GESTURE CONTROLLED TELEOPERATION

A Thesis submitted in the partial fulfillment of the requirements for the award of the degree of

MASTER OF TECHNOLOGY

(INFORMATION SYSTEMS)

Submitted By:

MANEESH DISODIA

(2K12/ISY/16)

Under the Guidance of

DR. N. S. RAGHAVA

Associate Professor

DEPARTMENT OF INFORMATION TECHNOLOGY
DELHI TECHNOLOGICAL UNIVERSITY
BAWANA ROAD, DELHI-110042
SESSION: 2012-2014

CERTIFICATE

This is to certify that **Maneesh Disodia (2K12/ISY/16)** has carried out the major project titled **"GESTURE CONTROLLED TELEOPERATION"** in partial fulfillment of the requirements for the award of Master of Technology degree in Information Systems by **Delhi Technological University**.

The major project is a bonafide piece of work carried out and completed under my supervision and guidance during the academic session **2012-2014**. To the best of my knowledge, the matter embodied in the thesis has not been submitted to any other University / Institute for the award of any Degree or Diploma.

Dr. N. S. Raghava

Associate Professor

Department of Information Technology

Delhi Technological University

Delhi-110042

ACKNOWLEDGEMENT

I take the opportunity to express my sincere gratitude to my project mentor Dr. N.S. Raghava, Associate Professor, Department of Information Technology, Delhi Technological University, Delhi for providing valuable guidance and constant encouragement throughout the project .It is my pleasure to record my sincere thanks to him for his constructive criticism and insight without which the project would not have shaped as it has.

I humbly extend my words of gratitude to other faculty members of this department for providing their valuable help and time whenever it was required.

Maneesh Disodia

Roll No. 2K12/ISY/16

M.Tech (Information Systems)

E-mail: maneeshdisodia@gmail.com

ABSTRACT

This work represents the development of gesture based control over large distance as an interface to integrate various control signal. Here we are integrating user interface framework based on various current input device. The purpose is to tele operate a puppet (various servo) using human gesture as a controlee. The advancement in robotics and mechanics and software engineering, along with human ergonomics, allowed us to take control real world entity as an object to control real world coordinate. By simulating real condition like human based mundane task as a stored procedure after that we can execute that task to make thing happen later on. Now we are controlling objects by joystick UI or over internet. But in real world control are like haptic feedback are more constraints towards perfection in control mechanics. Video game developer are using gesture tracking to adapt new game design technique. Also various movies are also embedded in gesture to make animation, so here we are representing gesture as a control sense. Open source and hardware have acquired a much essential place in the today's technological word. The development and revolution of open source is very fast as the blue prints are spread wide world and changes in the further development is very easy. This revolutionary step has tended all the experts and new projects to align towards the open source and open hardware environment. The aim of this experiment is to control the real time objects acquiring gestures from the user sitting at one end of the world and the object in any other part separated by distance but connected by internet. Gestures from one device are sent to other the device sending the gestures in form of signals is the server and the one receiving it is the client located at any remote place. Gesture controlling as various future uses like it can become a less bandwidth requiring telephony system for deaf people as communication by webcams need high speed data transfer rate which is not required in gesture communication and

saving bandwidth can increase the data transfer rate. Yet another use can be in the field of medicine where minute operations which are carried on by the robots can be controlled at very miniature level as the interaction is based on the coordinate transfer which would in turn reduce the error of any kind. Time utilization sitting anywhere can be done by communicating with your device which will also reduce the work pressure in the future. So, giving input to devices through our gesture and controlling their work is the coming future of human race.

TABLE OF CONTENTS

FIGURES AND TABLES

CHAPTER 1

INTRODUCTION

1.1 INTRODUCTION

The development of immersive virtual environments (VEs) debuted in the late 1960s, when Ivan Sutherland presented his prototype using real-time 3D rendering and a tracked head-mounted display. Since then, VEs has developed and been applied through a vast range of application areas, such as simulations, architecture, psychotherapy, entertainment, data visualization and education. One important aspect of VEs is the user interface (UI), through which the user interacts with the application. Such interfaces make use of various sensing (input) and output systems, which during the last years have become affordable i.e. through the availability of depth sensing cameras, lightweight head-mounted displays or projectors. These interfaces may include the users head motion, hand motion and even full-body motion to provide input for endless possibilities of interaction techniques (ITs).

In Human-Computer Interaction (HCI) research much effort has dealt with how to utilize such input for 3D interaction in VEs. One of many goals in this research is to allow for natural interaction, using human motor skills and real-world experiences while letting users focus on their tasks rather than operating the system (Froehlich & Bowman, 2009). 3D interfaces for VEs share the same principles found in many current HCI areas, such as in intelligent user interfaces, ambient intelligence, ubiquitous computing or natural-user interfaces. The shared paradigm for these interfaces is to augment the human intellect, to support thinking - rather than taking away cognitive resources for handling the system (Shneiderman, 2005). The development of such interfaces therefore draws on knowledge from many disciplines, such as psychology, cognition and perception, building on top of users pre-existing knowledge of the world. Often themes of reality are employed to guide the design, e.g. the use of naïve physics, incorporating the users' body, the surrounding environment and other people (Jacob et al., 2008). Especially in the design of VEs and 3D interfaces the usage of such guidelines and principles together with a fundamental understanding of the users cognitive capacities, is of great importance. The inherit complexity of VEs often take much of the users attention mostly in the visual domain. Designing 3D UIs on an established paradigm as e.g. direct manipulation (Shneiderman, 1983) found in most GUI / WIMP systems may thus result in misusing cognitive resources on handling the system rather than supporting the user in focusing on the tasks at hand. The classical motto of "What you see is what you get" (WYSIWYG) needs rethinking and the current trends in 3D UI design focus on the motto "What you do is what happens" (Kulik, 2009). Thus the focus shifts from visual representation to embodied interaction embracing direct motor skills in the design of ITs.

Research concerning interaction for VEs has since the late 1990's focused on defining formal frameworks and guidelines for the design of ITs. Bowman identified in his work (Bowman, 1998) three task categories in which most VE interactions fall: *Navigation, selection* and *manipulation*. In the last decade much effort in the UI community has dealt with further developing, identifying, describing and categorizing fundamental principles for designing these ITs. Our intention with the efforts and work in this master thesis is to further support the 3D UI research, by focusing on the controller free interaction possibilities in VEs. Considering most recent developments in affordable sensing hardware like depth cameras e.g. (Kinect, 2010) or (PrimeSensor, 2011) the trend of body controlled interfaces has been introduced to enter the living room. The applications currently supporting such hardware mainly focus on entertainment e.g. (Kinect Games, 2010). It is the intention of our work to utilize such affordable hardware and investigate the possibilities in the broader context of 3D user interfaces and Virtual Environments.

Issues regarding the deployment of required hardware, which in many cases is costly and not easy transportable and additionally may require maintenance staff, may be one of the reasons hindering the broad usage of VEs (Drettakis et al., 2007). A development which we think will change dramatically in the near future partly through the ongoing availability of affordable hardware, but more important through the development of alternative interaction possibilities which focus on users' needs and which are carefully designed to help in thinking.

Though body controlled interactions have gained large popularity in current entertainment applications, the use of such interactions in tools and applications is almost exclusively limited to research. In this thesis we will focus on identifying possible reasons and describe solutions for the use of body controlled interactions in VEs. The benefits of VEs are manifold and some reasons are among others the enhanced sense of presence (Slater & Steed, 2000) and enhanced spatial understanding (Schuchardt & Bowman, 2007) (Phillips et al., 2010). Using body controlled interaction may even benefit in usability and learning (Antle et al., 2009).

The motivation for the work presented throughout this thesis was inspired by the release of the Microsoft Kinect, which in the last months inspired a variety of developers around the globe to do alternative projects in all kind of directions. With this hardware the motivation is to investigate its possibilities as interaction device for VEs; an interaction device which does not require the user to wear heavy equipment wired with cables and which is affordable providing possible usage in future outside the lab applications.

In the following pre-analysis, we will revisit the literature to identify possible shortcomings and to establish an understanding of the state of the art in 3D interaction and VEs, starting with an investigation of the principles behind 3D interfaces, going to a review of ITs used in VEs. As a guiding problem for this pre-analysis the following initial problem statement is used:

Which interaction techniques are commonly used in Virtual Environments and what are the principles behind them?

1.2 PROBLEM STATEMENT

1.2.1 GOALS, FOCUS AND DELIMITATION

The intention in this thesis is to develop a theoretical model which we use as the bases for designing ITs. Also we want to present principles which unify the presented concepts and are useful in the design of controller free interactions. We will limit this investigation on visual selection tasks as we see it as the core task for controlling 3D user interfaces. In this way a selection task precede almost all manipulation tasks and some navigation tasks, and is one of the fundamental interaction techniques reoccurring when designing interfaces. Driven by the motivation of developing a cable free and low-cost VE interface, we will focus on body controlled selection interactions using optical tracking (Kinect) and stereoscopic screen based output for the implementation of ITs. Within this focus on selection we will exclude local selection, and focus on at a distance selection, since the local selection requires navigation.

