking the path the user has taken helps the user retrace their steps through the environment and aids navigation when returning to previously visited locations. *Darken et al.* [DP02] added directional markers to the trail, based on the concept of footprints, which they thought would be a useful addition to the basic trail. They found that during an exhaustive search of the environment users mostly ignored the directional markers due to the environment becoming to cluttered with footprints. Most users also ignored the trail markers placed in the environment in favor of the trail being marked on a virtual map they could invoke. *Grammenos et al.* [Gra+02] suggest several solutions to the problem of

clutter due to too many markers: A filtering system which lets the user select different attributes of markers such as their creator, or their time of creation and then displays only the markers with the desired attributes. Time-sensitive markers which disappear after a certain, selectable amount of time has passed and can then only be displayed using the filtering system. Connecting lines between markers disambiguating which trail a marker belongs to. A list, which lets the user select the desired marker based on its attributes if multiple markers overlap. They also suggest several new uses for such trails in multi-user environments, such as locating other users through their trails, visualizing which locations are often visited by users through the number of footprints at the location, or virtual tours by following the markers.

### Switching Between Exocentric And Egocentric Viewpoints

*Sherman et al.* describe the switch between an egocentric and an exocentric viewpoint as a useful wayfinding aid [SC03], allowing a user to orient themselves in a wider perspective such as third-person or even a world-in-miniature view while also using the benefits of the egocentric perspective such as increased immersion and an improved ability to see small details of objects up close. There are several different ways to implement such a switch, each providing different advantages and disadvantages.

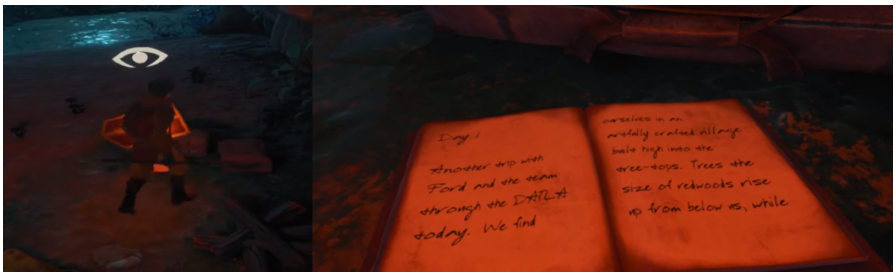


Figure 5.15: Screenshots from the game *Chronos* showing the switch between exocentric and egocentric viewpoints [agu16]

The virtual reality game *Chronos* [Gun16] is generally played in a third-person perspective. The player overlooks the current area the using the head-tracking functionality of the Oculus Rift HMD to move and rotate the camera and using a controller to control the character. However, in many instances the player is given the opportunity to switch to an egocentric perspective to interact with an object, read text on a virtual screen or book, or inspect an object. Figure 5.15 shows such an interaction: the icon appearing above the player character shows that the action to inspect an object is available, if the action is prompted by the user through pressing a button, the game shifts to an ego-centric perspective. This technique has the advantage of providing a view of almost

the entire room the player is located in. If one is trying to find the way forward or trying to accomplish tasks or puzzles in the environment this perspective provides an easy way for the application to supply the user with spatial information about the environment all the time, they can simply see it. Conversely, if small details or a smaller-scale interaction is needed the shift into an egocentric perspective allows for the user to see and act in a much greater level of detail. The disadvantage of this technique lies mainly in the fact that the majority of time is spent in a third-person perspective. As the player character moves from room to room, it is often necessary to change the position of the viewpoint in relation to the character between rooms leading to a need to reorient oneself when switching between rooms. Additionally, tracking the character as they are moved around the room leads to repeated left-to-right and right-to-left movement of the head, which is straining if the game is played in longer sessions [Pol16]. The third-person perspective introduces certain constraints on environment design, requiring enough space for a camera to view both the character and the entire room as well as limiting the amount of detail visible to the player unless switched to the egocentric perspective. Finally, controlling a character indirectly may reduce immersion for some users.

Another possible way to easily provide the user with the advantages of a exocentric perspective, while remaining in an egocentric perspective is the world in miniature approach previously discussed in the sections about selection and manipulation, and travel (see sections 5.1.1 and 5.2.1) [SCP95]. In this approach a virtual miniature model of the current environment is given to the user, specifically altered through backface culling or similar techniques to make the entirety of the environment visible by hiding the walls in such a way that the user is always able to see into the room (see figure 5.7). Whenever the user needs to orient themselves in the environment they can simply raise the prop into their field of view and search for the indicator showing their location and the direction of their gaze and compare the information seen there to what that they see around them. This perspective provides similar advantages to all exocentric perspectives, the ability to easily discern their position in relation to the entire environment but also provides the unique advantage of being able to be used as both a manipulation and a travel technique as well. As mentioned before, the biggest disadvantage is the limited scaling of the model to larger environments, however complex environments, such as those containing several rooms or vertical structures may also provide a wayfinding challenge if too many or too few walls are hidden in the model.

### Coordinates Display

Coordinates, whether they are Cartesian coordinates based on an origin point in the environment or grid coordinates based on an orthogonal grid dividing the environment, are an easy and accurate way for a user to ascertain their current position. *Darken et al.* [DS93] describe the process of wayfinding using a coordinate system as a series of exploratory movements while observing changes in the displayed coordinates, noting that users would usually align their view direction with one of the axes of the coordinate system when moving. They found that coordinate systems were useful in aiding users to return to a previous location that they memorized the coordinates of, as well as Cartesian coordinates aiding in search task as they could be used to partition the environment into a grid. *Sherman et al.* [SC03] note that if memorable placenames are used for locations in the virtual environment, an interface element displaying coordinates may also be used to display the names of and distance to nearby notable locations.

### Constrained Movement

One of the simplest ways to prevent users from getting lost in the virtual environment is to simply constrain their movement so they are always on the correct path. This may take many forms, such as extremely linear environment design in which the user can only take one path or move down a series of rooms and hallways, small environments in which the user cannot move very far or simply not allowing travel in wrong directions by blocking the user when they move too far away from the intended path. Small environments can be designed to be congruent with the tracking range of a virtual reality headset which has the additional advantage of being able to represent barriers in the real world or the edge of the tracking range as the edge of the virtual environment. An example for such a small environment is the virtual reality game *Hot Dogs, Horseshoes & Hand Grenades* [RUS18], which simulates gun ranges on which the player can interact with virtual guns, explosives and other items. The area of the gun ranges in which the player can move around are usually surrounded by either tables full of items, fencing or walls, naturally restricting the players movement to an area about as big as the headsets tracking area. Another option to avoid the user getting lost is to simply use a traveling technique that disallows user input in regards to location or direction such as those that simply move the user along a predetermined path.

## 5.3 System Control And Menus

Every simulation has some aspects that need to be configured by the user from within the simulation but that cannot be represented by interaction with the environment. Whether it is graphical options, controller calibrations or item selection in games, these aspects are represented through menus, gestural or voice commands and grouped into the task of system control.

### 5.3.1 Menus

Menus are by far the most common technique for any kind of system control task and thus a great variety of techniques to implement them exist. However, their use is not limited to system control but may also include tasks such as travel or selection. This section discusses examples of menus in virtual and augmented reality grouped by their hierarchical nature and structural layout.

#### Temporary Option Menus

Temporary option menus are only displayed for a short time and vanish after the user has selected a menu item. They usually display less than 7 items, which are most often options. They are usually of either a list, ring, matrix, or geometric structure.

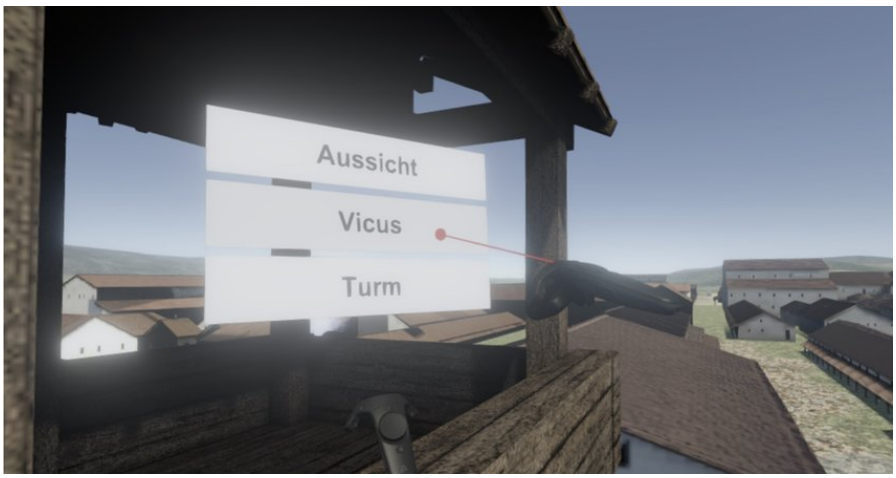



Figure 5.16: A screenshot from the virtual reality reconstruction of the legion camp *Vindonissa* showing a pop-up menu used in location selection [Chr17]

**List** Using a list-structure is one of the most simple ways to implement a temporary option menu. Common implementations of these list-structured menus are pull-down and pop-up menus. According to *Jacoby et al.* [JE92] a pop-up menu is a menu that appears at the cursor's (or a similar tool's) location, while a pull-down menu is pulled down from a menu bar. Another possible implementation of this menu is the look-at menu, which is an options bar displayed at the location of the user's gaze from which the user selects an item by looking at it and the pressing a button [MBS97]. One of the first uses of a pop-up menu is described by *Wloka et al.* in their work on a tool-based interaction technique for virtual reality called the virtual tricorder [WG95]. A virtual tool, mapped one-to-one to a real one in the user's hand, can be used to accomplish different tasks such as selection, manipulation, information visualization or navigation. To let the user switch between these tasks a list of them is displayed on top of the tool as a pop-up menu from which the user is able to select the desired task. *Mine et al.*'s work [MBS97] not only contains one of the first uses of look-at menus but also extends pull-down menus by hiding them in fixed locations relative to the user, outside their field of view, to avoid one of the biggest disadvantages of pull down menus: the need to either dedicate a physical button to invoke a menu bar or to have a menu bar always visible, increasing clutter. Using their technique allows the user to simply pull a menu from a known location. They note that using locations which are relative to the user's body increases recall and acquisition of frequently used controls. However, hiding menus outside the field of view of a user has the disadvantage of creating a need to teach the menus' location to the user instead of the user being able to find them themselves through exploration.

**Ring** One of the first applications using a ring-shape to display options was devised by *M. Mine* in 1995 for the ISAAC application, a virtual reality tool for construction of virtual worlds. To choose between different sets of tools, a ring-shaped menu attached to the users hand is used. After invoking the menu with the press of a button on the controller, the options are displayed around the users hand and they are able to rotate their hand to move the available options through a selection box in the direction of the rotation. When the button is depressed the tool inside the selection box is selected. The advantage of this solution lies in the speed at which a tool can be found and selected due to all tools being easily visible at once (see figure 5.17) which does however limit the amount of displayable options as adding too many items to the menu would make individual items too small to reliably see and select. A style of menu similar to the rotary tool chooser was developed by *Gerber et al.*, calling it a spin menu. [GB05]. They describe a menu appearing at the user's hand at the press of a button, with selections being cycled through a selection box depending on the direction the user rotates their

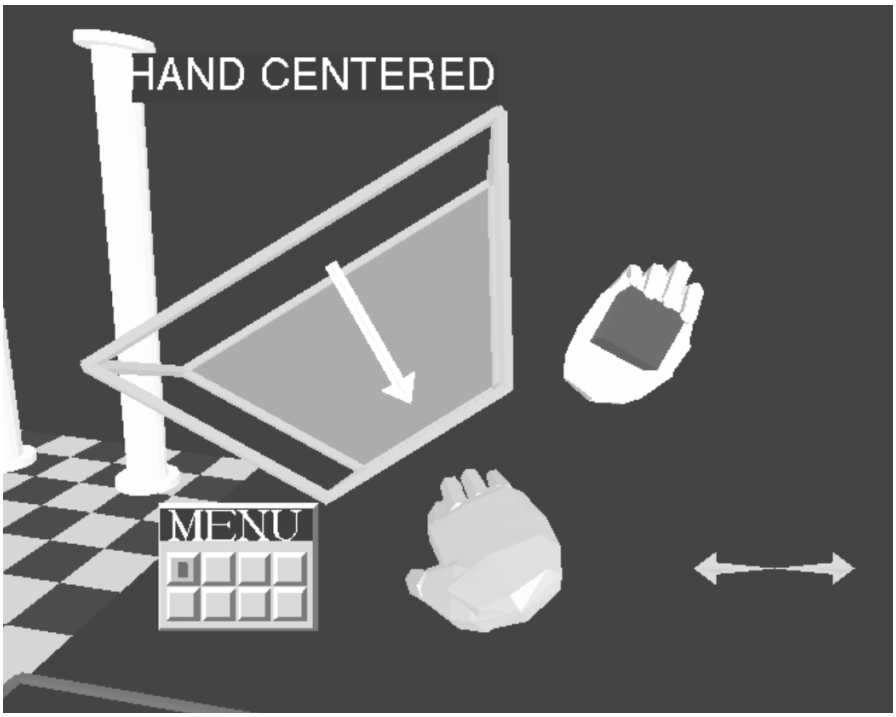



Figure 5.17: A screenshot from the *ISAAC* application showing the rotary tool chooser [Min95a]

hand in. Their design differs from the rotary tool chooser due to their addition of hierarchies to it. This extends the menu from a maximum of 9-11 displayed items to a much higher limit that is unlikely to be reached in practice without compromising usability. A spin menu may have several layers of submenus arranged in one of three different ways: crossed, in which a submenu is positioned orthogonally to the layer before it, concentric, where each layer wraps around the current, innermost one and stacked, where layers are placed in top of the previous one and the uppermost layer is the current one. If submenus are added to a spin menu, it is usually no longer considered a temporary option menu and instead becomes a menu system or hierarchy.