With this work we wish to contribute with a framework and specific usability evaluated ITs which can be used by other developers of novel body controlled interfaces. From these ITs we also wish

to generalize the experiences with developing and the results from evaluating them in form of a list of guidelines usable for body controlled interfaces.
The goals for this thesis can be summarized as the following:

- Development of a theoretical model
- Applying the model in the design of interaction techniques
- Evaluation of implemented interaction techniques
- Generalization in form of guidelines

1.2.2 PROBLEM STATEMENT

- Based on the investigation throughout the pre-analysis and the goals set for this thesis the following problem statement is formulated:
- Which body controlled interaction techniques are best suited for single and multiple selection of screen content, using optical marker less motion capture (Kinect and NITE) and which design elements constitute the success of a technique?
- This problem statement serves as the main focus of the ongoing investigation throughout the rest of this thesis. To clarify the meaning an explanation of the elements in the problem statement follows.
- When writing "best suited" we refer to the usability of the interaction techniques. Since usability has multiple components, which are defined differently throughout the literature, the definition of Jacob Nielsen, who described usability based on five attributes, is used. These attributes are: learnability, efficiency, memorability, errors and satisfaction (Nielsen, 1993). Additionally the comfort attribute will be part of this definition, since comfort is important for body controlled interaction. A detailed description of these parts constituting to the usability is included in the next section, the methodology.
- Another important factor to clarify at this point is that usability is mostly evaluated and understood in the context of a target group. In this way are the attributes weighted differently based on the intended user of a system. Since the problem targeted in this thesis deals with the fundamentals of selection techniques, and by this does not have a specific end product or target group, it was chosen to concentrate on novice users. This choice is based on the

assumption that most people have no experience with body controlled interfaces and by this novice users may be the greatest part of the target group of future systems using the techniques developed in this project.

The last part of the problem statement: “which design elements constitute to the success of a technique” should be understood as the intended outcome of the evaluation, which are design guidelines intended to help designers to understand and develop future system using body controlled interaction techniques.

CHAPTER 2

LITERATURE SURVEY

2.1 PRE-ANALYSIS

The pre-analysis will start with a broad perspective on interaction design, investigating the subjects of principle based design, the affordances that are needed to have a good interaction design and perceptional and human cognition consideration in regards to VEs.

In order to finalize this thesis focus, research in the area of interaction techniques will be presented and analyzed, the structure of this section has been inspired by Bowman & colleagues task taxonomy of interaction techniques (Bowman et al., 2005).

Since the focus of the thesis is towards interaction techniques that utilize Microsoft's Kinect depth camera together with the NITE tracking framework, the chosen input and output sources will be described and analyzed in regards to the design consequences.

The pre-analysis chapter will have a closing discussion concerning the presented research and derive to this thesis focused problem statement of an investigation into interaction techniques in VEs.

2.1.1 INTERACTION PRINCIPLES

The description and definition of principles with the intention to guide and inform the design of human computer interactions is of great importance for successful design. The amount of literature on formal design guidelines is vast, whereas only few regard the higher principles governing these guidelines. Principles can be seen as the fundamental pillars holding the more specific guidelines. Dennis Wixon (Microsoft Researcher) commented in his talk on principles and guidelines at the UX Week 2008 as the following:

"The principles are what drive the design. The guidelines are simply specific derivations of the principles for an individual context. Principals are what are important. Principals and data drive successful design." (Wixon, 2008)

The principles can be used to guide every phase of the design process; they can help generating ideas and concepts. These principles can be seen as the taxonomy of the core ideas driving the design.
In this section we will have a look on some of the principles that can be applied to VEs: *Realitybased interaction* (Jacob et al., 2008) and *Imagination-based interaction* (Kulik, 2009).

2.1.2 REALITY-BASED INTERACTION

With their RBI framework Robert Jacob and colleagues (Jacob et al., 2008) cover many emerging interaction styles found in present post-WIMP interfaces. The foundation of this framework is that many of these interaction styles draw their strengths by building on users pre-existing knowledge of the real-world. Four RBI themes are used as the basis for their framework: *Naïve physics, Body awareness & skills, Environment awareness & skills and Social awareness & skills*.
Basing interactions on *naïve physics* means to take advantage of the users' common sense knowledge about the physical world. This includes concepts like gravity, friction, velocity, the persistence of objects and their relative scale (Jacob et al., 2008). In relation to VEs this concept can be directly transferred by using novel physics engines, which simulate such concepts.
Body awareness and skills refers to the familiarity and understanding that people have of their own bodies, independent of the environment (Jacob et al., 2008). This concept draws it strengths from proprioception; a user is aware of the position of limbs and the range of motion. Interactions can be built on this strength including the users' body in the interactions, using two-handed or even whole body interaction. This concept is especially important in VEs and many of the interaction styles found for navigating, selecting and manipulating are employing it.
Environment Awareness and skills builds on the concept that humans are aware of their physical presence in a real world environment, surrounded by objects and landscape (Jacob et al., 2008). The clues provided by the surrounding environment facilitate a human's sense of orientation and spatial understanding. Such clues include for example depth clues like shadow, lighting, fog, atmospheric color and horizon. In VEs a common usage is e.g. reference object or landmarks to provide users with an understanding of position and distance. Also modern graphics engines are able to provide for real-time high detailed shadows and lighting to provide consistent clues for spatial awareness. For VE interactions the user's location and orientation in the environment can be further utilized to e.g. display information corresponding to this position. This concept also relates to the spatial

awareness of objects, which in turn can be altered or manipulated by users, thus including skills such as picking up, positioning and arranging objects, which also draw concepts from the body awareness and naïve physics.

Social awareness and skills relates to social interaction skills and a user's general awareness of the presence of others. This concept includes verbal and non-verbal communication, the ability to exchange physical objects, and the ability of collaboration (Jacob et al., 2008). Such interaction styles can be supported by direct co located communication or mediated communication. In VEs there is possibility for both, though the use of technology has a big influence as e.g. HMDs mostly not allow for direct visual communication or contact. Often the use of either video images or even avatars is used in such environments to support communication, whereas the latter can support the communication by making the users actions visible through e.g. whole body interaction.

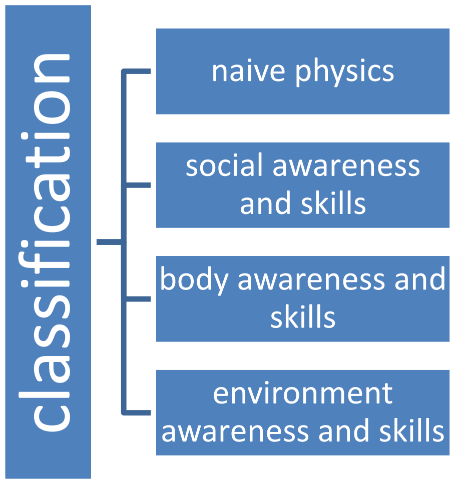

classification
naive physics
social awareness and skills
body awareness and skills
environment awareness and skills

Figure 1 - Elements of Reality-based Interaction framework

In their work Jacob et al. also describe some of the tradeoffs that reality based interaction adheres. The mimicking of reality is not always desired and the RBI principles should be traded in return for other desired qualities, such as: *Expressive power* (users can perform a variety of tasks within a given application domain), *Efficiency* (users can perform tasks rapidly), *Versatility* (users can perform many tasks from different application domains), *ergonomics* (users can perform a task without physical injury or fatigue), *accessibility* (users with a variety of abilities can perform a task), *practicality* (the system is practical to develop and produce) (Jacob et al., 2008). These considerations will be included in the next section, which describes some of the concepts of Imagination-based interaction

2.1.3 IMAGINATION-BASED INTERACTION

Many researchers describe that building VEs that try to approach the richness of 3D reality may often fail to address usability and ergonomics (Shneiderman, 2003), (Jacob et al., 2008), (Kulik, 2009). Though the most realistic VE interactions may be easy to comprehend, since they build on users' common knowledge, they may be unpractical in many senses. Jacob and colleagues described some of the desired qualities which should be traded for RBI (as described above), but they state that these qualities only should be traded explicitly and only in return for these qualities and they do not suggest a structured methodology for using these tradeoffs (Jacob et al., 2008). Alexander Kulik on the other hand suggests to explicitly exploit the use of what he coined *imagination-based interaction* (IBI) (Kulik, 2009). Most of the concepts and metaphors in WIMP based systems emerged from office work and most of them are inadequate for 3D applications. Alex-ander Kulik suggest to include other sources for comprehensible metaphors to take inspiration from popular science fiction stories, common knowledge about technological achievements, and the arbitrary combination of aspects from previously learned patterns.

"Unrealistic actions aren't necessarily impossible. As long as they're computable and users can imagine or vaguely understand the resulting functionality, they might be useful." (Kulik, 2009)

In his work he also suggests themes complementary to RBI, which are: *suspension of naïve physics*, *scaling of geometry and motion*, *automation*, *magic spells*, and *mode changes.*

Geometric scaling refers to the possibility of scaling computer graphics to provide more detail or improve accuracy. In multi-touch interaction this concept has become broadly accepted,

whereas in 3D UIs there is still no scaling concept with broad acceptance. In difference to 2D or mono scopic 3D UIs a full stereoscopic 3D display can offer the illusion of real scaling, whereas the others may only be perceived as a form of zooming (Kulik, 2009).

Motion scaling is related to scaling the motion input in order to tackle the tradeoff between accuracy and rapidity. This concept can take advantage of influencing the motion of pointers and other motion operators and can greatly improve usability. The tradeoff is that the real motion of a person cannot be influenced and therefore it may result in confusion. This concept is widely used in WIMP systems where the mouse pointer has increased acceleration under rapid motions (Control:Display ratio (C:D)).

Automation generally refers to the computers capability to automate processes. This can take various forms of semi to full automated interactions. In VEs examples can be found in navigation, where a distance is travelled automatic by e.g. selecting a target destination on an overview map or by selecting landmarks.

Magic Spells refers to computationally dynamically assigning meaning and automated processing to any involved object, which can be adapted to a variety of tasks and interaction styles (Kulik, 2009). In a VE this could for example involve selecting an object and assigning a color, without physically painting the object. Any arbitrary process could be applied and many interesting metaphors may be found in current computer games, which often utilize the concept of magic.

Mode changes refer to the properties of objects and tools in computer applications which might frequently change. This is especially important for avoiding confusion and a clear representation and distinguishable interface states are core properties. Emphasizing the transitions of system states can help support the user's cognitive processes (Kulik, 2009).

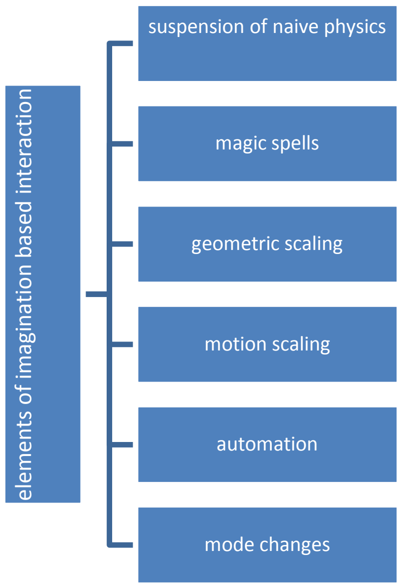

elements of imagination based interaction
suspension of naive physics
magic spells
geometric scaling
motion scaling
automation
mode changes

Figure 2 - Elements of the Imagination-based Interaction

2.1.4 DISCUSSION

The principles of RBI and IBI have been chosen as a starting point for finding out which principles and metaphors could describe interactions for VEs. It has to be noted that there is a large amount of literature about design guidelines and methodology, and we could include many other principles which has been proposed in the field of HCI. One thing we noted is that many of these principles or guidelines are either very specific for a given area, or they are to general for guiding 3D and VE interface design. Also we did not find any principles particularly suited for body controlled interaction, except the described ones. Since this form of interaction is fundamentally different from the form of human-computer interaction we are used to (mouse and keyboard), we are eager to find out if there could be a set of principles suited for this form of interaction. We recognize the RBI and IBI principles could be used for guiding VE interaction design, yet we miss a set of principles which unify the concepts of reality (RBI) and imagination (IBI). Also the proposed principles are theoretically and the authors did not prove their applicability, by e.g. basing a VE design on them. Such theoretical design principles are most likely derived by thinking about them and thus there exists a gap between theory and practice.

2.2 AFFORDANCES, CONSTRAINS, GOOD MAPPING AND FEEDBACK

While the principles described in the previous section may apply to a large range of present emerging interfaces there are a range of principles, which can be applied to almost any human computer interface as well as many everyday artifacts. We will in the following describe some of these principles that are of importance for VEs. We will also have a look on their origins from psychology, with a focus on the concepts of embodiment and cognition from an ecological point of view.

Donald Norman described many of the well-established concepts and principles used across most HCI disciplines in his work (Norman, 1990). The principles described by Norman are based on a discussion of everyday artifacts and a big part of these principles have their origin in the psychological concepts of James Gibson's ecological approach to visual perception (Gibson, 1979). Norman identifies four principles which are characteristics for usable artifacts: *affordances, constrains, good mapping* and *feedback*. He also includes some cognitive concepts in his work: *the gulfs of execution and evaluation*.

Affordances "…refers to the perceived and actual properties of the thing, primarily those fundamental properties that determine just how the thing could possibly be used." (Norman, 1990) Affordances provide strong clues to the operations of things, for instance a chair affords sitting, a knob affords turning, slots are for inserting things, etc. (Norman, 1990). *Constrains* are the limitations on the use of an object, with the intention to guide users into proper action. For example the holes in a pair of scissors limit the amount of possible fingers to constrain and guide the action (Norman, 1990). In 3D applications constrains are often used to limit the possible amount of manipulations, for instance translating objects in space is often constrained by one of the three major axis. *Mapping* refers to the relationship between two things, for example between the controls and their movement and the result in the world. This also refers to the conceptual model or metaphor on which the relation between action and outcome is based (Norman, 1990). The term *natural mapping* refers to taking advantage of physical analogies and cultural standards, which should lead to immediate understanding. *Feedback* is about sending information back to the user after performing an action, about what result has been accomplished (Norman, 1990). This concepts also relates to a psychology of causality (cause and effect), where something that happens right after an action appears to be caused by that action. Proper feedback is essential in all applications to not confuse the user and to actually communicate the success of actions. *The gulfs of execution and evaluation* refer to the gaps (or gulfs) that separate mental states from physical ones. Each of these reflects one aspect of the distance between mental representation of the person and the physical components and states of the environment (Norman, 1990). The gulf of execution refers to the difference between the intentions and the allowable actions. The gulf of evaluation reflects on the amount of effort that a user must exert to interpret the physical state of the system and to determine how well the expectations and intentions have been met (Norman, 1990). This gap is small when the system provides meaningful feedback, which should be easy to interpret and matches the users' mental model of the system. In relation to the previous mentioned RBI principles this gap is small as most of these principles are based on the users pre-existing knowledge from everyday life.

2.2.1 ECOLOGICAL PERCEPTION AND EMBODIMENT

Gibson's ecological approach to perception is in direct contrast to the traditional cognitive psychologist's viewpoint on perception. In the traditional view there is direct access only to sensations which integrated with memory provide higher level interpretations of actions or objects.

In Gibson’s view there is a directly perceivable complex group of features, which he denoted *invariants* (Gibson, 1979). In his viewpoint these invariants are directly perceivable and are not mediated by memory or inference. Though this approach is controversial his descriptions of embodiment is interesting to note for the field of body controlled interaction and VEs.

Affordances are the fundamental concept of the ecological approach to perception (Gibson, 1979). Here the definition is similar to Normans:

“Affordances are opportunities for action that objects, events or places in the environment provide for an animal.” (Hirose, 2002)

With this definition he also highlights that affordances are properties of the environment but not of the animal (e.g. human) itself. The means for acting that an animal can use to realize a specific affordance are called *affectivities*. These affectivities may change as the state of the animal changes, because they are properties of the animal (Hirose, 2002). *Tools* can extend or enhance these affectivities, for example a telescope amplifies the visual ability, or a hammer amplifies striking ability. Separated from the body these tools are functional extensions of the environment. In use, the tools are treated as functional extensions of the user; it extends the users affectivities to realize affordances of the environment (Hirose, 2002). By this tools may extend the users *body*; they shift the boundary between the body and the environment. For instance a blind persons stick extends the scope of touch, in use it is regarded as being part of the body, it extends the body. In sum the ecological approach to embodiment regards that the boundary of the body is changeable and can be extended by tools (Gibson, 1979) (Hirose, 2002). In relation to interactions in VEs this concept of extending the body by tools seems interesting; it may reveal different forms of ITs when used as a perspective for investigation. The different ITs used in VEs also draw a large part of their inspiration from this ecological view, where for example users can extend their arms to select and manipulate distant objects. Furthermore a lot of potential is offered by this ecological view, considering VEs where virtual human body avatars can be extended, parts replaced or even bodies completely replaced by any possible mean, giving possibilities for a broader application of ITs.