**Matrix** Originally designed for workbench environments, in which 3D objects appear to be floating above a table, the command and control cube devised by *Grosjean et al.* was designed to provide the functionality of hotkeys in three dimensions [GC01]. Whenever the user performs a pinching gesture on a 6-DOF tracked controller to invoke the menu, a spherical pointer appears inside the central cube of a 3x3x3 cubic structure divided into 27 smaller cubes called slots. The user then moves the pointer into one of

the 26 other cubes and stops the gesture to select the option associated with the chosen slot. As choosing a direction is more efficient than selecting an item [Cal+88], once a user has memorized the layout of the cube, this implementation is more efficient than other solutions while also being able to display up to 26 options (the central cube is reserved for canceling without choosing an option). Another advantage of the cube lies in the fact that it is designed to be used by the user's non-dominant hand while their dominant hand is used in other tasks. To aid in visibility, only the horizontal plane the sphere is currently located in is rendered, the other planes are displayed only when moved to. The main disadvantage of this solution lies in the fact that it forces a certain number of options on the menu as an asymmetrically filled cube would confuse the user.

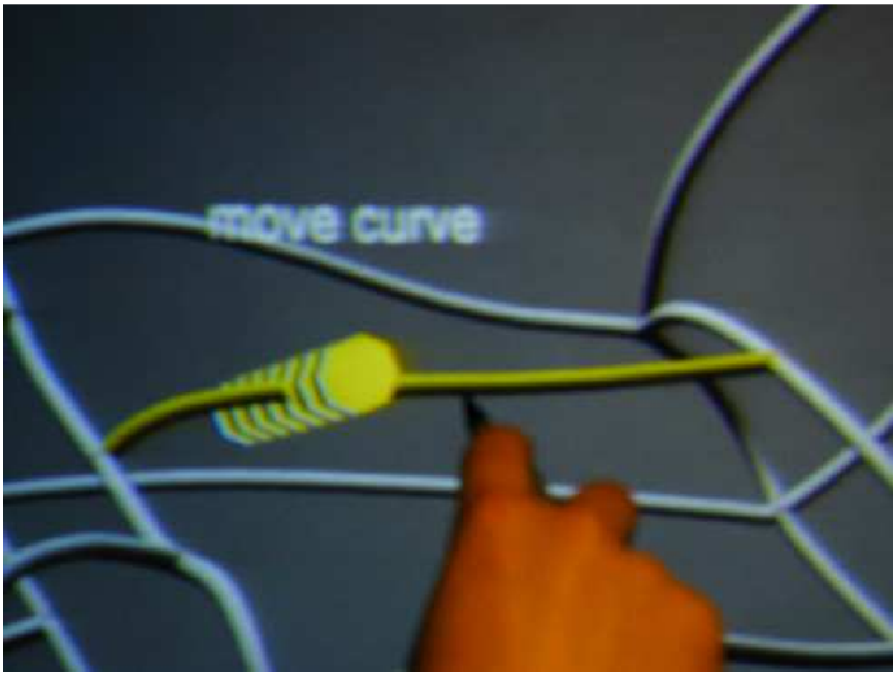



Figure 5.18: A screenshot showing the ToolFinger being used to manipulate a curve [Wes03]

**Geometric Structure** The ToolFinger designed by *G. Wesche* is a menu designed to let the user choose between different manipulation tools. It is of a cylindric shape and divided into smaller segments by gray lines. Though the first segment is shaped like a cone to give the ToolFinger the appearance of a pencil it is functionally identical to the other segments. The position of the Toolfinger in the virtual environment corresponds

to the position of a tracked stylus in the user's hand. Each segment on the ToolFinger corresponds to a tool such as manipulation or deletion and a tool is chosen by moving the finger until the corresponding segment is touching the desired object and pressing a button on the stylus. To aid the user in selecting the correct tool, a small line of text is shown near the finger which describes the tool currently intersecting the object (see figure 5.18) which also allows the user to explore all available tools simply by moving the stylus back and forth across the object. The main advantage of the ToolFinger is its integration of the tool selection directly into the tool requiring very little movement and use of only one hand to switch between different tools, improving usability and reducing the distraction caused by it. The main disadvantages are its reliance on extremely accurate controller tracking as inaccuracies may result in selecting the wrong tool, having to reselect the same tool when switching between objects and its inability to be used for general system control tasks aside from tool-switching.

### Single Menus

Single menus are similar to temporary option menus but may stay visible for a longer time or even be displayed at all times. They may display a high number of selectable items which may be comprised of any type of item that can be displayed in a menu.

**List** List-structured single menus are perhaps the most well-known and, according to *Dachselt et al.* [DH07], one of the most common menu solutions. A very popular implementation in both 2D and 3D is the drop down menu. Drop-down menus usually have a root that is always visible and which if invoked "drops down" a list of selectable items. They are often overlaid into the world as purely 2D objects such as in *Feiner et al.*'s work [Fei+93] as interacting with them in a 2-dimensional fashion often results in greater efficiency [DC99]. This may result in having to provide different input devices or methods for menus and interaction with the environment which may lead to losses in overall efficiency or usability depending on the complexity of switching between input methods. This effect, however, may be offset by the advantage of providing the user with a menu solution that is already familiar from past experiences in 2D, decreasing the time needed to learn about the interface. Alternatively, not all list-based interfaces are necessarily adapted directly from 2D. In the FingARtips application, an Augmented Reality urban planning application, the user can place streets and skyscrapers on a map [Buc+04]. To switch between different types of objects to be placed, the application always displays 3D models of the three types of skyscrapers, a street, and a button to switch to an immersive view in a row at the left side of the workspace (see figure 5.19). A menu item is selected by simply grabbing it if it is a 3D model, which will then place a duplicate of it into the user's hand, or pressing the image if the item is represented

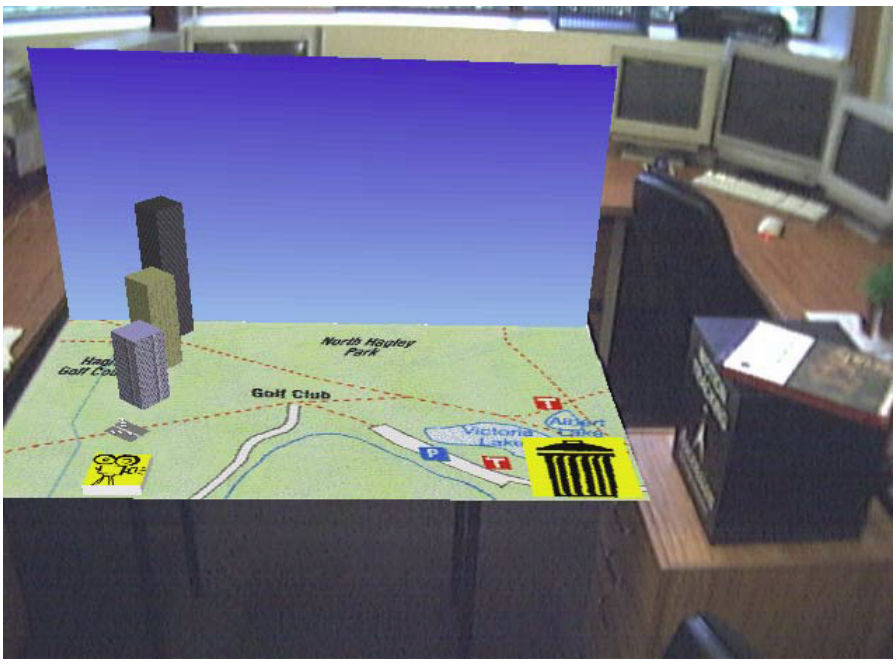



Figure 5.19: A screenshot showing the menu in the FingARtips application, consisting of three types of skyscrapers, a street and an immersive view button [Buc+04]

by an icon. This type of menu has the advantage of only requiring a single type of input method for interacting with the menu and the rest of the simulation, seamlessly integrating the menu into the environment. Its biggest disadvantage lies in the 3D models and buttons being a part of the workspace both reducing usable space and occluding objects behind the menu. Another possible implementation is the Tinmith system [PT02] in which a menu consisting of eight option is always visible at the bottom of the user's display as a transparent green bar. The menu options are assigned to each of the user's fingers and are selected by performing a pinching gesture with the finger corresponding to the desired option and the thumb. While a device-referenced menu may lead to visual clutter due to always taking up some part of the view, the Tinmith menu attempts to mitigate some of these drawbacks by being both transparent and able to be repositioned. Another option is to use a tracked pen and tablet as an input the device, mapping the real tablet to a virtual one on which menu items are displayed and can be selected with a pen. This may be used for tool selection or general system control tasks such as loading or deleting objects [Bil+97] or even to browse and start entire sub-applications [SG97]. While the method to interact with menus in an almost 2D way used in this method greatly increases the efficiency of that specific

interaction the biggest disadvantage of this method is its need for a specific input device which while well-suited to tasks requiring 2D interaction is rather unsuited for most other 3D interaction tasks such as selection of objects in the environment using any method other than a menu. Finally, *Poston et al.* suggest the ToolRack as a menu implementation which has its list-structure emulate real world objects [PS96]. The ToolRack is a virtual wooden rack on which several selectable icons are placed representing both the currently available selection of tools and basic system control actions such as an undo button. Its biggest advantage is its simplicity and familiarity, due to its resemblance to a real tool rack, making it easier for users to quickly grasp its function and use. This is a result of the design specifically being intended for users without experience in virtual and augmented reality that lack motivation to learn complex commands.

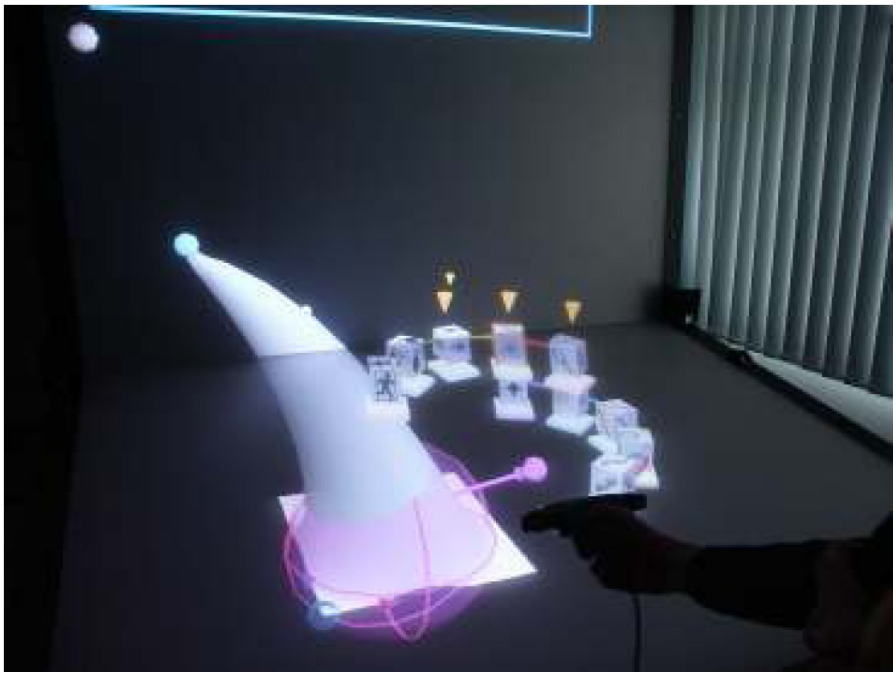


Figure 5.20: A screenshot showing the ring menu implementation by *Gerber et al.* [GB04]

**Ring** The ring menu was created by *Liang et al.* as an iterative design, improving on an earlier sphere-shaped design [LG94]. On the sphere-shaped design menu items were placed on a sphere which could be rotated around three axes to place an item in a selection box at the front where it could then be selected using a button. This design came with several disadvantages, such as issues perceiving the depth of items

on the sphere and users having trouble rotating the sphere in three dimensions. To alleviate these issues, the authors devised the ring menu, in which a ring of menu items is placed in between two semi-transparent rings, which make it easier to perceive the depth of items on the ring. An opening in the rings which always faces the user acts as a selection basket into which the items can be placed by rotating the ring around the x-axis while rotational input around other axes merely tilts the entire ring in the specified direction. The design of the ring menu was further improved by *Gerber et al.*. Their design limits the radius of the ring to a fixed size while allotting four inches of the perimeter to each menu item, thus adaptively changing the space used along the rings perimeter to match the number of menu items it contains (up to a maximum of 160 degrees of the perimeter to avoid occlusion). *Gerber et al.*'s research suggests that fixing the ring's size and then allotting a relative amount of space to each item would disturb user interaction. The currently selected item is highlighted using a colored outline (see figure 5.20) and rotation of the wrist rotates the ring in the desired direction of the rotation highlighting the current item's neighbor in the direction of rotation. *Gerber et al.* note that the main advantages of a ring menu lie in the fact that it represents a one-dimensional list, making it easy to remember the position of items, the fact that rotating the wrist is fast and requires little movement, and the fact that it is easier to maintain an orientation of the wrist than it is to maintain an unconstrained position. Its disadvantages lie in the need for a hierarchy to display any more than a small number of items.

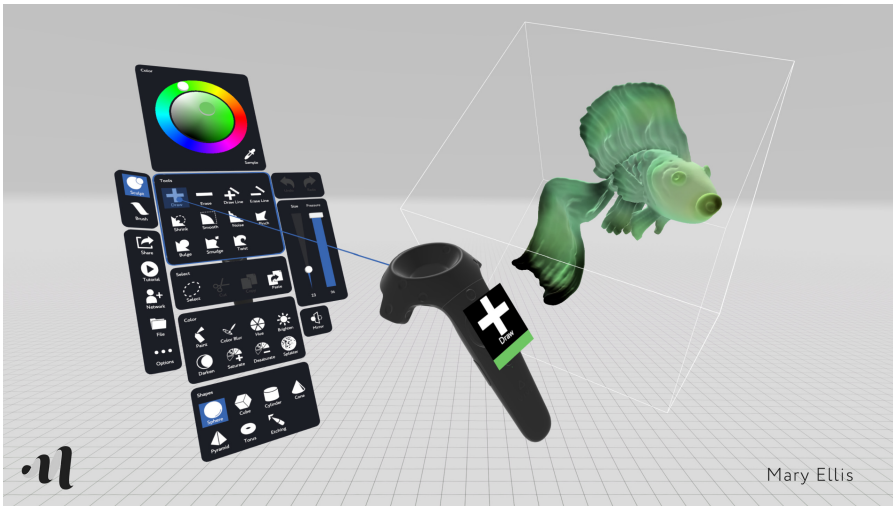



Figure 5.21: A screenshot showing a 3D palette in the Masterpiece VR 3D content creation software [Mas17]

**Matrix** While *Serra et al.* mention the use of a palette of colors for a color widget in 1995 [Ser+95] the 3D palette technique was named and developed by *Billinghurst et al.* in 1997 for rapid content creation inside a virtual environment [Bil+97]. The user is given a pen and a tablet which are mapped to a virtual pen and tablet in the simulation. On the tablet there are a number of icons which trigger different functions when tapped with the stylus such as creating a color widget or displaying a selection of different objects on the tablet which create a copy of the object when tapped. Objects can be created by drawing object profiles on the tablet which are then turned into an object either through rotation into a volume of revolution or by extruding them from the tablet surface. Created objects stay attached to the tablet until they are placed in the world. Additionally, the interface contains an option to sketch textures using a palette of colors and a canvas, both displayed on the virtual tablet. Palettes are also used in the Toolspace technique by *Pierce et al.* in which toolspaces, openings into a virtual space distinct from the scene which move with the user, act as containers for virtual objects, tools and widgets [PSP99]. Any tool or object can be dragged into the opening of a toolspace and is then stored inside. Widgets inside the toolspace can be interacted without removing them from the space. One example of such a widget is a palette of objects which can be copied and placed into the scene. The authors also suggest a color palette, from which a user could choose a color to paint the scene with. In general, 3D palettes provide the advantage of combining all choices for tools or options for the use of tools (such as colors for a brush) into a single menu solution that is usually interacted with in 2D, making the interface efficient and intuitive. If too many choices exist, however, the interface may become cluttered and usability may suffer.

**Geometric Structure** The stack menu designed by *Kim et al.* is designed to allow the user to easily remember how they arrived at their current selection by always displaying the selection path in addition to the options selectable at the current level [Kim+00]. If the selection path is displayed at the top of the menu it is called a fixed stack menu while a basket stack menu displays the selection path at any point on the screen, usually at a corner. These menus share the same general advantages and disadvantages of list-structured single menus with the added advantage of the overview the user gains from the selection path display. Another possible implementation of single menus shaped like a geometric structure are virtual shelves such as the ones used by the Meta 2 Augmented Reality headset [Met17b]. The headset is designed to expand a desktop workspace by adding additional Augmented Reality screens and applications. A number of applications are placed on the boards of a virtual shelf from which they may be picked [VR 18]. Dragging an application out of the shelf invokes it into the workspace. The main advantage of this technique is the immediate familiarity of the

Figure 5.22: A screenshot showing the virtual shelf of the Meta 2 Augmented Reality headset being used [VR 18]

design: Real shelves are designed for the storage of objects which may be picked up and the concept is transferred directly into Augmented Reality. The main disadvantages are the shape of the shelf limiting the amount and type of menu items which can be placed into it as well as the extensive 3D interaction needed to select an item from the shelf using grasping selection which reduces efficiency.

**Free Layout** Menu books, as used in the tile interface designed by *Poupyrev et al.* [Pou+02] are a good example of a single menu with a free structure. The tile interface consists of tiles, small magnetic cardboard cards with an AR marker, a whiteboard, and a book. The tiles serve as icons, virtual objects and tools to interact with either, such as, for example, a help tile being placed next to another tile invoking an explanation of that tile's purpose. Serving as both a catalog and a menu, each page of the book contains a different tile representing a tool. To select a tool, the user places an empty data tile next to the desired tile in the book. The tool from the book is copied onto the data tile which can now be placed onto the whiteboard. The greatest advantage of the menu book is the fact that it is a real, tangible book which can be "invoked" simply by picking it up, dismissed by putting it away and flipped through to find the desired tool. There is very little learning, and no experience in virtual environments needed to choose and pick a tool. Its disadvantages lie in the fact that adding tools to the book increases its physical size, limiting the number of selectable items lest the book becomes unwieldy and the fact that to use the menu the user needs to use both

hands, one holding the book, the other a tile, which makes adding new tools to the whiteboard a time consuming task. Another example of a free layout being used for a single menu is the start palette used in the Task Gallery, a 3D window manager developed by *Robertson et al.* [Rob+00]. In the Task Gallery, the workspace is represented by an art gallery and the current task and its corresponding windows are placed on a stage at the end of the gallery. Other tasks are placed on the walls, ceiling, and floor of the gallery and can be moved to the stage by clicking on them. The start palette is stored in a toolspace (previously discussed in the section on matrix-structured single menus) and shaped like a traditional painters palette. It contains icons representing web pages, applications and favorite documents, which, when clicked, open up a new application window into the current task. The author's research suggests that searching for an icon is easier and faster when using a start palette compared to the traditional Windows start menu.

#### Menu Systems

Menu systems are single menus that may contain one submenu for each selectable item, making them menu hierarchies with a maximum depth of two.

**List** While the spin menu designed by *Gerber et al.* [GB05] (previously discussed in the section on ring-structured temporary option menus) is mainly used as a temporary option menu, its ability to be extended through the addition of hierarchies makes it possible to use them as menu systems. Specifically, the crossed-layout, in which the next layer is displayed and manipulated orthogonally to the first one, displaying the selection path as a list in the middle, is classified as a list-structured single menu by *Dachselt et al.* [DH07]. It combines the advantages of a ring-shaped menu, such as quick and efficient operation, with the additional number of selectable items of a menu system. It is however still limited to a maximum number of 9-11 selectable items per layer and the change of manipulation and display direction may initially confuse users.

**Ring** A common implementation of ring-structured menu systems is the model of a revolving stage as described by *Dachselt et al.* [Dac99]. The implementation, designed to showcase the product catalog of an implant dentistry company, consists of two circular virtual revolving stages, facing each other. The stage in the back, consisting of columns showing all products of a particular category, can be rotated to select a category. If a category is selected, the list displayed on the column can be browsed like a traditional list-structured menu by scrolling and selecting products from it. If selected, a product is rendered in 3D in front of the column and can be rotated and zoomed. The stage in the front can be switched between this product presentation mode and a virtual surgery

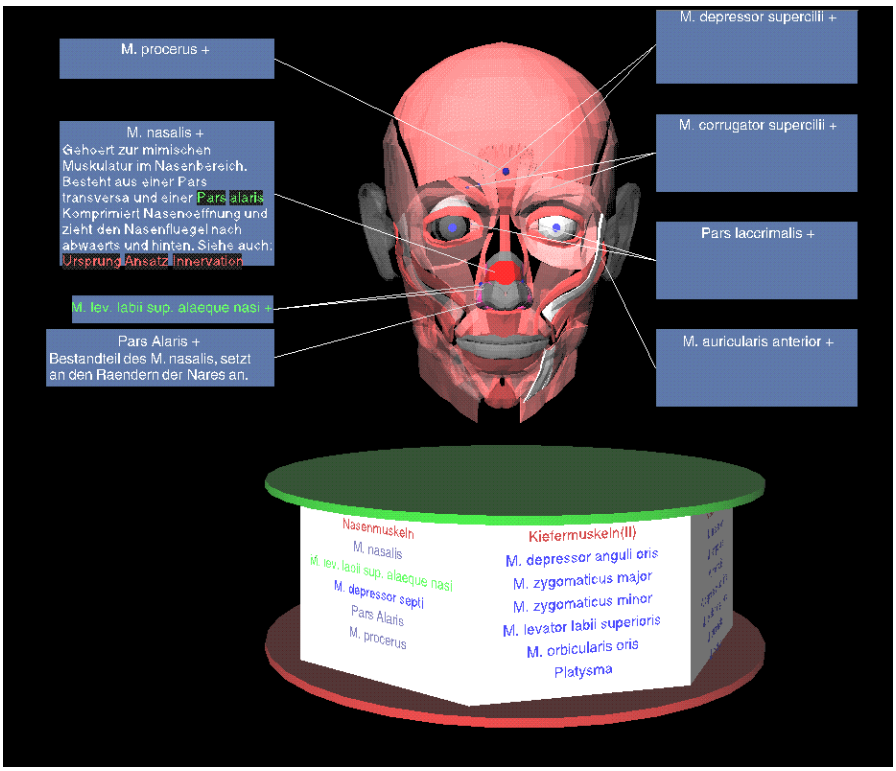



Figure 5.23: A screenshot showing the rondell being used to display and hide information nodes on a model [PRS97]

mode by rotating. In the virtual surgery mode, usage of the products can be simulated, videos demonstrating procedures are shown and a surgery table can be configured. The revolving stage model has the advantage of being able to use the ring structure for the part of the hierarchy with fewer options (in this case product categories) where ring menus are most efficient, while using a list structure when there are more options (in this case products in a category) and a ring menu would be inefficient compared to a list. If there are are a large number of options at a depth of one, this type of menu would be unsuitable. The same is true for the front stage if selecting a menu option does result in a reaction that requires a second large interaction space in front of the menu. A very similar implementation is the rondell developed by *Preim et al.* [PRS97]. The rondell is of a circular shape with flattened sides and a disc on the top and bottom (see figure 5.23). Menu options are grouped into categories with each side of the rondell displaying a list of items of one category which can be selected by clicking. The current side can be chosen by rotating the rondell which is achieved by clicking on either the

upper or lower disk which rotate the rondell left or right, respectively. While the rondell has the advantage of being able to easily display options for a number of categories the authors note that there is little familiarity to it and it will have to be learned to be used.

### Menu Hierarchies

Menu hierarchies, often called cascading or tree-structured menus, are menus which allow any number of items to be arranged in any number of submenus (with at least one submenu hierarchy reaching a depth of three).

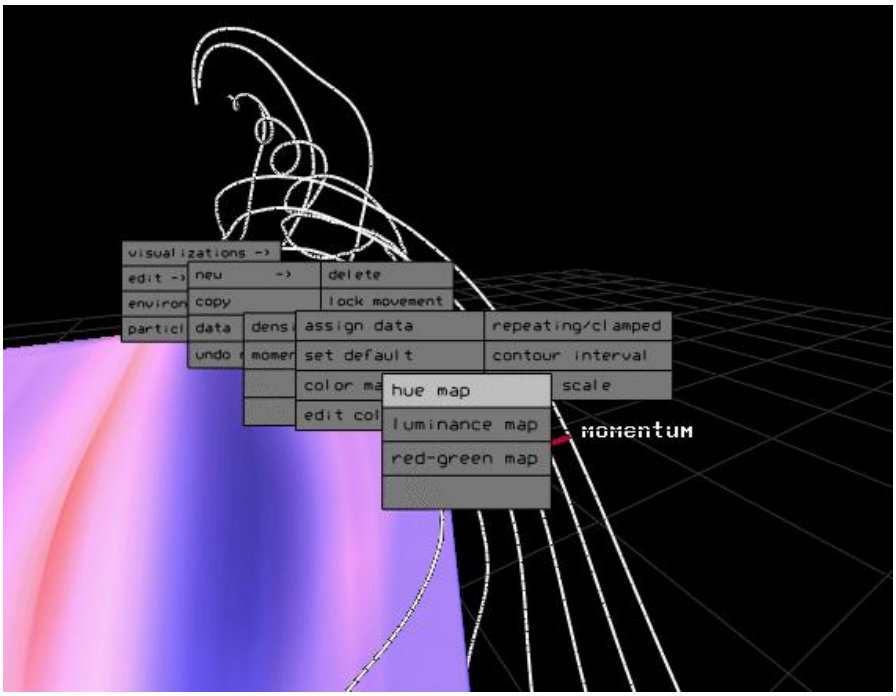



Figure 5.24: A screenshot showing the hierarchical pop-up menu used in the virtual windtunnel application [NAS95]

**List** Possibly the simplest menu hierarchy is the hands-off interaction menu designed by *R. Darken* [Dar94]. The menu consists of a simple two-dimensional overlay over the viewplane on which a textual menu is displayed. The menu consists of one or two lines of textual menu items which may either be commands or submenus and are selected through speech recognition. This type of menu has the advantage of being extremely simple to understand and use and not requiring the use of the user's hands keeping them free. However, the speech recognition-based input limits the menu to either purely

textual options or options containing text. A similar menu implementation results from adding submenus as menu options to the Tinmith system (previously discussed in the section on list-structured single menus). This has the advantage of being able to display any type of data as the Tinmith system uses a pinch gesture to select menu items. The virtual windtunnel application designed by *S. Bryson* uses hierarchical pop-up menus for system control [Bry93]. If one uses a dataglove to make a pointing gesture in empty space, a pop-up menu is displayed at that point in space for as long as the gesture is held. If the gesture is released while pointing at an option the menu is closed and that option is executed. Unlike the pop-up menus discussed in the section on temporary option menus, these menus support submenus of arbitrary depth giving them the advantage of being able to contain a large number of items. Additionally these menus are familiar to almost any user as their design strongly resembles those of pop-up menus used in many two-dimensional applications.