2.3 PERCEPTION AND HUMAN COGNITION

Working with multimodal systems like VE’s, further understanding of the processes human undergo while interacting in such systems is needed. Nadine Sarter (Sarter, 2006) argues that most research within human-computer interaction has treated system input and output as different domains. Similar division into two directions is found in the literature concerning the human processes: perceptual and cognitive psychology. Perceptual psychology (the input) studies how human perceive their surroundings through their senses. Cognitive psychology (reasoning and output) focuses on the mental processes, how people learn, reason etc. According to (Buxton et al., 2011) these two processes are interlinked, and we cannot discuss one process without the other.

Within the research area of human perception, there is a common understanding that perception is a multimodal process (Pai, 2005). According to Pai (2005) the most known multisensory phenomenon is the ventriloquism effect, the spatial location of the sound is where the location of correlating visual stimuli is situated. The McGurk effect (McGurk & MacDonald, 1976) is a known illustration of speech being a multimodal process. In haptics, surface and spatial information gathering is seen as a multimodal process combining touch, audio and vision (Klatzky & Lederman, 2003).

Most Virtual Environment applications have focused on visual perception, the reason is according to Bowman and colleagues that most immersive VEs are highly visual (Bowman et al., 2005). According to them an important aspect in VEs is the perception of depth in the environment, where different visual cues can help the subject to determine the depth of the scene. These cues are divided into monocular or binocular, static or dynamic.

When dealing with interaction and interfaces, words like natural interaction and intuitive designs are often mentioned. Oxford dictionary defines intuitive as: "*Of knowledge or mental perception: That consists in immediate apprehension, without the intervention of any reasoning process."* meaning the cognitive process that is needed to be capable of performing a task without thinking about it.

Many cognitive theorists have tried to map the cognitive process, where one of the best known is the "*Modal Model of Memory*" suggested by Atkinson and Shiffrin in 1968. According to them there are three types of memory storage: *sensory, short term memory and long term memory.*

The model suggests that all memory pass through short term to a longterm store after a period of time. Miller's law suggests that the short time memory can store seven plus minus two units at the time (Miller, 1955). Long term memory on the other hand can store a vast amount of information in a long period of time (Ericsson & Kintsch, 1995).

Although the "*Modal Model of Memory*" theory has been criticized by later theorists for being a simplification of the mental processes, e.g. (Eysenck & Keane, 2005) and (Baddeley, 1997), it has been the basis for many cognitive theories.

According to Eysenck and Keane the key component of the working memory is the central executive and it has limited capacity, the other three are subsystems that are relative independent of each other. The assumption of working memory is:

- *If two tasks use the same component, they cannot be performed successfully together.*
- *If two tasks use different components, it should be possible to perform them as well together as separately*

(Eysenck & Keane, 2005)

Knowing these limitations of the working memory will aid to a better utilization of multimodal sensory elements in an interactive VE design.

2.4 INTERACTION TECHNIQUES

The following chapter will introduce novel research within the area of ITs in VEs. The research is presented in the order using Bowman & Colleagues definition of interaction areas (Bowman et al., 2005). This definition consists of three main areas, which are defined as:

- *Navigation:* travel or view point motion control
- *Selection*: involves the specification of one or more virtual objects by the user for some purpose
- *Manipulation:* setting the position and or orientation of a virtual object

(Bowman, 1999).

Another element of VEs are User Interfaces (UI), such as menus, toolboxes, palettes etc. These are defined by Bowman as *System Control techniques*. Chen & Bowman refers to theses generic task levels as universal, meaning that they can be applied to any application without taking the context into account (Chen & Bowman, 2009).

Having these definitions in mind the next step is to examine what has previously been done in the area of ITs for VEs. The main focus will be to examine the ITs, how they were supported by a medium, and finally examine to which degree they are usable to the current investigation.

To ease the discussion of past research, an outline of research done in each of the task areas navigation, selection, manipulation and system operations, will be illustrated in tables. The information about each project we see relevant at this point is the input/output interface and the ITs used in order to perform each of the tasks.

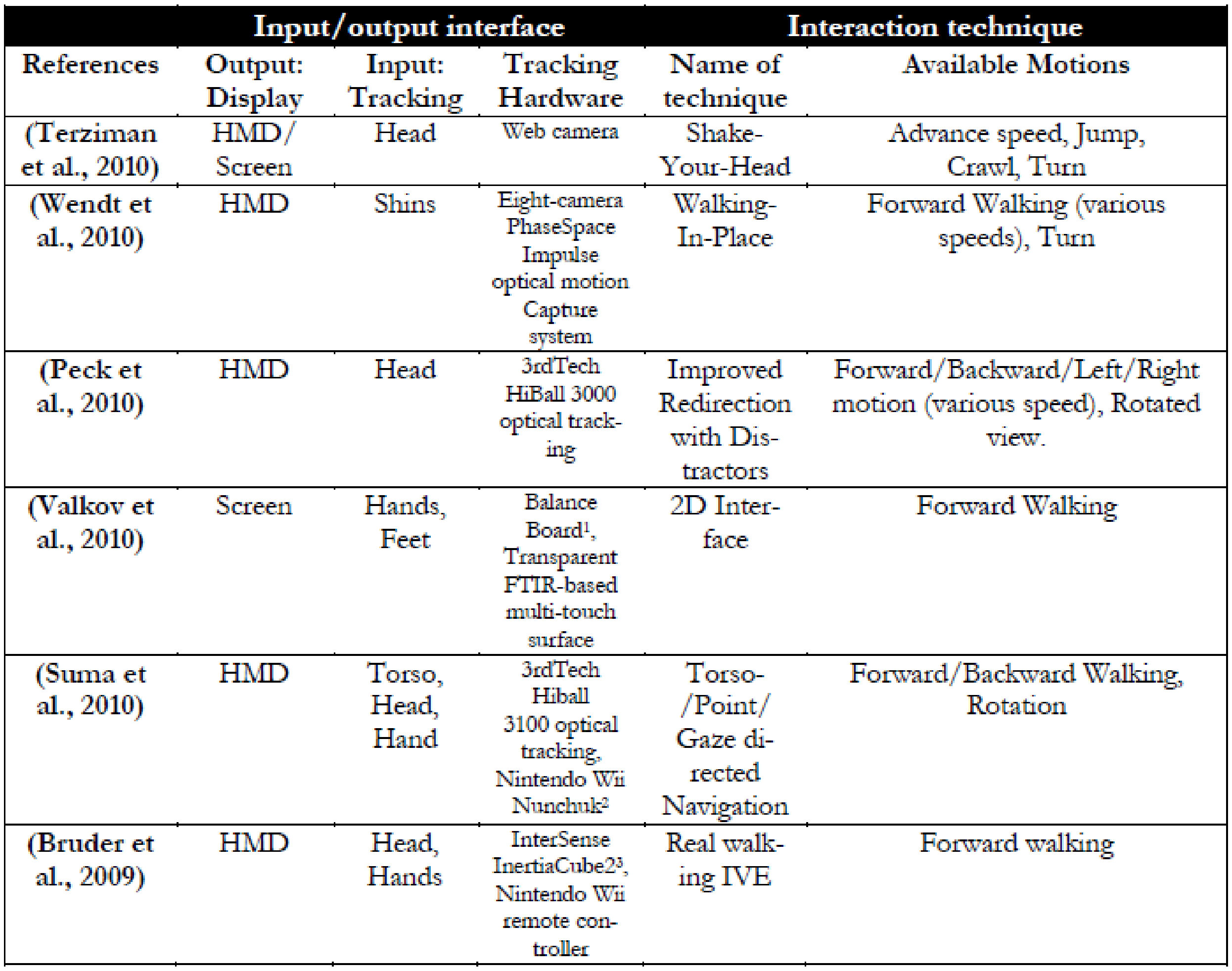
2.4.1 NAVIGATION

The articles mentioned in the following section and in Table 1 are a cross section of the most reoccurring ITs concerning navigating in VEs, which according to Suma et al. are: "*real walking, walking-in-place, gaze-directed, pointing-directed, torso-directed, and traditional desktop controls*" (Suma et al., 2010). Some projects use a single technique, while others choose to use a combination of different techniques.

The problem with navigation in VEs is the fact that one is confined to a real world physical space while travelling a possible endless virtual space. There have been many attempts to investigate this problem. Researchers have been trying to find out if one can map the virtual space to the limited physical space for navigation (Peck et al., 2010), others have constrained the physical boundaries, may it be the space or movement of the subject (Valkov et al., 2010), (Terziman et al., 2010), (Wendt et al., 2010), while others to a certain degree have implemented real life navigation from a physical space to a virtual one (Bruder et al., 2009), (Suma et al., 2010).