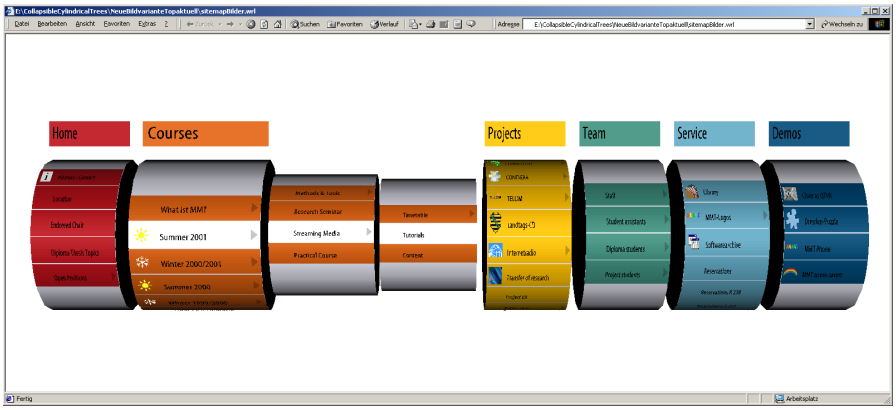



Figure 5.25: A screenshot showing a sitemap of a website being displayed as collapsible cylindrical trees, demonstrating their layout [DE01]

**Ring** A simple ring-structured menu hierarchy called the fade-up menu was developed for the Holosketch application by *M. Deering* [Dee95]. The fade-up menu is a large pie-shaped pop-up menu that is invoked by holding down a button on the tracked wand used as an interaction tool and vanishes when the button is no longer held. Upon pressing the button, the current workspace fades into the background and the menu fades in. The menu appears at the position of and is centered around the tip of the wand. Menu buttons may either change current modes, cause an immediate action (these buttons light up when approached) or open submenus and are selected by touching them with the tip of the wand. If a button invoking a submenu is pressed, the fade-up menu moves away from the user and the (pie-shaped) submenu appears

at the current position of and cantered around the tip of the wand. If the wand is moved too far away from the submenu, the submenu fades out and the main menu moves back forward. While this shape of menu requires the user to move the hand holding the wand throughout a large area to select menu items, reducing efficiency, it has the advantages of being easy to understand, being able to display a large number of items at once and making it easy for the user to view all selectable items at once. Another simple menu hierarchy is the spin menu (previously discussed in the section on list-structured temporary option menus) devised by *Gerber et al.* if it is used with submenus in a concentric layout [GB05]. In this layout, if a submenu is invoked from a button on a layer, the submenu is arranged around the current layer forming concentric circles. This retains the benefits of the spin menu while mitigating the disadvantage of being able to display only a small number of items per layer by allowing for a hierarchy with a large depth. A more complex solution, which combines the hierarchical structure of two-dimensional menus with a 3D menu, are the collapsible cylindrical trees which were designed by *Dachselt et al.* [DE01]. This design arranges menu items as a list around a cylinder, displaying a row of cylinders each representing a category of options. The number of items displayed on these cylinders is not limited by their radius, as the items currently not facing the user may be exchanged dynamically allowing the list to contain an arbitrary number of items. If an option on a cylinder that invokes a submenu is selected, the cylinders which are not on the current selection path are scaled down horizontally while a smaller cylinder appears to the right of the currently selected one representing the submenu. As the authors designed this system to accommodate hierarchies of large depths, if a certain minimum of size is reached for the cylinders not on the path, the cylinders on the path may also be scaled down starting with the leftmost cylinder on the path (the original parent). This layout has the advantage of allowing for an arbitrary number of menu items in each category and submenu though it is limited to a small number of categories due to the cylinders of all categories always being visible.

**Geometric Structure** Cone trees, as described by *Robertson et al.* are a data structure designed to visualize and browse large amounts of selectable hierarchical data, which makes them easily adapted to serve as a menu [RMC91]. A cone tree is a tree with layers of equal height, the height of the room divided by the depth of the tree, with the root placed at the ceiling and the lowermost leaves on the floor of a virtual room. A transparent cone spreads from the root and the root's child nodes are placed evenly spaced around its base, each child node being the root of its own cone of children and so on. The diameter of the child cones is progressively reduced with increasing depth to ensure that the entire tree's base fits into the virtual environment. The selection of a

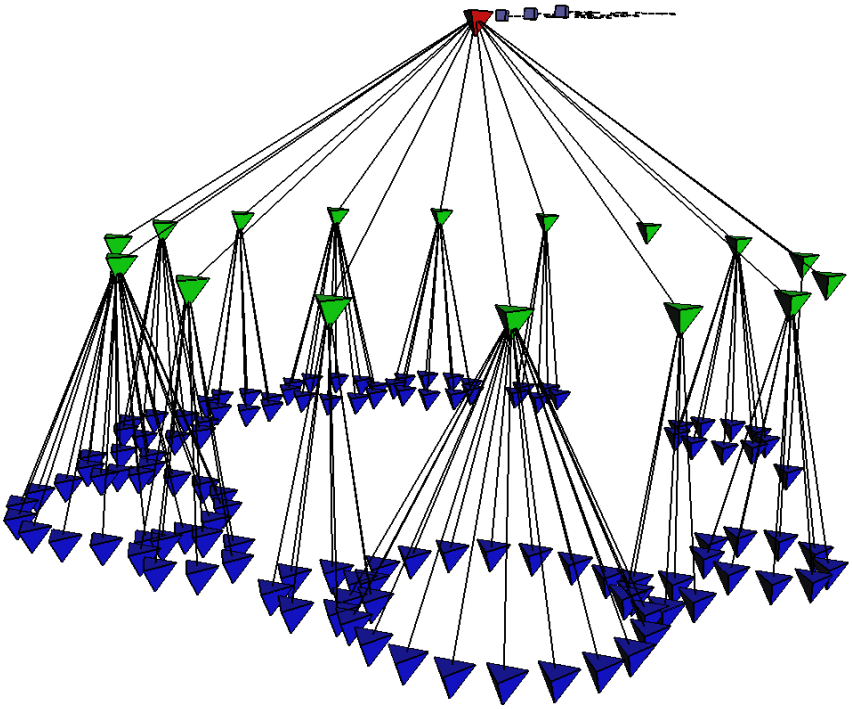


Figure 5.26: A screenshot showing the layout of a cone tree with two layers of nodes [MB95]

node using the mouse triggers both a highlight of the node and the current selection path and a rotation of the tree to bring both to the center of the user's field of view. The authors recommend animating the rotation instead of performing it instantly as the user might otherwise lose track of substructure relationships. The greatest advantage of this structure is the ability to display, and thus give the user an overview of, up to a 1000 nodes without cluttering as long as the depth is no more than 10 and the maximum branching factor is no more than 30. This allows the user to select the correct node from a large amount both more quickly and more efficiently. The authors do note however, that the tree is much more usable if the hierarchy is unbalanced as balanced trees make it harder for users to keep track of substructure relationships due too their uniformity. *Butz et al.* have designed a tangible user interface device called

Tuister to browse hierarchical menus which uses cone trees as its metaphor for menu interaction [BGK04]. The Tuister is a cylindrical device consisting of a handle and a cylindrical (possibly segmented) display which can be moved independently of the handle. The primary part of the display (or primary display) at first displays the first layer's first node, until the display is turned against the handle with the handle staying fixed, which scrolls through the first layer. To select an entry the display is held fixed and the handle is turned clockwise, which selects the node and, if it invokes a submenu, moves to the submenu. Conversely, moving the handle counterclockwise will move to the layer above the current one. The authors note that they believe this design will enable users to interact with hierarchical menus even more efficiently but stress the importance of both user studies to verify their theories and further refinement of the design.

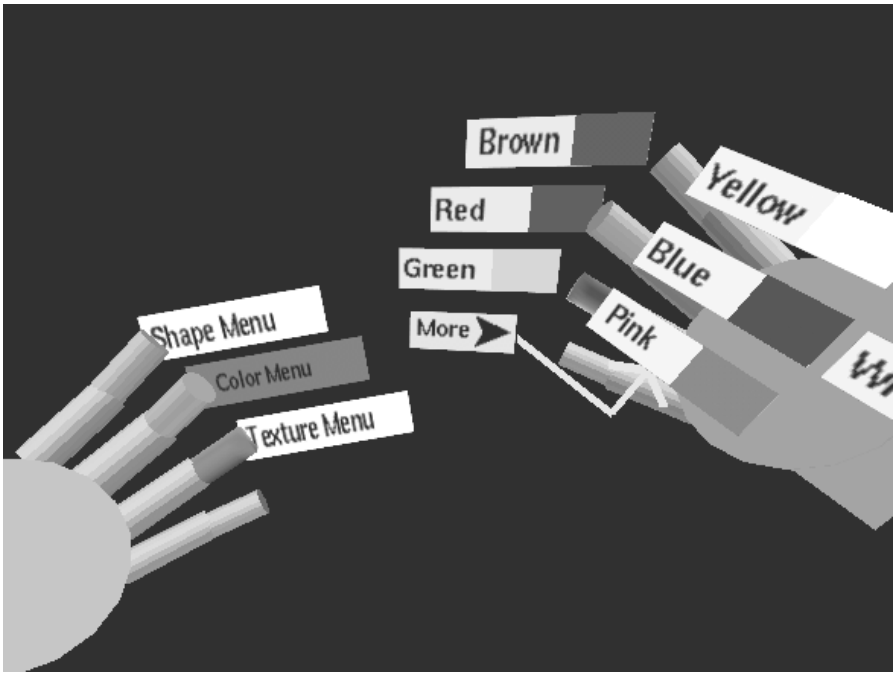



Figure 5.27: A screenshot showing the layout of the Tulip menu [BW01]

**Free Layout** The Tulip system, designed by *Bowman et al.*, combines pinch-glove menus (such as the Tinmith system discussed in the section on list-structured single menus) which have each finger represent a menu item with a hierarchical structure to be able to display more data [BW01]. It was formed from two prototypes through formative evaluation, those prototypes being the scrolling menu and the three-up menu.

Both prototypes use pinch gestures to select top-level menus placed on the fingers of the non-dominant hand but the scrolling menu prototype places a scrollable list of items in the dominant hand with its fingers being used to scroll through and select items. The three-up menu places a menu item on each of the first three fingers of the dominant hand with the pinky finger representing a "more options" option which can be selected to replace the current three items with the next ones. While the scrolling menu was able to show more items at a time and was preferred by users, the three-up design required fewer movements and time to select the desired option. As both options did not satisfy the requirements the authors combined them into the three-up, labels-in-oalm (TULIP) system. The Tulip system uses the three-up system of selecting menu options placed on fingers of the dominant hand but the options which will be scrolled to by selecting "more options" on the pinky finger are displayed on the palm in groups of three and connected to the pinky with an arrow. If "more options" is selected the current options are replaced with the first three options displayed on the palm and all options appear to move towards the fingers. While the Tulip menu is not suited to large and complex hierarchies it is a fast and efficient system which may even be used "eyes-off" by experienced users according to the authors. Its biggest drawbacks are its initial adjustment period as the authors' study has shown that it was noticeably more difficult for novice users than other systems and the inability to correct errors made in the selection process as a pinch cannot be taken back in the same way as a stylus can be moved off an erroneous selection.

### 5.3.2 Tools

A virtual tool is either a purely virtual representation of a system control technique as an object or a virtual representation of a real world device or object performing a system control function. The prevalence of tools and menus that were created just to switch between different tools in surveyed applications (see section 5.3.1) shows that they are one of the most common and important system control and interaction methods. *LaViola et al.* [LaV+17] identified three different forms of tools, purely virtual tools which have no physical representation, tangibles which are virtual tools represented by abstract shapes in the real world without any connection between their form and function, and physical tools which are mapped one-to-one to a virtual tool of the same shape in the virtual environment.

An example of a physical tool called the virtual tricorder is found in *Wloka et al.*'s work [WG95] (previously discussed in section 5.3.1). The physical tricorder is held in the user's hand and its position and orientation is mapped to its virtual representation. The tool can be used to accomplish different system control tasks in the virtual environment, such as selection through a cone-shaped selection volume originating at the tricorder,

manipulation of the selected object by manipulating the tricorder, or magnification of objects by looking through a magnification lens on top of the tricorder. The tricorder may also be used as a tool for traveling, as it can simulate both a virtual grappling iron which can be thrown and pulled to move the user, or virtual jet thrusters which move the user who steers using the tricorder. The user can switch between these functions using a pop-up menu which is displayed on top of the virtual tricorder.

An example of a tangible user interface are the menu tiles developed by *Poupyrev et al.* [Pou+02] (previously discussed in section 5.3.1). The menu tiles are magnetic cardboard tiles which have Augmented Reality markers printed on them and are designed so they can be placed on a whiteboard serving as a workspace. The tiles can represent both tools and the objects they interact with, their current function being displayed on top of them using the markers. The functions the tiles representing tools can fulfill include deletion of an object, copying an object or invoking a help function displaying further information. The function of these tiles can be chosen from a book of tiles by simply placing a tile not currently representing a function or data next to the page of the book representing that function, at which point the function is copied onto the empty tile.

Finally, an example of a purely virtual tool are the tools inside the toolspaces developed by *Pierce et al.* [Pie+99] (previously discussed in section 5.3.1). Toolspaces are an easily accessible virtual storage system into which the user can place objects and tools at any time. The tools stored in the toolspace, such as object and color palettes or portable widgets, can be removed from the toolspace at any time or location to perform the task they were designed for.

Each approach comes with its own unique set of advantages and disadvantages. Physical tools, and to an extent tangibles as well, allow eyes-off interaction and a greater sense of immersion due to the user being able to touch the tool in the real world and receiving appropriate haptic feedback. On the other hand, a physical tool that becomes too generalized to accommodate a larger set of tasks, may end up being less suited to each individual task or as abstract as tangible, which may reduce immersion. Virtual tool techniques on the other hand have the advantage of being able to invoke a tool of any shape, form and function at any time and location without having to find or carry, or without the developer having to manufacture, a physical tool. The opposite is also true, as both tangibles and physical tools have to be put away or carried while they are not in use, as opposed to virtual tools which can simply vanish once the user no longer needs them.

## 5.4 Communication And Collaboration

As both collaborative virtual environments and multiplayer games become more and more common in virtual and Augmented Reality, it thus becomes more and more important to properly handle both how multiple users are displayed by the application and how they may communicate with one another.

### 5.4.1 Shared Environments

There are numerous challenges one encounters when designing for a shared environment, such as how to display other immersed users in the virtual environment or how user interfaces may facilitate interaction between users. As these aspects strongly depend on the nature of the shared experience, this section is grouped based on the classes of shared environments defined by *Sherman et al.* [SC03].

Figure 5.28: A picture showing several non-immersed players discussing instructions for the immersed player of the game Keep Talking and Nobody Explodes [Ste18]

#### One Immersed Person With One Or Several Onlookers

As this style of shared environment is used mostly for demonstrations of virtual environments, with one user wearing a headset and others seeing a display of their view on a screen, or for entertainment purposes, such as *Jaron Lanier*'s concerts in which he played instruments in virtual reality while the audience saw the environment

displayed on a screen and heard his virtual instrument [Swe94]. As the audience has no way of interacting with the environment except through the immersed user, there are very few if any differences to user interfaces in non-shared environments. One exception is found in the game *Keep Talking And Nobody Explodes* [Ste18] in which one player wearing a virtual reality headset is trying to defuse a bomb with no instructions being displayed to them. The other players are handed a printed set of instructions on how to defuse each component of the bomb, working together with the immersed player by using the immersed player's description of the components to derive the correct instructions for the immersed player to carry out. Here the printed pages act as a display of information and the immersed player acts as an extension of the user interface for the non-immersed players.


Figure 5.29: A screenshot from the game *Catan VR* showing the avatars of three other users [4PL18]

### Two Or More Immersed Persons

While *Sherman et al.* note that immersing two ore more persons in the same environment may be achieved using different methods for each participant, such as a headset user sharing an environment with the user of a CAVE system [SC03], two or more persons being immersed using headsets is perhaps the most interesting case as it covers most of the shared environments being developed for end-users today.