The question of which technique is best suited has been asked and to a certain degree answered. According to Suma et al. there is no significant difference in travel performance between real walking, gaze directed or torso directed navigation. What they were focusing on was the degree of information the user gained while navigating in VEs, but on the other hand real walking out performed pointing direction techniques. This opens the discussion about to what degree the pointing direction technique is suited for navigation. Suma & colleagues and others have proven that one does not need real life navigation in order to gain information about the VE. Bowman pointed out that the user needs to have a fluent stream of information, in order not to be disoriented in the Virtual Environment (Bowman et al., 1997). He describes that actions like teleportation from one place to another will be more confusing for the user than having him travelling from one location to another via traditional travelling means, meaning linear travel like walking.

Table 1 - An overview of research done in the area of navigating in the VE. The figure is inspired by (Terziman et al., 2010).



	Input/output interface			**Interaction technique**	
References	**Output: Display**	**Input: Tracking**	**Tracking Hardware**	**Name of technique**	**Available Motions**
(Terziman et al., 2010)	HMD/ Screen	Head	Web camera	Shake-Your-Head	Advance speed, Jump, Crawl, Turn
(Wendt et al., 2010)	HMD	Shins	Eight-camera PhaseSpace Impulse optical motion Capture system	Walking-In-Place	Forward Walking (various speeds), Turn
(Peck et al., 2010)	HMD	Head	3rdTech HiBall 3000 optical tracking	Improved Redirection with Distractors	Forward/Backward/Left/Right motion (various speed), Rotated view.
(Valkov et al., 2010)	Screen	Hands, Feet	Balance Board[1], Transparent FTIR-based multi-touch surface	2D Interface	Forward Walking
(Suma et al., 2010)	HMD	Torso, Head, Hand	3rdTech Hiball 3100 optical tracking, Nintendo Wii Nunchuk[2]	Torso-/Point/ Gaze directed Navigation	Forward/Backward Walking, Rotation
(Bruder et al., 2009)	HMD	Head, Hands	InterSense InertiaCube2[3], Nintendo Wii remote controller	Real walking IVE	Forward walking

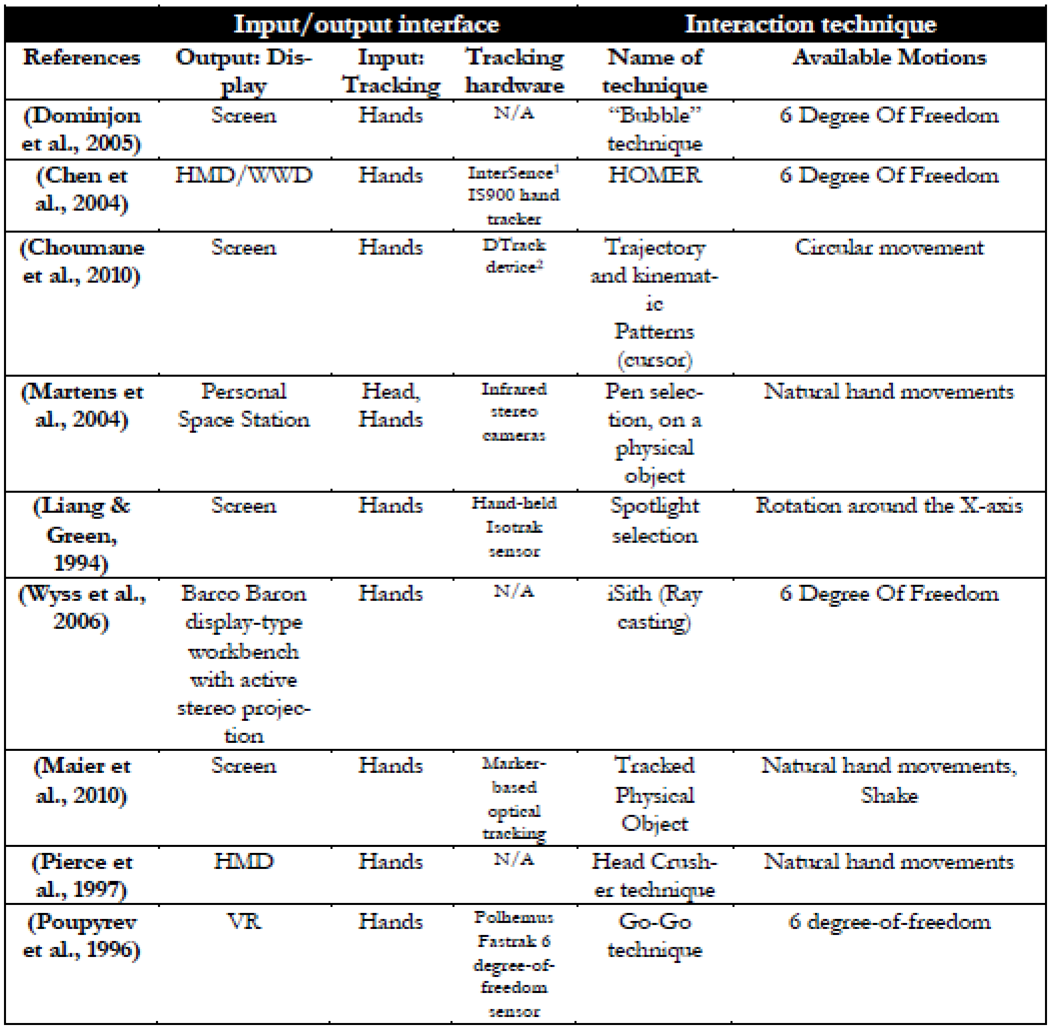
There is a fair amount of research done in the area of navigating in VEs, and the outcome of this research seem to mount out in the fact that in order to design a navigation method that will give the user most information about the virtual environment that he/she is travelling in, the best approach would be to give the user a consistent fluent stream of information as he/she would have when travelling in a physical environment.

The research shows further that one is not necessary bound to the physical real life exploration options, when travelling in VEs. When claiming that the question of best suited navigation techniques is to a certain degree answered, is due to the fact that it depends on what one is aiming for: is it an attempt to find the most user friendly way of travelling? Or is it to find a way of navigating which affects the presence or engagement? These choices open up for another discussion which is not within the scope of this project.

2.4.2 SELECTION

Mine defines two main categories of selection techniques as: *local* and *at-a-distance* (Mine, 1995). Local, as the name suggests, concerns the process of selecting objects that are within the subject's reach. All of the mentioned articles in Table 2 have an element of local selection involved in their research, but (Martens et al., 2004), (Maier et al., 2010) focus both on selecting objects within reach

Table 2 - An overview of research done in the area of close and out of range selection techniques in the VE (source Wikipedia).



	Input/output interface			Interaction technique	
References	**Output: Display**	**Input: Tracking**	**Tracking hardware**	**Name of technique**	**Available Motions**
(Dominjon et al., 2005)	Screen	Hands	N/A	"Bubble" technique	6 Degree Of Freedom
(Chen et al., 2004)	HMD/WWD	Hands	InterSence[1] IS900 hand tracker	HOMER	6 Degree Of Freedom
(Choumane et al., 2010)	Screen	Hands	DTrack device[2]	Trajectory and kinematic Patterns (cursor)	Circular movement
(Martens et al., 2004)	Personal Space Station	Head, Hands	Infrared stereo cameras	Pen selection, on a physical object	Natural hand movements
(Liang & Green, 1994)	Screen	Hands	Hand-held Isotrak sensor	Spotlight selection	Rotation around the X-axis
(Wyss et al., 2006)	Barco Baron display-type workbench with active stereo projection	Hands	N/A	iSith (Ray casting)	6 Degree Of Freedom
(Maier et al., 2010)	Screen	Hands	Marker-based optical tracking	Tracked Physical Object	Natural hand movements, Shake
(Pierce et al., 1997)	HMD	Hands	N/A	Head Crusher technique	Natural hand movements
(Poupyrev et al., 1996)	VR	Hands	Polhemus Fastrak 6 degree-of-freedom sensor	Go-Go technique	6 degree-of-freedom

and on having a physical object as aid to the selection. Martens et al. claim that having physical objects as aid will support the natural interaction with the 3D world. These techniques have the limitation of the physical object, although one can argue that this object can serve as a container of virtual possibilities, such as a 2D widget interface in a VE. According to Wingrave, Bowman, & Ramakrishnan (2002) there are in the area of selection techniques three main metaphoric naming approaches: *ray casting* (Dominjon et al., 2005) (Wyss et al., 2006) (Liang & Green, 1994) (Chen et al., 2004), *occlusion* (Pierce et al., 1997) and *arm extension* (Poupyrev et al., 1996)**.** Linking these metaphors to the category of selection *at-a-distance*, it becomes a question of how well the user understands the metaphor of selecting objects that are far away and how well the feedback is designed. One can argue that having a ray as a pointing support to the process of selection can be helpful, while others will argue having a "bullseye" occlusion selection technique (Wingrave et al., 2002), helps the user to set the object of interest in focus. With arm extension selection techniques the user can to a higher extent relate to the action of selection, since it is to some point a more human like action.