Perhaps the most important issue when developing this kind of application is how other users are displayed. Usually every user is given an avatar to represent themselves in the environment, the appearance of which may vary based on whether all users are

performing the same tasks or fulfilling similar roles, and what those tasks and roles entail. A simple example is the virtual reality application *Catan VR* [Asm18] which allows users to play the popular board game Catan in virtual reality. Its multiplayer mode allows up to four players to play the game against each other, representing each user as a floating mask, the style of which can be selected, and a pair of floating hands in the same style as the mask. This style of avatar has the advantage of being just humanoid enough for players to associate them with another player without needing additional details such as facial animation. Additionally, the floating hands of the avatar are sufficient to represent all movements needed to play the game, as no interaction other than selection and manipulation of menu items and game pieces is needed, while also adding a layer of communication by accurately representing gestures of users. Other applications, such as the virtual reality social platform *AltspaceVR* [Mic18] opt for entirely humanoid, albeit stylized, avatars. These avatars have the advantage of entirely mimicking an actual human, which is useful to incentivize communication and interaction in a social application. It does this without falling into the "uncanny valley", which makes users uncomfortable if an obviously virtual avatar resembles a human too much [MMK12], due to its avatars being designed to look like cartoon characters. Another popular approach to avatars is simply using vehicles to represent the players found in, for example, the game *Eve Valkyrie* [CCP18] in which the player pilots a small spaceship in dogfights against other players. A similar albeit slightly different implementation is found in *Ubisoft*'s game *Eagle Flight* [Ubi18] in which the player assumes the role of an eagle, which is then steered like a vehicle around a virtual environment.

#### Open Display

According to *Sherman et al.* open display environments consist of large displays which are seen by several users at once with one or several of them, but not all of them, being able to interact with the environment through tracked or tracking devices which can be shared with other users. The user interfaces, however, are still usually designed to be interacted with by one person at a time and the ability to simply see other users in the real world instead of needing avatars makes it unnecessary to create intricate inter-user interaction techniques, thus making open displays uninteresting from the perspective of visible user interface research.

#### Multi-Person Cockpit

Multi-person cockpit environments, defined by *Sherman et al.* [SC03] as consisting of one large screen through which an outside virtual environment is seen are almost always

interacted with through two-dimensional user interfaces on the screen or tangible cockpit instruments serving as input and sometimes output devices, thus making this environment uninteresting for the design of 3D user interfaces.

### 5.4.2 Communication

If there is to be collaboration in a shared environment, communication of some form is a necessity. In real life communication is usually done through speech, either in face-to-face conversations, over the phone, through voice or video chat, or through voice messages. Face-to-face and video conversations additionally add a layer of body language and gestures to the conversation, which may aid in conveying both emotions, through both, and spatial information, which is conveyed through gestures such as pointing or indicating shapes and sizes. Another common method of communication is through text, either in real-time chat systems or delayed messaging systems. As all of these communication methods may be adapted to VR in some way, *Sherman et al.* differentiate between synchronous communication, in which all users communicate at the same time, and asynchronous communication, in which users may initiate communication at any time but the recipient may access the message at a later time.

#### Synchronous Communication

Synchronous communication may take very different forms depending on the type of application, type of environment and where the application falls on the spectrum between augmented and virtual reality. The simplest form of synchronous communication is a voice chat system, either between a group of users or all users present in the environment. An example of a system that connects all users is the voice chat system used in the *VRChat* application [VRC18] in which all users in the environment can be heard by all users. To avoid the issues caused by too many people talking at once in different conversations, *VRChat* uses spatialized audio which lowers the volume of other users voice in proportion to the distance of their avatar from the user's avatar, and adds directional audio indicating their direction from the user, thus being able to mimic real world gatherings and their multitude of conversations. Additionally, the user's avatar's mouth moves and a loudspeaker icon is displayed above their head if they are speaking, making it easier to keep track of who is currently speaking in a conversation. If a private conversation is desired, solutions like the Oculus Rift's Parties system [Ocu18] are more suitable. Using the Parties system, users can invite up to four other users from their friends list into a party, who are then able to join a virtual room together, launch an application together, but most importantly, are placed into a private Voice-over-IP (VoIP) conversation which persists even when launching

other applications. While inside a virtual room in the Oculus Home application, the user's avatar moves its lips when it is talking, however there is no other indicator to help differentiate who is currently talking. As gestures make it easier to convey information and emotions, many social applications using avatars such as, for example, *VRChat* [VRC18], track the position of the user's hands, usually through controllers held in both hands, and use this information to move the arms and hands of the user's avatars in VR to perform the same gestures. While this is, of course, beneficial to communication it also increases the users' immersion as it makes the avatars movements and behavior during conversations seem more natural.

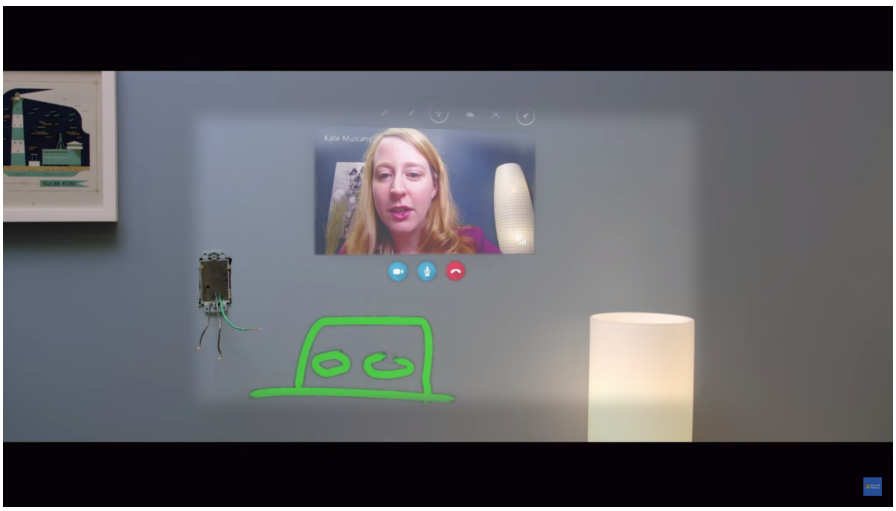


Figure 5.30: A screenshot showing a desktop user drawing on the view of a Hololens user in a video call using the *Skype* application [Mic16]

The simplest way to transmit both gestures and voice, however, is a video call. While VR headsets are unsuitable for video calls due to obscuring large parts of the user's face, both see-through Augmented Reality headsets and webcam-based Augmented Reality can make use of video calls. The ZugSTAR system [Zug18], a video conference application using the user's webcam, allows users to use Augmented Reality functions in their video call, displaying both their own and the other user's augmented video stream. The Meta 2 Augmented Reality headset [Met17b], due to its suggested use case of being tethered to a desktop system and its ability to render a virtual window displaying a video stream in the user's viw [Met17a], is extremely suitable for video calls using an external webcam, and such calls are one of the features suggested for development by its developers. Another possible application is the ability to stream the current view of the headset's user in a video call and letting the other party augment

that view, a function which is present in the *Skype* application on the Microsoft Hololens headset [Mic16]. In video calls between a Hololens user and a desktop user, the desktop user is able to draw shapes which are then displayed in the view of the Hololens user in addition to the video stream of the call, which can be used to add additional information into a conversation, such as, for example, detailed instructions on using or repairing devices with which the desktop user is more familiar than the Hololens user.

**Asynchronous Communication**

Asynchronous communication usually takes place in one of two forms, changes made to the virtual environment or a dedicated asynchronous messaging system. Another, more rarely used technique is the playback of virtual experiences. Changes made to the virtual environment for the purpose of communication are usually done through some form of annotation. *Sherman et al.* [SC03] define four different forms of content an annotation can contain: voice, text, gestural and pictorial. Voice annotations are easily recorded using a microphone which is present in most systems and require little effort from the user. Text annotations are easy to read even outside a virtual environment and are searchable, but may be cumbersome to write depending on the application's symbolic input method. Gestural annotations are easy to enter for an immersed user if a method to track gestures is present in the setup but require treatment when they are replayed such as either a copy of the user's avatar or a generic avatar performing the gestures. Finally, pictorial annotations may take the form of screenshots of the user's current view or of an external perspective and are easily displayed both inside and outside the environment. These annotations may then be attached to locations or objects, at a particular point of view, a specific time or a combination of any of these possibilities such as, for example an annotation at a location. One of the simplest form of annotations is found in the VR and AR-capable model viewer of the online 3D model directory *Sketchfab* [Jam18] which supports text and pictorial annotations attached anywhere on a model, represented by a small circular icon displaying the index of the annotation. As these models may also be used to represent a larger virtual environment and the viewing application supports jumping from the location of one annotation to the location of the next one, these annotations can be used like a guided tour of a virtual environment. An example of this are the wreck tours created by the *Maritime Archeology Trust* [Mar18] which are models of a ship wreck at the bottom of the sea designed to be explored in virtual reality, with a set of ordered annotations containing information about the wrecks history, discovery, and interesting objects and locations around it serving as a virtual guide around the environment. Common asynchronous communication methods include text-based communication techniques such as messenger applications, e-mail or chat rooms. While they are rarely used in

Virtual Reality due to the clunkyness of current symbolic input methods, Augmented Reality headsets which allow the user to see, and type on an actual keyboard, are very well suited to this kind of asynchronous communication. Headsets like the Meta 2 device [Met17b] which allow users to open web browser windows in Augmented Reality already support this functionality simply by browsing to a website or web interface containing such a communication service.

# 6 Future Work

The research discussed in this thesis has shown that while many of the core concepts of three-dimensional user interfaces have been developed 20 or more years ago, and many of the core design guidelines can be traced back even further to the groundwork laid by Human-computer-Interaction (HCI) researchers working on two-dimensional interfaces, the problems posed in the field of three-dimensional HCI are still far from being completely solved. The advent of consumer hardware has accelerated development with many researchers, students, game developers and firms worldwide working on new techniques and interfaces. In turn, this necessitates constant work keeping up on newest developments and regular updates to surveys simply to be able to gain an overview of the state-of-the-art of the field. Thus, the best solution for researchers and developers wishing to gain an overview would have to be dynamic enough to accommodate regular updates, structured simply enough to be easily comprehended while also being able to provide detailed information on each task. A searchable database of user interface techniques would likely be one of the best ways to accomplish this task. Using the classification system developed in this work to enforce an order over all entries in the database would make it easier to both gain an overview of all methods existing to solve a task, enabling developers to perform a top-down search of available techniques,

Such a database could then serve as a base for, and could be expanded by, further 3D HCI research. A likely first step would be creating a virtual environment in which several representative tasks are to be solved and a suite of tools measuring a user's performance in these tasks analyzing different aspects, such as errors made, accuracy, and the time needed to complete the task. The user interfaces cataloged in the database would then be implemented in this environment which would enable researchers to not only perform a quantitative analysis of all techniques but also make the results of that analysis comparable to each other, which was previously impossible due to task setup differing from study to study. The environment could also be used to study the effectiveness of modifications and improvements to current techniques and serve as a convenient setup for comparative qualitative analysis, studying, for example, the difference in the level of user fatigue or comfort between different techniques.

# 7 Conclusions

Virtual and Augmented Reality will become an even larger part of industrial, medical, educational, and consumer entertainment applications. As their importance grows, so does the importance of 3D Human-computer-Interaction (HCI) research, as the quality and efficiency of user interfaces are almost entirely responsible for the quality of the user experience and efficiency of the usage of an application. While games and entertainment are predicted by many to be the "killer app" which will lead to widespread adoption of 3D user interfaces [LaV+17] and a surge in the development of new techniques and technologies, efficiency is perhaps the most important property of 3D user interfaces for industrial or medical purposes. This makes it likely that the development of applications for these fields may lead to new discoveries regarding the design of 3D user interfaces for efficiency.

This thesis aimed to explore the current state-of-the-art, survey and evaluate past and present developments and classify them. To this end evaluation methods and criteria were discussed and an informal evaluation method was decided on, which consisted of comparing each technique discussed to those trying to fulfill the same task and discussing their advantages and disadvantages, giving special attention to the evaluation criteria found in research on user interface evaluation. Using classification methods and taxonomies developed in several related surveys, a new classification system was created by merging several task-based and metaphor-based classification systems. This new classification system was used as a basic hierarchy of interaction tasks and possible solutions which was then expanded with examples, discussions and evaluations of these solutions.

While we believe our survey and discussion to be as comprehensive as the scope of this thesis allowed, each implementation of a solution may contain details or small changes unique to its specific application and each task and class of solutions discussed warrants further exploration, especially in light of ongoing development. Additionally, we believe the classification system presented in this work to be suitable as a base for a catalog or database of possible UI solutions, which could then serve as a base for further research such as comparative qualitative analysis of the efficiency of different solutions.

# List of Figures

# Bibliography


[4PL18] 4PLAYERS.DE. *Catan VR Bilder*. http://www.4players.de/4players.php/screenshot_list/Allgemein/39086/Screenshots/83487/0/Catan_VR.html. [Accessed: 3-June-2018]. 2018.

[agu16] agumonkk. *Chronos Walkthrough Part 2: The Pan*. https://www.youtube.com/watch?v=h3gv97qtDAY. [Accessed: 17-April-2018]. 2016.

[Asm18] Asmodee Digital, SAS. *Catan VR*. http://www.asmodee-digital.com/en/catan-vr/. [Accessed: 30-May-2018]. 2018.

[Azu97] R. T. Azuma. "A Survey of Augmented Reality." In: *Presence: Teleoper. Virtual Environ.* 6.4 (Aug. 1997), pp. 355–385. ISSN: 1054-7460. DOI: 10.1162/pres.1997.6.4.355.

[BCL15] M. Billinghurst, A. Clark, and G. Lee. *A Survey of Augmented Reality*. Now Foundations and Trends, 2015, pp. 218–. ISBN: 9781601989208. DOI: 10.1561/1100000049.

[BF07] H. Benko and S. Feiner. "Balloon Selection: A Multi-Finger Technique for Accurate Low-Fatigue 3D Selection." In: *2007 IEEE Symposium on 3D User Interfaces*. Washington, DC, USA: IEEE Computer Society, Mar. 2007. DOI: 10.1109/3DUI.2007.340778.

[BGK04] A. Butz, M. Groß, and A. Krüger. "TUISTER: A Tangible UI for Hierarchical Structures." In: *Proceedings of the 9th International Conference on Intelligent User Interfaces*. IUI '04. Funchal, Madeira, Portugal: ACM, 2004, pp. 223–225. ISBN: 1-58113-815-6. DOI: 10.1145/964442.964486.

[BH97] D. A. Bowman and L. 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*. I3D '97. Providence, Rhode Island, USA: ACM, 1997, 35–ff. ISBN: 0-89791-884-3. DOI: 10.1145/253284.253301.

[BH99] D. A. Bowman and L. F. Hodges. "Formalizing the Design, Evaluation, and Application of Interaction Techniques for Immersive Virtual Environments." In: *Journal of Visual Languages & Computing* 10.1 (1999), pp. 37–53. ISSN: 1045-926X. DOI: https://doi.org/10.1006/jvlc.1998.0111.

[BI05] C. W. Borst and A. P. Indugula. "Realistic Virtual Grasping." In: *Proceedings of the 2005 IEEE Conference 2005 on Virtual Reality*. VR '05. Washington, DC, USA: IEEE Computer Society, 2005, pp. 91–98, 320. ISBN: 0-7803-8929-8. DOI: 10.1109/VR.2005.65.

[Bil+97] M. Billinghurst, S. Baldis, L. Matheson, and M. Philips. "3D Palette: A Virtual Reality Content Creation Tool." In: *Proceedings of the ACM Symposium on Virtual Reality Software and Technology*. VRST '97. Lausanne, Switzerland: ACM, 1997, pp. 155–156. ISBN: 0-89791-953-X. DOI: 10.1145/261135.261163.

[BKH97] D. A. Bowman, D. Koller, and L. F. Hodges. "Travel in Immersive Virtual Environments: An Evaluation of Viewpoint Motion Control Techniques." In: *Proceedings of the 1997 Virtual Reality Annual International Symposium (VRAIS '97)*. VRAIS '97. Washington, DC, USA: IEEE Computer Society, 1997, pp. 45–. ISBN: 0-8186-7843-7.

[BLA16] BLACKISH. *VRTX - VR 3D Modeling - Prototype Test 2*. https://www.youtube.com/watch?v=-BHNQmSs3BA. [Accessed: 3-May-2018]. 2016.

[Ble18] Blender Foundation. *Blender*. https://www.blender.org/. [Accessed: 27-April-2018]. 2018.

[BLS15] G. Bruder, P. Lubos, and F. Steinicke. "Cognitive Resource Demands of Redirected Walking." In: *IEEE Transactions on Visualization and Computer Graphics* 21.4 (Apr. 2015), pp. 539–544. ISSN: 1077-2626. DOI: 10.1109/TVCG.2015.2391864.

[Bow+98] D. A. Bowman, J. Wineman, L. F. Hodges, and D. Allison. "Designing Animal Habitats Within an Immersive VE." In: *IEEE Comput. Graph. Appl.* 18.5 (Sept. 1998), pp. 9–13. ISSN: 0272-1716. DOI: 10.1109/38.708555.

[Bow+99] D. A. Bowman, E. T. Davis, L. F. Hodges, and A. N. Badre. "Maintaining Spatial Orientation During Travel in an Immersive Virtual Environment." In: *Presence: Teleoper. Virtual Environ.* 8.6 (Dec. 1999), pp. 618–631. ISSN: 1054-7460. DOI: 10.1162/105474699566521.

[Bro17] M. Brown. *How to customize the HTC Vive's Chaperone with SteamVR*. https://www.vrheads.com/how-customize-htc-vives-chaperone-steamvr. [Accessed: 2-February-2018]. 2017.

[Bro82] D. Broderick. *The Judas Mandala*. Pocket science fiction. Pocket Books, 1982. ISBN: 9780671450328.

[Bry93] S. Bryson. "The Virtual Windtunnel: A High-performance Virtual Reality Application." In: *Proceedings of the 1993 IEEE Virtual Reality Annual International Symposium*. VRAIS '93. Washington, DC, USA: IEEE Computer Society, 1993, pp. 20–26. ISBN: 0-7803-1363-1. DOI: 10.1109/VRAIS.1993.380803.

[Buc+04] V. Buchmann, S. Violich, M. Billinghurst, and A. Cockburn. "FingARtips: Gesture Based Direct Manipulation in Augmented Reality." In: *Proceedings of the 2Nd International Conference on Computer Graphics and Interactive Techniques in Australasia and South East Asia*. GRAPHITE '04. Singapore: ACM, 2004, pp. 212–221. ISBN: 1-58113-883-0. DOI: 10.1145/988834.988871.

[BV18] Boundless Dynamics, LLC and Valve Corporation. *VTOL VR*. https://store.steampowered.com/app/667970/VTOL_VR/. [Accessed: 01-July-2018]. 2018.

[BW01] D. A. Bowman and C. A. Wingrave. "Design and Evaluation of Menu Systems for Immersive Virtual Environments." In: *Proceedings of the Virtual Reality 2001 Conference (VR'01)*. VR '01. Washington, DC, USA: IEEE Computer Society, 2001, pp. 149–. ISBN: 0-7695-0948-7.

[Cal+88] J. Callahan, D. Hopkins, M. Weiser, and B. Shneiderman. "An Empirical Comparison of Pie vs. Linear Menus." In: *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. CHI '88. Washington, D.C., USA: ACM, 1988, pp. 95–100. ISBN: 0-201-14237-6. DOI: 10.1145/57167.57182.

[CCP17] CCP. *Mini Masterclass - Missile Deployment*. https://www.evevalkyrie.com/articles/news/mini-masterclass-missile-deployment. [Accessed: 20-June-2018]. 2017.

[CCP18] CCP. *Eve Valkyrie*. https://www.evevalkyrie.com/. [Accessed: 3-June-2018]. 2018.

[CH02] Y. S. Chee and C. M. Hooi. "C-VISions: Socialized Learning Through Collaborative, Virtual, Interactive Simulations." In: *Proceedings of the Conference on Computer Support for Collaborative Learning: Foundations for a CSCL Community*. CSCL '02. Boulder, Colorado: International Society of the Learning Sciences, 2002, pp. 687–696.

[Chr17] J. Christen. "Reconstructing Vindonissa as a living document–A case-study of digital reconstruction for output to pre-rendered and real-time applications." In: *Studies in Digital Heritage* 1.2 (2017), pp. 396–408. DOI: https://doi.org/https://doi.org/10.14434/sdh.v1i2.23280..

[Con+92] B. D. Conner, S. S. Snibbe, K. P. Herndon, D. C. Robbins, R. C. Zeleznik, and A. van Dam. "Three-dimensional Widgets." In: *Proceedings of the 1992 Symposium on Interactive 3D Graphics*. I3D '92. Cambridge, Massachusetts, USA: ACM, 1992, pp. 183–188. ISBN: 0-89791-467-8. DOI: 10.1145/147156.147199.

[CVD17] Croteam VR, Valve Corporation, and Devolver Digital. *Serious Sam VR: The Last Hope*. https://store.steampowered.com/app/465240/Serious_Sam_VR_The_Last_Hope/. [Accessed: 05-July-2018]. 2017.

[CW15] I. Cho and Z. Wartell. "Evaluation of a bimanual simultaneous 7DOF interaction technique in virtual environments." In: *2015 IEEE Symposium on 3D User Interfaces (3DUI)*. Washington, DC, USA: IEEE Computer Society, Mar. 2015, pp. 133–136. ISBN: 978-1-4673-6886-5. DOI: 10.1109/3DUI.2015.7131738.

[Dac99] R. Dachselt. "The Challenge to Build Flexible User Interface Components for Non-immersive 3D Environments." In: *Proceedings of the HCI International '99 (the 8th International Conference on Human-Computer Interaction) on Human-Computer Interaction: Communication, Cooperation, and Application Design-Volume 2 - Volume 2*. Hillsdale, NJ, USA: L. Erlbaum Associates Inc., 1999, pp. 1055–1059. ISBN: 0-8058-3392-7.

[Dar94] R. P. Darken. "Hands-off interaction with menus in virtual spaces." In: *Proceedings of SPIE*. Vol. 2177. San Jose, CA, United States, 1994, pp. 365–371. DOI: 10.1117/12.173893.

[DC99] R. P. Darken and H. Cevik. "Map Usage in Virtual Environments: Orientation Issues." In: *Proceedings of the IEEE Virtual Reality*. VR '99. Washington, DC, USA: IEEE Computer Society, 1999, pp. 133–. ISBN: 0-7695-0093-5.

[DE01] R. Dachselt and J. Ebert. "Collapsible Cylindrical Trees: A Fast Hierarchical Navigation Technique." In: *Proceedings of the IEEE Symposium on Information Visualization 2001 (INFOVIS'01)*. INFOVIS '01. Washington, DC, USA: IEEE Computer Society, 2001, pp. 79–. ISBN: 0-7695-1342-5.

[Dee95] M. F. Deering. "HoloSketch: A Virtual Reality Sketching/Animation Tool." In: *ACM Trans. Comput.-Hum. Interact.* 2.3 (Sept. 1995), pp. 220–238. ISSN: 1073-0516. DOI: 10.1145/210079.210087.

[DH07] R. Dachselt and A. Hübner. "Virtual Environments: Three-dimensional Menus: A Survey and Taxonomy." In: *Computers & Graphics* 31.1 (Jan. 2007), pp. 53–65. ISSN: 0097-8493. DOI: 10.1016/j.cag.2006.09.006.

[Dog17] A. Dogtiev. *Pokémon GO Revenue and Usage Statistics*. http://www.businessofapps.com/data/pokemon-go-statistics/. [Accessed: 16-January-2018]. 2017.

[DP02] R. P. Darken and B. Peterson. "Spatial orientation, wayfinding, and representation." In: *Handbook of virtual environments: Design, implementation, and applications.* Human factors and ergonomics. Mahwah, NJ, US: Lawrence Erlbaum Associates Publishers, 2002, pp. 493–518. ISBN: 0-8058-3270-X.

[DS93] R. P. Darken and J. L. Sibert. "A Toolset for Navigation in Virtual Environments." In: *Proceedings of the 6th Annual ACM Symposium on User Interface Software and Technology*. UIST '93. Atlanta, Georgia, USA: ACM, 1993, pp. 157–165. ISBN: 0-89791-628-X. DOI: 10.1145/168642.168658.

[Fei+93] S. Feiner, B. MacIntyre, M. Haupt, and E. Solomon. "Windows on the World: 2D Windows for 3D Augmented Reality." In: *Proceedings of the 6th Annual ACM Symposium on User Interface Software and Technology*. UIST '93. Atlanta, Georgia, USA: ACM, 1993, pp. 145–155. ISBN: 0-89791-628-X. DOI: 10.1145/168642.168657.

[FHZ96] A. Forsberg, K. Herndon, and R. Zeleznik. "Aperture Based Selection for Immersive Virtual Environments." In: *Proceedings of the 9th Annual ACM Symposium on User Interface Software and Technology*. UIST '96. Seattle, Washington, USA: ACM, 1996, pp. 95–96. ISBN: 0-89791-798-7. DOI: 10.1145/237091.237105.

[GB04] D. Gerber and D. Bechmann. "Design and evaluation of the ring menu in virtual environments." In: *Proceedings of 8th International Immersive Projection Technology Workshop*. IPT'04, 2004.

[GB05] D. Gerber and D. Bechmann. "The Spin Menu: A Menu System for Virtual Environments." In: *Proceedings of the 2005 IEEE Conference 2005 on Virtual Reality*. VR '05. Washington, DC, USA: IEEE Computer Society, 2005, pp. 271–272. ISBN: 0-7803-8929-8. DOI: 10.1109/VR.2005.81.

[GC01] J. Grosjean and S. Coquillart. "Command & Control Cube: A Shortcut Paradigm for Virtual Environments." In: *Proceedings of the 7th Eurographics Conference on Virtual Environments & 5th Immersive Projection Technology*. EGVE'01. Stuttgart, Germany: Eurographics Association, 2001, pp. 1–12. ISBN: 3-211-83671-3. DOI: 10.2312/EGVE/EGVE01/001-012.

[Gep17] M. Gepp. *Roomscale 101 – An Introduction to Roomscale VR*. https://blog.vive.com/us/2017/10/25/roomscale-101/. [Accessed: 1-February-2018]. 2017.

[GL85] J. D. Gould and C. Lewis. "Designing for Usability: Key Principles and What Designers Think." In: *Commun. ACM* 28.3 (Mar. 1985), pp. 300–311. ISSN: 0001-0782. DOI: 10.1145/3166.3170.

[Gol99] R. G. Golledge. *Wayfinding Behavior: Cognitive mapping and other spatial processes*. Baltimore: JHU press, 1999. ISBN: 9780801859939.

[Gra+02] D. Grammenos, M. Filou, P. Papadakos, and C. Stephanidis. "Virtual Prints: Leaving Trails in Virtual Environments." In: *Proceedings of the Workshop on Virtual Environments 2002*. EGVE '02. Barcelona, Spain: Eurographics Association, 2002, 131–ff. ISBN: 1-58113-535-1.

[Gun16] Gunfiregames, LLC. *Chronos*. http://gunfiregames.com/chronos/. [Accessed: 17-April-2018]. 2016.

[GVH14] A. Giesler, D. Valkov, and K. Hinrichs. "Void Shadows: Multi-touch Interaction with Stereoscopic Objects on the Tabletop." In: *Proceedings of the 2Nd ACM Symposium on Spatial User Interaction*. SUI '14. Honolulu, Hawaii, USA: ACM, 2014, pp. 104–112. ISBN: 978-1-4503-2820-3. DOI: 10.1145/2659766.2659779.

[Ham17] I. Hamilton. *Exclusive: The Well Is A Gorgeous RPG From Turtle Rock Studios*. https://uploadvr.com/the-well-turtle-rock-rpg-oculus-gear-vr/. [Accessed: 5-May-2018]. 2017.

[Han+06] M. S. Hancock, S. Carpendale, F. D. Vernier, D. Wigdor, and C. Shen. "Rotation and Translation Mechanisms for Tabletop Interaction." In: *Proceedings of the First IEEE International Workshop on Horizontal Interactive Human-Computer Systems*. TABLETOP '06. Washington, DC, USA: IEEE Computer Society, 2006, pp. 79–88. ISBN: 0-7695-2494-X. DOI: 10.1109/TABLETOP.2006.26.