Most of the research done in the area of object selection has focused on the visual feedback the user gets when performing an action and has focused on the hands as the primary source of selection. Another observation made is that most feedback used in selection techniques is visual feedback. This seems to be a great limitation when having the possibility of a multimodal VE. It seems that sound has not been prioritized when dealing with selection techniques which could be subject for future research.

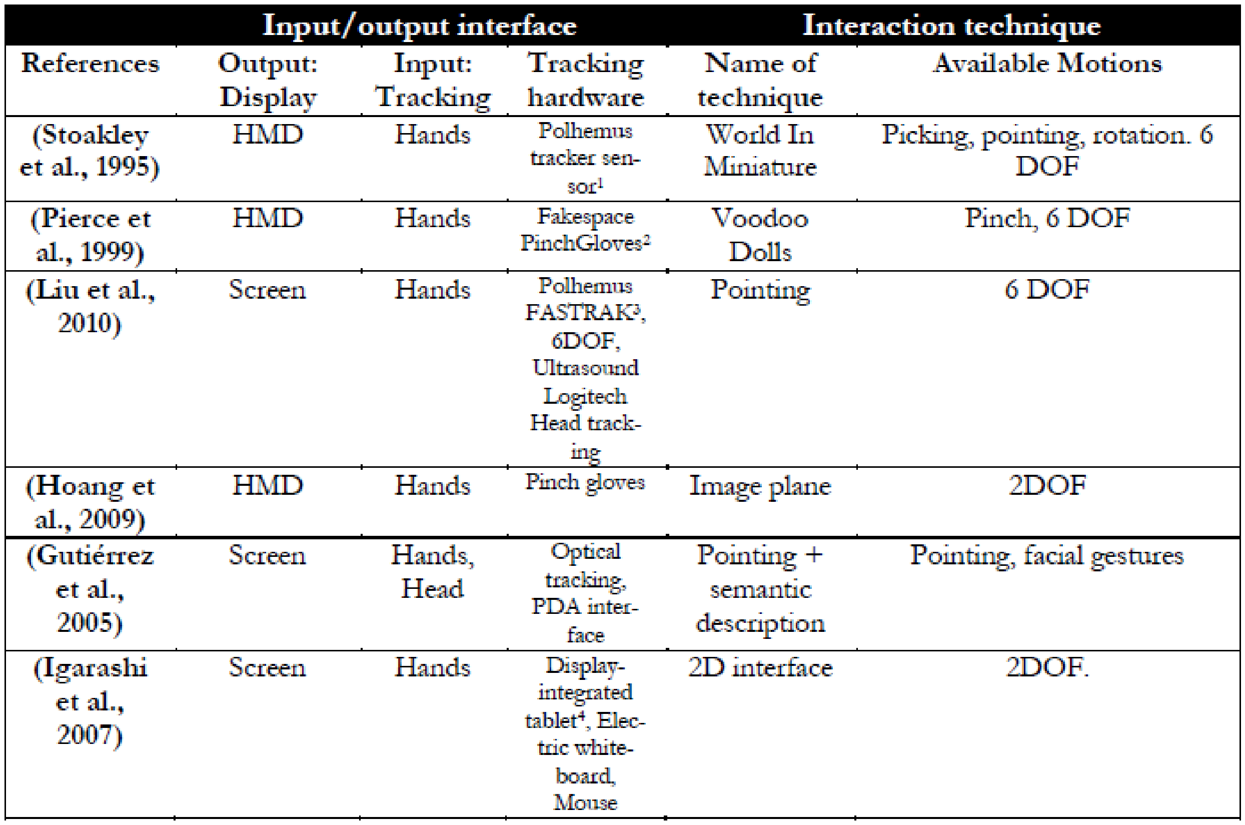
2.4.3 MANIPULATION

Bowman et al. (2005) defines manipulation as a set of tasks, where to a degree each manipulation contains a combination of the three basic tasks *Selection, Positioning and Rotation* (Bowman et al., 2005). Further Bowman et al. (2005) separates the manipulation ITs being either *isomorphic* or *non-isomorphic* view.

Isomorphic view suggests that the physical world and virtual world's actions are geometrically mapped one-to-one. A non-isometric view suggests a more open frame of mapping physical actions to virtual actions. Further Bowman classifies the techniques as series of metaphors. These classifications will assist in organizing the techniques presented in Table 3.

We can derive from Bowman’s definition of manipulation that selection techniques to a certain degree also can be considered as manipulation techniques. Some of the in the selection techniques section mentioned researchers operate within the isomorphic point of view, while (Poupyrev et al., 1996), (Chen et al., 2004) and (Pierce et al., 1997) have a more non isomorphic approach towards manipulation. The selection part of the manipulation often has a isomorphic approach, other manipulation function often use a non-isomorphic method, as e.g. a *plane* based manipulation (Hoang et al., 2009), *Voodoo doll technique* (Pierce et al., 1999), *2D interface technique* (Igarashi et al., 2007), or *WIM* (Stoakley et al., 1995).

Table 3 - An outline of manipulation techniques. (source Wikipedia)



	Input/output interface			Interaction technique	
References	**Output: Display**	**Input: Tracking**	**Tracking hardware**	**Name of technique**	**Available Motions**
(Stoakley et al., 1995)	HMD	Hands	Polhemus tracker sensor[1]	World In Miniature	Picking, pointing, rotation. 6 DOF
(Pierce et al., 1999)	HMD	Hands	Fakespace PinchGloves[2]	Voodoo Dolls	Pinch, 6 DOF
(Liu et al., 2010)	Screen	Hands	Polhemus FASTRAK[3], 6DOF, Ultrasound Logitech Head tracking	Pointing	6 DOF
(Hoang et al., 2009)	HMD	Hands	Pinch gloves	Image plane	2DOF
(Gutiérrez et al., 2005)	Screen	Hands, Head	Optical tracking, PDA interface	Pointing + semantic description	Pointing, facial gestures
(Igarashi et al., 2007)	Screen	Hands	Display-integrated tablet[4], Electric whiteboard, Mouse	2D interface	2DOF.

There is a vast amount of literature concerning manipulation techniques that utilize hand tracking in some manner, while whole body movements are to a lesser degree employed in the design of manipulation techniques. This focus of interest has the effect that most metaphoric representations of manipulations are limited to tasks that can be subjected to hand labor.