[HC11] U. Hinrichs and S. Carpendale. "Gestures in the Wild: Studying Multi-touch Gesture Sequences on Interactive Tabletop Exhibits." In: *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. CHI '11. Vancouver, BC, Canada: ACM, 2011, pp. 3023–3032. ISBN: 978-1-4503-0228-9. DOI: 10.1145/1978942.1979391.

[HH93] D. Hix and H. R. Hartson. *Developing User Interfaces: Ensuring Usability Through Product & Process*. New York, NY, USA: John Wiley & Sons, Inc., 1993. ISBN: 0-471-57813-4.

[Hou92] S. Houde. "Iterative Design of an Interface for Easy 3-D Direct Manipulation." In: *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. CHI '92. Monterey, California, USA: ACM, 1992, pp. 135–142. ISBN: 0-89791-513-5. DOI: 10.1145/142750.142772.

[I+98] P. I., I. T., W. S., and B. M. "Egocentric Object Manipulation in Virtual Environments: Empirical Evaluation of Interaction Techniques." In: *Computer Graphics Forum* 17.3 (1998), pp. 41–52. DOI: 10.1111/1467-8659.00252. eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1111/1467-8659.00252.

[Iga+98] T. Igarashi, R. Kadobayashi, K. Mase, and H. Tanaka. "Path Drawing for 3D Walkthrough." In: *Proceedings of the 11th Annual ACM Symposium on User Interface Software and Technology*. UIST '98. San Francisco, California, USA: ACM, 1998, pp. 173–174. ISBN: 1-58113-034-1. DOI: 10.1145/288392.288599.

[IRA07] V. Interrante, B. Ries, and L. Anderson. "Seven League Boots: A New Metaphor for Augmented Locomotion through Moderately Large Scale Immersive Virtual Environments." In: *2007 IEEE Symposium on 3D User Interfaces*. Washington, DC, USA: IEEE Computer Society, Mar. 2007. ISBN: 1-4244-0907-1. DOI: 10.1109/3DUI.2007.340791.

[Jam18] James. *Annotations*. https://help.sketchfab.com/hc/en-us/articles/202512456-Annotations. [Accessed: 9-June-2018]. 2018.

[JE92] R. H. Jacoby and S. R. Ellis. "Using virtual menus in a virtual environment." In: *Proceedings of SPIE*. Vol. 1668. 1992. DOI: 10.1117/12.59654.

[JF11] J. Jacobs and B. Froehlich. "A Soft Hand Model for Physically-based Manipulation of Virtual Objects." In: *Proceedings of the 2011 IEEE Virtual Reality Conference*. VR '11. Washington, DC, USA: IEEE Computer Society, 2011, pp. 11–18. ISBN: 978-1-4577-0039-2. DOI: 10.1109/VR.2011.5759430.

[KBB11] R. Kopper, F. Bacim, and D. A. Bowman. "Rapid and Accurate 3D Selection by Progressive Refinement." In: *Proceedings of the 2011 IEEE Symposium on 3D User Interfaces*. 3DUI '11. Washington, DC, USA: IEEE Computer Society, 2011, pp. 67–74. ISBN: 978-1-4577-0063-7.

[KH11] S. Knoedel and M. Hachet. "Multi-touch RST in 2D and 3D spaces: Studying the impact of directness on user performance." In: *2011 IEEE Symposium on 3D User Interfaces (3DUI)*. Mar. 2011, pp. 75–78. DOI: 10.1109/3DUI.2011.5759220.

[Kim+00] N. Kim, G. J. Kim, C.-M. Park, I. Lee, and S. H. Lim. "Multimodal Menu Presentation and Selection in Immersive Virtual Environments." In: *Proceedings of the IEEE Virtual Reality 2000 Conference*. VR '00. Washington, DC, USA: IEEE Computer Society, 2000, pp. 281–. ISBN: 0-7695-0478-7. DOI: 10.1109/VR.2000.840509.

[KRF11] A. von Kapri, T. Rick, and S. Feiner. "Comparing Steering-based Travel Techniques for Search Tasks in a CAVE." In: *Proceedings of the 2011 IEEE Virtual Reality Conference*. VR '11. Washington, DC, USA: IEEE Computer Society, 2011, pp. 91–94. ISBN: 978-1-4577-0039-2. DOI: 10.1109/VR.2011.5759443.

[Kru+16] E. Kruijff, A. Marquardt, C. Trepkowski, R. W. Lindeman, A. Hinkenjann, J. Maiero, and B. E. Riecke. "On Your Feet!: Enhancing Vection in Leaning-Based Interfaces Through Multisensory Stimuli." In: *Proceedings of the 2016 Symposium on Spatial User Interaction*. SUI '16. Tokyo, Japan: ACM, 2016, pp. 149–158. ISBN: 978-1-4503-4068-7. DOI: 10.1145/2983310.2985759.

[Kru83] M. Krueger. *Artificial Reality*. Addison-Wesley, 1983. ISBN: 9780201047653.

[Kru91] M. Krueger. *Artificial Reality II*. Polar Research. Addison-Wesley, 1991. ISBN: 9780201522600.

[Lak17] M. Lake. *Echo Arena - Oculus Rift (Ender's game in VR!) - Amazing!* https://www.youtube.com/watch?v=KYlLV8Ifeew. [Accessed: 15-June-2018]. 2017.

[Lan16] B. Lang. *Hands-on: 'Budget Cuts' Inventive Locomotion is a Lesson for VR Developers*. https://www.roadtovr.com/hands-on-budget-cuts-inventive-locomotion-is-a-lesson-for-vr-developers/. [Accessed: 06-July-2018]. 2016.

[LaV+05] J. J. LaViola, D. Bowman, E. Kruijff, and I. P. Poupyrev. *3D User Interfaces: Theory and Practice*. Pearson Education. Boston, MA, USA: Addison-Wesley, 2005. ISBN: 9780201758672.

[LaV+17] J. J. LaViola, D. Bowman, E. Kruijff, and I. P. Poupyrev. *3D User Interfaces: Theory and Practice; 2nd ed.* Boston, MA: Addison-Wesley, 2017.

[LG94] J. Liang and M. Green. "JDCAD: A highly interactive 3D modeling system." In: *Computers & Graphics* 18.4 (1994), pp. 499–506. ISSN: 0097-8493. DOI: https://doi.org/10.1016/0097-8493(94)90062-0.

[LJ60] K. Lynch and Joint Center for Urban Studies. *The Image of the City*. Harvard-MIT Joint Center for Urban Studies Series. Cambridge, MA: MIT Press, 1960. ISBN: 9780262620017.

[Mac13] I. S. MacKenzie. *Human-Computer Interaction: An Empirical Research Perspective*. 1st. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 2013. ISBN: 0124058655, 9780124058651.

[Mar18] Maritime Archaeology Trust. *HMD John Mitchell (1917), Armed Drifter*. `https://sketchfab.com/models/26c020cee6604c47933ec7defe5dbbfc`. [Accessed: 9-June-2018]. 2018.

[Mas17] MasterPiece VR. *Masterpiece VR Press Kit*. `https://www.masterpiecevr.com/press`. [Accessed: 25-May-2018]. 2017.

[MB95] T. Munzner and P. Burchard. "Visualizing the Structure of the World Wide Web in 3D Hyperbolic Space." In: *Proceedings of the First Symposium on Virtual Reality Modeling Language*. VRML '95. San Diego, California, USA: ACM, 1995, pp. 33–38. ISBN: 0-89791-818-5. DOI: 10.1145/217306.217311.

[MBS97] M. R. Mine, F. P. Brooks Jr., and C. H. Sequin. "Moving Objects in Space: Exploiting Proprioception in Virtual-environment Interaction." In: *Proceedings of the 24th Annual Conference on Computer Graphics and Interactive Techniques*. SIGGRAPH '97. New York, NY, USA: ACM Press/Addison-Wesley Publishing Co., 1997, pp. 19–26. ISBN: 0-89791-896-7. DOI: 10.1145/258734.258747.

[McM+12] R. P. McMahan, D. A. Bowman, D. J. Zielinski, and R. B. Brady. "Evaluating Display Fidelity and Interaction Fidelity in a Virtual Reality Game." In: *IEEE Transactions on Visualization and Computer Graphics* 18.4 (Apr. 2012), pp. 626–633. ISSN: 1077-2626. DOI: 10.1109/TVCG.2012.43.

[Met17a] Meta. *Meta Workspace: Rethink the Way You Work*. `https://www.youtube.com/watch?v=-KIgvc-LVDs`. [Accessed: 6-June-2018]. 2017.

[Met17b] Meta Company. *Meta | Augmented Reality*. `http://www.metavision.com/`. [Accessed: 27-May-2018]. 2017.

[Mic16] Microsoft Hololens. *Microsoft HoloLens: Skype*. `https://www.youtube.com/watch?v=4QiGYtd3qNI`. [Accessed: 6-June-2018]. 2016.

[Mic18] Microsoft. *AltspaceVR*. `https://altvr.com/`. [Accessed: 3-June-2018]. 2018.

[Mil+95] P. Milgram, H. Takemura, A. Utsumi, and F. Kishino. "Augmented reality: A Class of Displays on the Reality-Virtuality Continuum." In: *Telemanipulator and telepresence technologies*. Vol. 2351. International Society for Optics and Photonics. 1995, pp. 282–293.

[Min95a] M. R. Mine. *ISAAC: A Virtual Environment Tool for the Interactive Construction of Virtual Worlds*. Tech. rep. Chapel Hill, NC, USA: University of North Carolina at Chapel Hill, June 1995.

[Min95b] M. R. Mine. *Virtual Environment Interaction Techniques*. Tech. rep. Chapel Hill, NC, USA, 1995.

[MM95] D. P. Mapes and J. M. Moshell. "A Two-handed Interface for Object Manipulation in Virtual Environments." In: *Presence: Teleoper. Virtual Environ.* 4.4 (Jan. 1995), pp. 403–416. ISSN: 1054-7460. DOI: 10.1162/pres.1995.4.4.403.

[MMK12] M. Mori, K. F. MacDorman, and N. Kageki. "The Uncanny Valley [From the Field]." In: *IEEE Robotics Automation Magazine* 19.2 (June 2012), pp. 98–100. ISSN: 1070-9932. DOI: 10.1109/MRA.2012.2192811.

[MN90] R. Molich and J. Nielsen. "Improving a Human-computer Dialogue." In: *Commun. ACM* 33.3 (Mar. 1990), pp. 338–348. ISSN: 0001-0782. DOI: 10.1145/77481.77486.

[NAS95] NASA. *Interface Widgets in the Virtual Windtunnel*. https://www.nas.nasa.gov/Software/VWT/widgets.html. [Accessed: 30-May-2018]. 1995.

[Nel80] T. Nelson. "Interactive systems and the design of Virtuality." In: *Creative Computing* 6.11 (1980), pp. 56–62.

[NM90] J. Nielsen and R. Molich. "Heuristic Evaluation of User Interfaces." In: *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. CHI '90. Seattle, Washington, USA: ACM, 1990, pp. 249–256. ISBN: 0-201-50932-6. DOI: 10.1145/97243.97281.

[Ocu18] Oculus VR, LLC. *Parties*. https://developer.oculus.com/documentation/platform/latest/concepts/dg-cc-parties/. [Accessed: 6-June-2018]. 2018.

[OF03] A. Olwal and s. Feiner. "The Flexible Pointer: An Interaction Technique for Selection in Augmented and Virtual Reality." In: *UIST '03: Adjunct Proceedings of the 16th Annual ACM Symposium on User Interface Software and Technology*. Vancouver, Canada: ACM, Nov. 2003, pp. 81–82.

[Ort+16] F. R. Ortega, F. Abyarjoo, A. Barreto, N. Rishe, and M. Adjouadi. *Interaction Design for 3D User Interfaces: The World of Modern Input Devices for Research, Applications, and Game Development*. Natick, MA, USA: A. K. Peters, Ltd., 2016. ISBN: 1482216949, 9781482216943.

[Pau+95] R. Pausch, T. Burnette, D. Brockway, and M. 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*. SIGGRAPH '95. New York, NY, USA: ACM, 1995, pp. 399–400. ISBN: 0-89791-701-4. DOI: 10.1145/218380.218495.

[PB12] M. Prachyabrued and C. W. Borst. "Virtual Grasp Release Method and Evaluation." In: *Int. J. Hum.-Comput. Stud.* 70.11 (Nov. 2012), pp. 828–848. ISSN: 1071-5819. DOI: 10.1016/j.ijhcs.2012.06.002.

[Pie+97] J. S. Pierce, A. S. Forsberg, M. J. Conway, S. Hong, R. C. Zeleznik, and M. R. Mine. "Image Plane Interaction Techniques in 3D Immersive Environments." In: *Proceedings of the 1997 Symposium on Interactive 3D Graphics*. I3D '97. Providence, Rhode Island, USA: ACM, 1997, 39–ff. ISBN: 0-89791-884-3. DOI: 10.1145/253284.253303.

[Pie+99] J. S. Pierce, M. Conway, M. van Dantzich, and G. Robertson. "Toolspaces and Glances: Storing, Accessing, and Retrieving Objects in 3D Desktop Applications." In: *Proceedings of the 1999 Symposium on Interactive 3D Graphics*. I3D '99. Atlanta, Georgia, USA: ACM, 1999, pp. 163–168. ISBN: 1-58113-082-1. DOI: 10.1145/300523.300545.

[Pol+92] P. G. Polson, C. Lewis, J. Rieman, and C. Wharton. "Cognitive walkthroughs: a method for theory-based evaluation of user interfaces." In: *International Journal of man-machine studies* 36.5 (1992), pp. 741–773.

[Pol16] Polygon. *Chronos Oculus Rift Gameplay*. https://www.youtube.com/watch?v=u47uN544HYQ. [Accessed: 17-April-2018]. 2016.

[Pou+02] I. Poupyrev, D. S. Tan, M. Billinghurst, H. Kato, H. Regenbrecht, and N. Tetsutani. "Developing a Generic Augmented-Reality Interface." In: *Computer* 35.3 (Mar. 2002), pp. 44–50. ISSN: 0018-9162. DOI: 10.1109/2.989929.

[Pou+96] I. Poupyrev, M. Billinghurst, S. Weghorst, and T. 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*. UIST '96. Seattle, Washington, USA: ACM, 1996, pp. 79–80. ISBN: 0-89791-798-7. DOI: 10.1145/237091.237102.