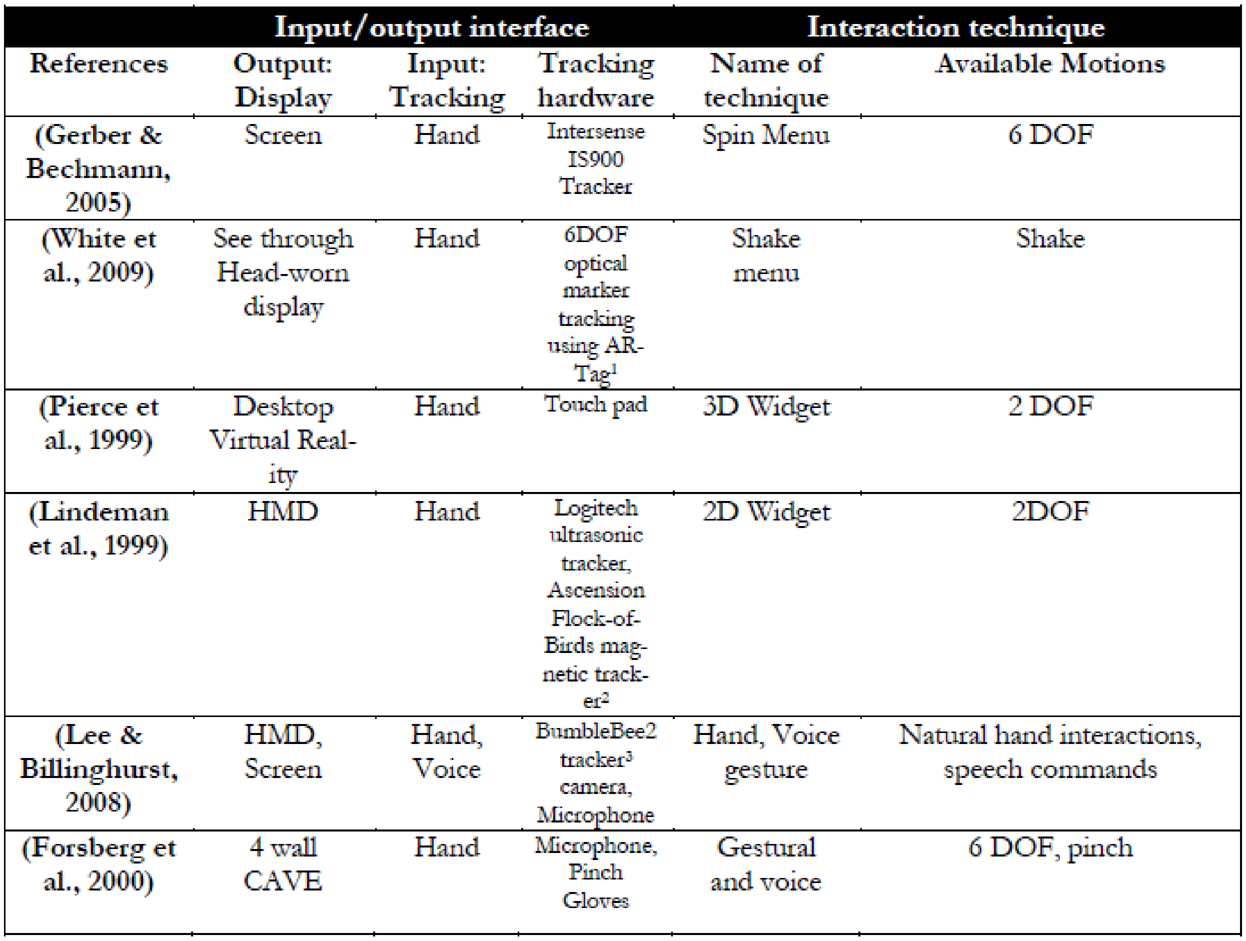
2.4.4 SYSTEM CONTROL

Bowman et al. (2005) composed a taxonomy that sums up the different areas of system control, *Graphical menu* (Lindeman et al., 1999) (Pierce et al., 1999), *Voice command* (Lee & Billinghurst, 2008), *Gesture command* (White et al., 2009) and *Tools* (Forsberg et al., 2000).

Pierce et al. (1999) identified drawbacks of using 2D or 3D widgets in a VE, 2D consuming screen space, and a constant shift between 2D desktop and 3D world. 3D widgets have the issue of access, how to gain access to them, and the danger of cluttering up the 3D world. Voice commands have been examined, by e.g. (McTear, 2001). Lee & Billinghurst (2008) concluded in their research, that in order to use speech recognition key words are needed.

Gesture commands are often limited to the input technology used in the setup. Bowman et al. (2005) define three types of gesture input techniques; *glove-based recognition, camera-based recognition, surface-based recognition*. White & colleagues (2009) use a camera-based gesture recognition technique, which is based on a shake technique. They argue that this has an advantage over simple button press, by the fact that any object can be tracked and function as a menu controller. Tools in VEs are defined by Bowman & colleagues (2005) as either physical or virtual. They argue that the use of familiar real-world devices will increase the 3D interaction usability. Forsberg & colleagues (2000) used a virtual tool belt as aid for interacting in the VE; the belt contained different 3D metaphorical widgets. Design of graphical menus seems to be largely inspired by earlier desktop application designs, based on the WIMP metaphor. This is not directly a negative direction to take, most of the methods are tested and verified to a full extend, but we argue that 3D environments must and can offer more than just existing 2D interaction methods.

Table 4 - An overview of different techniques within the area of system control.(Forsberg & colleagues (2000))



	Input/output interface			Interaction technique	
References	Output: Display	Input: Tracking	Tracking hardware	Name of technique	Available Motions
(Gerber & Bechmann, 2005)	Screen	Hand	Intersense IS900 Tracker	Spin Menu	6 DOF
(White et al., 2009)	See through Head-worn display	Hand	6DOF optical marker tracking using AR-Tag[1]	Shake menu	Shake
(Pierce et al., 1999)	Desktop Virtual Real-ity	Hand	Touch pad	3D Widget	2 DOF
(Lindeman et al., 1999)	HMD	Hand	Logitech ultrasonic tracker, Ascension Flock-of-Birds mag-netic track-er[2]	2D Widget	2DOF
(Lee & Billinghurst, 2008)	HMD, Screen	Hand, Voice	BumbleBee2 tracker[3] camera, Microphone	Hand, Voice gesture	Natural hand interactions, speech commands
(Forsberg et al., 2000)	4 wall CAVE	Hand	Microphone, Pinch Gloves	Gestural and voice	6 DOF, pinch

2.5 VE INPUT AND OUTPUT METHODS

Previous research of ITs has to a large degree focused on obtaining user input data by devices, such as e.g. pinch gloves and tracking devices which often are very expensive and exclusively used in lab settings. Our focus will be on the recent trend of affordable depth sensing cameras, starting with the possibilities these products can offer. The following section will cover the area of different frameworks that are used in user tracking. Finally the output devices will be covered, here the focus will be on 3D output source, since the goal of this study is to create IT for 3D Virtual Environments.

2.5.1 INPUT

Since our motivation is to create a VE that allows users to perform actions without wires attached to the user, the following section will cover optical tracking devices allowing such interaction. Optical tracking utilizes the electromagnetic spectrum of light, as depicted in Figure 3.

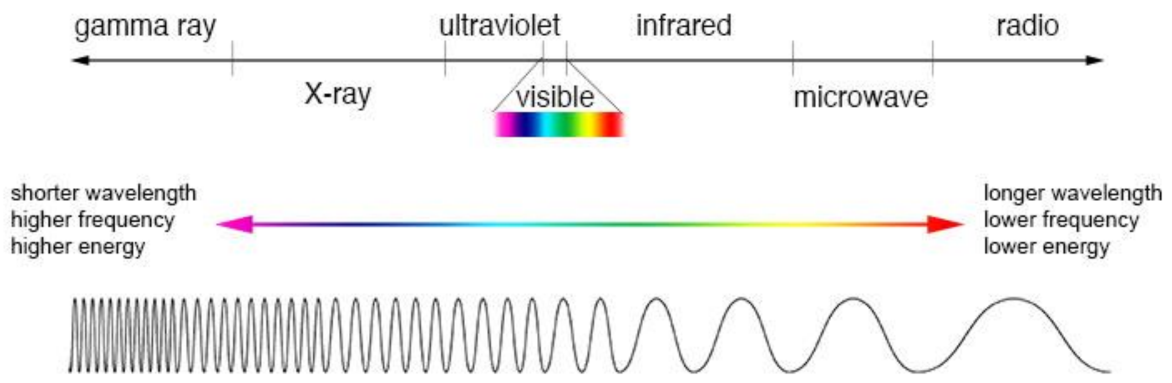


Figure 3 - The electromagnetic spectrum ranging from about 1012 Hz to more than 1022 Hz.

Source of figure (Foxlin, 2002)

According to Foxlin (2002) common and inexpensive cameras based on CCD and CMOS driven sensors can be used to record visible and near Infrared (IR) light. Since most optical tracking systems up to date work in that area, due to complications toward reaching in the mid-IR and UV light (Foxlin, 2002), this may be considered as a relative cheap method of tracking by e.g. using multiple web cameras.

There are different approaches towards optical tracking, these are beacontracking, optical TOF ranging and structured light. PlayStation Move uses the beacon tracking, whereas the Kinect utilizes a structured light methodology (PrimeSense Ltd., 2010).

Beacon tracking is where the subject wears an active or passive vision based tracker, and the system recognizes the position and orientation of the subject. There are two systems of beacon-tracking; outside-in and inside-out. Outside-in is better for positioning of the subject, while in-side-out is better to determine the orientation of the subject. The main disadvantage of the optical system is occlusion of the tracked objects; that is when the line of sight is not clear between the sensor and the object that is being tracked.

The structured light approach, taken with i.a. the Kinect, has been to send out IR light in the scene and using a CMOS sensor to read the coded light back from the scene (PrimeSense Ltd., 2010). The result is a 3D representation of the scene, which then can be further processed by computer vision software.