[PRS97] B. Preim, A. Raab, and T. Strothotte. "Coherent Zooming of Illustrations with 3D-graphics and Text." In: *Proceedings of the Conference on Graphics Interface '97*. Kelowna, British Columbia, Canada: Canadian Information Processing Society, 1997, pp. 105–113. ISBN: 0-9695338-6-1.

[PS96] T. Poston and L. Serra. "Dextrous Virtual Work." In: *Commun. ACM* 39.5 (May 1996), pp. 37–45. ISSN: 0001-0782. DOI: 10.1145/229459.229464.

[PSP99] J. S. Pierce, B. C. Stearns, and R. Pausch. "Voodoo Dolls: Seamless Interaction at Multiple Scales in Virtual Environments." In: *Proceedings of the 1999 Symposium on Interactive 3D Graphics*. I3D '99. Atlanta, Georgia, USA: ACM, 1999, pp. 141–145. ISBN: 1-58113-082-1. DOI: 10.1145/300523.300540.

[PT02] W. Piekarski and B. H. Thomas. "The Tinmith System: Demonstrating New Techniques for Mobile Augmented Reality Modelling." In: *Proceedings of the Third Australasian Conference on User Interfaces - Volume 7*. AUIC '02. Melbourne, Victoria, Australia: Australian Computer Society, Inc., 2002, pp. 61–70. ISBN: 0-909925-85-2.

[Raz+02] S. Razzaque, D. Swapp, M. Slater, M. C. Whitton, and A. Steed. "Redirected Walking in Place." In: *Proceedings of the Workshop on Virtual Environments 2002*. EGVE '02. Barcelona, Spain: Eurographics Association, 2002, pp. 123–130. ISBN: 1-58113-535-1.

[Rea18] Ready At Dawn. *Echo Arena*. http://www.readyatdawn.com/game-list/echo-arena/. [Accessed: 15-June-2018]. 2018.

[RFR95] J. Rieman, M. Franzke, and D. Redmiles. "Usability evaluation with the cognitive walkthrough." In: *Conference companion on Human factors in computing systems*. ACM. 1995, pp. 387–388.

[RJ07] A. R. Rivers and D. L. James. "FastLSM: Fast Lattice Shape Matching for Robust Real-time Deformation." In: *ACM SIGGRAPH 2007 Papers*. SIGGRAPH '07. San Diego, California: ACM, 2007. DOI: 10.1145/1275808.1276480.

[RMC91] G. G. Robertson, J. D. Mackinlay, and S. K. Card. "Cone Trees: Animated 3D Visualizations of Hierarchical Information." In: *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. CHI '91. New Orleans, Louisiana, USA: ACM, 1991, pp. 189–194. ISBN: 0-89791-383-3. DOI: 10.1145/108844.108883.

[Rob+00] G. Robertson, M. van Dantzich, D. Robbins, M. Czerwinski, K. Hinckley, K. Risden, D. Thiel, and V. Gorokhovsky. "The Task Gallery: A 3D Window Manager." In: *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. CHI '00. The Hague, The Netherlands: ACM, 2000, pp. 494–501. ISBN: 1-58113-216-6. DOI: 10.1145/332040.332482.

[RUS18] RUST LTD. *Hot Dogs, Horseshoes & Hand Grenades*. http://www.h3vr.com/. [Accessed: 11-June-2018]. 2018.

[San16] Sanctuary Media LLC. *RogueVR - Roguelike Virtual Reality*. http://www.indiedb.com/games/roguevr-roguelike-virtual-reality. [Accessed: 5-May-2018]. 2016.

[SC03] W. Sherman and A. Craig. *Understanding Virtual Reality: Interface, Application, and Design*. Morgan Kaufmann Series in Computer Graphics and Geometric Modeling. Morgan Kaufmann, 2003. ISBN: 9781558603530.

[Sch+12] U. Schultheis, J. Jerald, F. Toledo, A. Yoganandan, and P. Mlyniec. "Comparison of a two-handed interface to a wand interface and a mouse interface for fundamental 3D tasks." In: *2012 IEEE Symposium on 3D User Interfaces (3DUI)*. Washington, DC, USA: IEEE Computer Society, Mar. 2012, pp. 117–124. ISBN: 978-1-4673-1204-2. DOI: 10.1109/3DUI.2012.6184195.

[Sci15] D. Scimeca. *We grabbed an Oculus Touch, and hopped aboard the Bullet Train*. https://www.dailydot.com/parsec/bullet-train-demo-oculus-touch-connect-2/. [Accessed: 06-July-2018]. 2015.

[SCP95] R. Stoakley, M. J. Conway, and R. Pausch. "Virtual Reality on a WIM: Interactive Worlds in Miniature." In: *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. CHI '95. Denver, Colorado, USA: ACM Press/Addison-Wesley Publishing Co., 1995, pp. 265–272. ISBN: 0-201-84705-1. DOI: 10.1145/223904.223938.

[Scr67] M. S. Scriven. "The Methodology of Evaluation." In: *Perspectives of Curriculum Evaluation*. Ed. by R. E. Stake. American Educational Research Association, 1967, pp. 39–83.

[Ser+95] L. Serra, T. Poston, N. Hern, C. Beng Choon, and J. Waterworth. "Interaction Techniques for a Virtual Workspace." In: *ICAT/VRST'95, International Conference on Artificial Reality and Tele-Existence/Conference on Virtual Reality Software and Technology : November 21 - 22, 1995, Makuhari Messe, Chiba, Japan*. Chiba, Japan: Nihon Keizai Shimbun, Nov. 1995, pp. 221–230.

[SG97] Z. Szalavári and M. Gervautz. "The Personal Interaction Panel – a Two-Handed Interface for Augmented Reality." In: *Computer Graphics Forum* 16.3 (1997), pp. C335–C346. DOI: 10.1111/1467-8659.00137. eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1111/1467-8659.00137.

[SH16] D. Schmalstieg and T. Hollerer. *Augmented Reality: Principles and Practice*. Pearson Education. Addison-Wesley Professional, 2016. ISBN: 9780133153200.

[Sim16] A. L. Simeone. "Indirect touch manipulation for interaction with stereoscopic displays." In: *2016 IEEE Symposium on 3D User Interfaces (3DUI)*. Mar. 2016, pp. 13–22. DOI: 10.1109/3DUI.2016.7460025.

[SM00] S. D. Steck and H. A. Mallot. "The Role of Global and Local Landmarks in Virtual Environment Navigation." In: *Presence: Teleoper. Virtual Environ.* 9.1 (Feb. 2000), pp. 69–83. ISSN: 1054-7460. DOI: 10.1162/105474600566628.

[Ste+10] F. Steinicke, G. Bruder, J. Jerald, H. Frenz, and M. Lappe. "Estimation of Detection Thresholds for Redirected Walking Techniques." In: *IEEE Transactions on Visualization and Computer Graphics* 16.1 (Jan. 2010), pp. 17–27. ISSN: 1077-2626. DOI: 10.1109/TVCG.2009.62.

[Ste18] Steel Crate Games. *Keep Talking And Nobody Explodes*. http://www.keeptalkinggame.com/. [Accessed: 30-May-2018]. 2018.

[Sum+12] E. A. Suma, Z. Lipps, S. Finkelstein, D. M. Krum, and M. Bolas. "Impossible Spaces: Maximizing Natural Walking in Virtual Environments with Self-Overlapping Architecture." In: *IEEE Transactions on Visualization and Computer Graphics* 18.4 (Apr. 2012), pp. 555–564. ISSN: 1077-2626. DOI: 10.1109/TVCG.2012.47.

[Sut65] I. Sutherland. "The Ultimate Display." In: *Proceedings of IFIP Congress 65*. 1965, pp. 506–508.

[SVH11] S. Strothoff, D. Valkov, and K. Hinrichs. "Triangle Cursor: Interactions with Objects Above the Tabletop." In: *Proceedings of the ACM International Conference on Interactive Tabletops and Surfaces*. ITS '11. Kobe, Japan: ACM, 2011, pp. 111–119. ISBN: 978-1-4503-0871-7. DOI: 10.1145/2076354.2076377.

[Swe94] M. Swed. "Virtually Real Virtual Reality: Computer expert and virtual reality prophet Jaron Lanier is making new connections between previously unconnected musical cultures–but on real instruments, not synthetic ones." In: *Los Angeles Times* (Sept. 11, 1994).

[TML13] A. Talvas, M. Marchal, and A. Lécuyer. "The god-finger method for improving 3D interaction with virtual objects through simulation of contact area." In: *2013 IEEE Symposium on 3D User Interfaces (3DUI)*. Washington, DC, USA: IEEE Computer Society, Mar. 2013, pp. 111–114. ISBN: 978-1-4673-6097-5. DOI: 10.1109/3DUI.2013.6550206.

[Ubi18] Ubisoft. *Eagle Flight*. https://www.ubisoft.com/de-de/game/eagle-flight/. [Accessed: 3-June-2018]. 2018.

[Uli+09] A. C. Ulinski, Z. Wartell, P. Goolkasian, E. A. Suma, and L. F. Hodges. "Selection Performance Based on Classes of Bimanual Actions." In: *Proceedings of the 2009 IEEE Symposium on 3D User Interfaces*. 3DUI '09. Washington, DC, USA: IEEE Computer Society, 2009, pp. 51–58. ISBN: 978-1-4244-3965-2. DOI: 10.1109/3DUI.2009.4811205.

[Val18] Valve Corporation. *SteamVR*. https://store.steampowered.com/steamvr. [Accessed: 20-June-2018]. 2018.

[Vas+13] K. Vasylevska, H. Kaufmann, M. Bolas, and E. A. Suma. "Flexible spaces: Dynamic layout generation for infinite walking in virtual environments." In: *2013 IEEE Symposium on 3D User Interfaces (3DUI)*. Washington, DC, USA: IEEE Computer Society, Mar. 2013, pp. 39–42. DOI: 10.1109/3DUI.2013.6550194.

[Vin99] N. G. Vinson. "Design Guidelines for Landmarks to Support Navigation in Virtual Environments." In: *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. CHI '99. Pittsburgh, Pennsylvania, USA: ACM, 1999, pp. 278–285. ISBN: 0-201-48559-1. DOI: 10.1145/302979.303062.

[VR 18] VR Nerds. *Meta 2 AR-Brille im Test*. https://www.youtube.com/watch?v=Kg1DQH_rClA. [Accessed: 27-May-2018]. 2018.

[VRC18] VRChat Inc. *VRChat*. https://www.vrchat.net/. [Accessed: 6-June-2018]. 2018.

[WBB06] H. P. Wyss, R. Blach, and M. Bues. "iSith - Intersection-based Spatial Interaction for Two Hands." In: *Proceedings of the IEEE Conference on Virtual Reality*. VR '06. Washington, DC, USA: IEEE Computer Society, 2006, pp. 123–. ISBN: 1-4244-0224-7. DOI: 10.1109/VR.2006.93.

[Wes03] G. Wesche. "The ToolFinger: Supporting Complex Direct Manipulation in Virtual Environments." In: *Proceedings of the Workshop on Virtual Environments 2003*. EGVE '03. Zurich, Switzerland: ACM, 2003, pp. 39–45. ISBN: 1-58113-686-2. DOI: 10.1145/769953.769958.

[WG95] M. M. Wloka and E. Greenfield. "The Virtual Tricorder: A Uniform Interface for Virtual Reality." In: *Proceedings of the 8th Annual ACM Symposium on User Interface and Software Technology*. UIST '95. Pittsburgh, Pennsylvania, USA: ACM, 1995, pp. 39–40. ISBN: 0-89791-709-X. DOI: 10.1145/215585.215647.

[WHB06] C. A. Wingrave, Y. Haciahmetoglu, and D. A. Bowman. "Overcoming World in Miniature Limitations by a Scaled and Scrolling WIM." In: *Proceedings of the 3D User Interfaces*. 3DUI '06. Washington, DC, USA: IEEE Computer Society, 2006, pp. 11–16. ISBN: 1-4244-0225-5. DOI: 10.1109/VR.2006.106.

[WMW09] J. O. Wobbrock, M. R. Morris, and A. D. Wilson. "User-defined Gestures for Surface Computing." In: *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. CHI '09. Boston, MA, USA: ACM, 2009, pp. 1083–1092. ISBN: 978-1-60558-246-7. DOI: 10.1145/1518701.1518866.

[WO90] C. Ware and S. Osborne. "Exploration and Virtual Camera Control in Virtual Three Dimensional Environments." In: *Proceedings of the 1990 Symposium on Interactive 3D Graphics*. I3D '90. Snowbird, Utah, USA: ACM, 1990, pp. 175–183. ISBN: 0-89791-351-5. DOI: 10.1145/91385.91442.

[WPA96] M. Wells, B. Peterson, and J. Aten. "The virtual motion controller: A sufficient-motion walking simulator." In: *Proceedings of the 1996 IEEE Virtual Reality Annual International Symposium (VRAIS '96)*. Vol. 97. Washington, DC, USA: IEEE Computer Society, 1996, pp. 1–8.

[ZBM94] S. Zhai, W. Buxton, and P. Milgram. "The &Ldquo;Silk Cursor&Rdquo;: Investigating Transparency for 3D Target Acquisition." In: *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. CHI '94. Boston, Massachusetts, USA: ACM, 1994, pp. 459–464. ISBN: 0-89791-650-6. DOI: 10.1145/191666.191822.

[Zel+02] R. C. Zeleznik, J. J. LaViola Jr, D. A. Feliz, and D. F. Keefe. "Pop Through Button Devices for VE Navigation and Interaction." In: *Proceedings of the IEEE Virtual Reality Conference 2002*. VR '02. Washington, DC, USA: IEEE Computer Society, 2002, pp. 127–. ISBN: 0-7695-1492-8.

[ZS95] C. B. Zilles and J. K. Salisbury. "A Constraint-based God-object Method for Haptic Display." In: *Proceedings of the International Conference on Intelligent Robots and Systems-Volume 3 - Volume 3*. IROS '95. Washington, DC, USA: IEEE Computer Society, 1995, pp. 3146–. ISBN: 0-8186-7108-4.

[Zug18] Zugara, Inc. *ZugSTAR*. https://zugara.com/augmented-reality-and-virtual-reality-technology/augmented-reality-video-conferencing. [Accessed: 6-June-2018]. 2018.