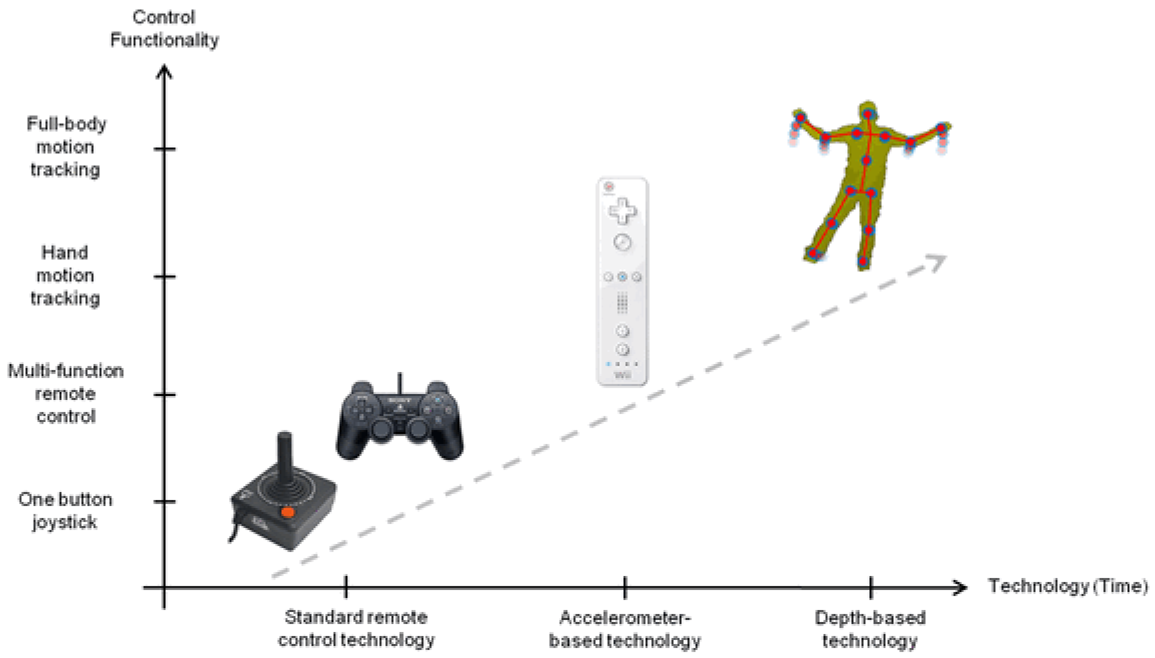

Control Functionality
Full-body motion tracking
Hand motion tracking
Multi-function remote control
One button joystick
Standard remote control technology
Accelerometer-based technology
Depth-based technology
Technology (Time)

Figure 4 – How PrimeSense capturing all the existing controlling mechanism.(Source PrimeSense.com)

The advantage of the structured light approach over the other optical tracking devices is that it does not require any form of physical addition on the subject (as e.g. a device in the hand or markers on the body).

For further details on tracking devices see (Bowman et al., 2005), (Welch & Foxlin, 2002) and (Foxlin, 2002).

2.5.2 BODY PART RECOGNITION

The choice of using the Kinect as an input source has the advantage of having access to the publicly available NITE tracking framework ((OpenNI, 2011) (PrimeSense, 2011)) which is capable of retrieving the depth information and from that recognizing body parts. Since the focus of our investigation is on the creation of ITs, the access to such a tracking framework was considered as an advantage that could not be overlooked.

According to Shottom & colleagues it is possible from a single input depth image to have a perpixel body part distribution, which is color labeled in order to conduct weight estimation resulting in a proposition of each skeletons joint position, as seen in Figure 5. This information can be used in order to construct different interaction methods in a VE. See (Shotton et al., 2011) for more details concerning this tracking methodology.

This methodology will give the position of each major body part, but according to (OpenNI, 2010) individual finger information cannot be obtained; only the center of the palm. Hence the interaction techniques will be confined to body joints.

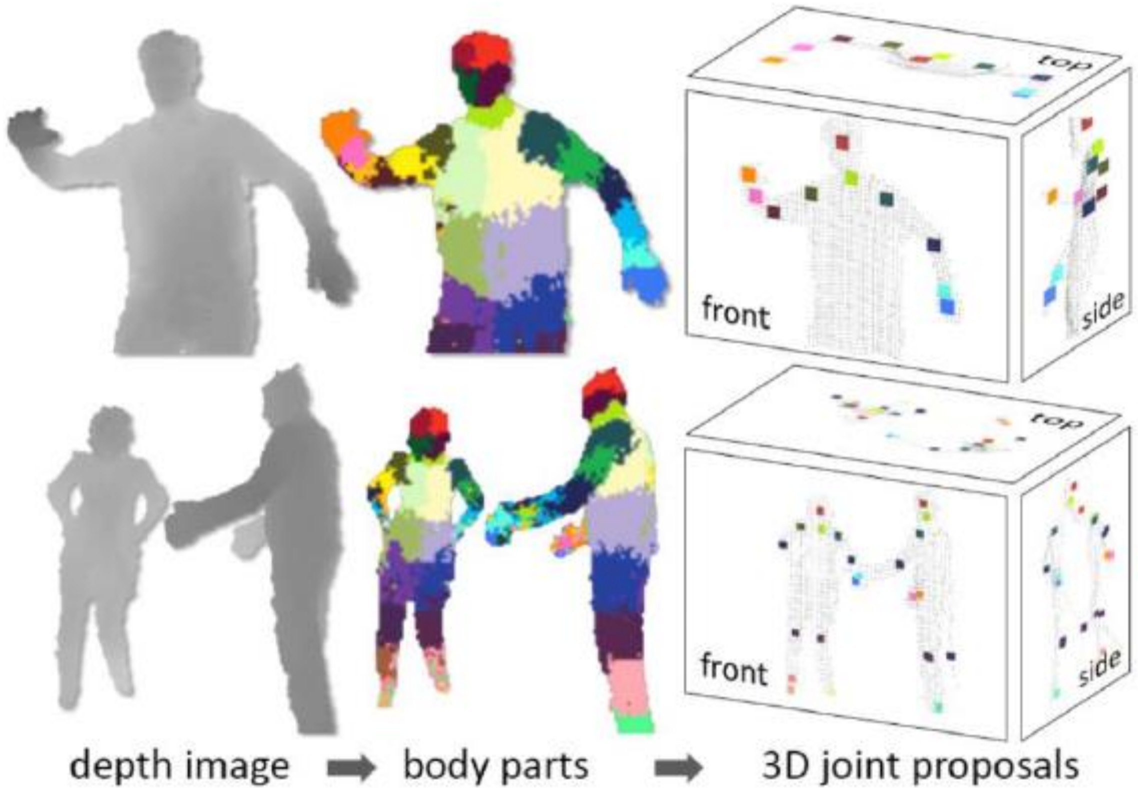


Figure 5 - Method of retrieving camera depth information and predicting 3d positions of body joints. Source of image: (Shotton et al., 2011)

2.5.3 OUTPUT

According to Foxlin (2002) in the field of 3D visual displays, there are two types of displays; Head Mounted Displays (HMD) and Fixed Surface Displays (FSD). Due to the motivation of having a VE that has no wires connected to the user, the choice of output was to use FSD.

Fixed Surface Displays denote any stationary display station ranging from a desktop display to ImmersaDesk™, to a wall projection that can be up to a full Spatially Immersive Display (SID) e.g. CAVE™. These FSD can be equipped for stereoscopic 3D view, using shutter glasses, polarized glasses or auto stereoscopic display techniques (Foxlin, 2002).

The choice of output display for a VE fell on using a wall projection with stereoscopic capabilities accompanied with shutter glasses. The reason was due to accessibility of materials.

2.6 DISCUSSION

For the development of ITs *two* different approaches can be used as the bases for design, where most ITs use *metaphors or principles* to form the fundamental mental model of the technique, as for example the metaphors presented in this pre-analysis. The other approach is *systematic task decomposition*, where tasks are classified and divided into components, which serve as basic elements for constructing and combining new ITs, as used in the work of (Bowman et al., 2005).

Previous research in ITs has been focusing on the hands as a primary mean of *selecting* or *manipulating* objects in the virtual world, *system control* differs to a degree since there have also been implemented voice commands. In the area of *navigation*, researchers have implemented methods that involve more body parts, such as torso, feet and head.

In many research prototypes data gloves or 3D pointer devices are used for selection and manipulation tasks. Although many other examples can be found, there has only been little attention on using continuous input devices for all tasks. Most often a combination of devices is used for input, e.g. tracking devices together with data gloves or 3D pointer devices. In most cases prototypes use buttons, bend sensors or other form of finger controlled sensors to execute actions, such as confirming a selection. Only little attention has been paid on solely using tracking as input. We think that in most cases the combination of input devices was used to avoid dealing with the problems that come when relying on a single input device. We do recognize that using multiple input devices may give advantages for many tasks, but we believe that at least an equal usability can be achieved with a single input device, such as the Kinect. In this context it is interesting to note that one of the guidelines from the literature for choosing input devices is the following:

"It is often better to have a series of specialized devices that work well for a specific group of interaction techniques rather than one or two generic input devices for all of the techniques in 3D applications." (Bowman et al., 2005) - p. 128

It is our intention to investigate the problems that come with using a single continuous input device and to find out if the Kinect together with the NITE framework qualifies as useable interaction device for VEs. One of the important problems to deal with in this case is how to *confirm or execute* *actions or selections* (coined "*confirmation of selection*" by (Bowman et al., 2005)), since the input is continuous and there is no distinct state indicating a confirmation.

2.7 METHODOLOGY

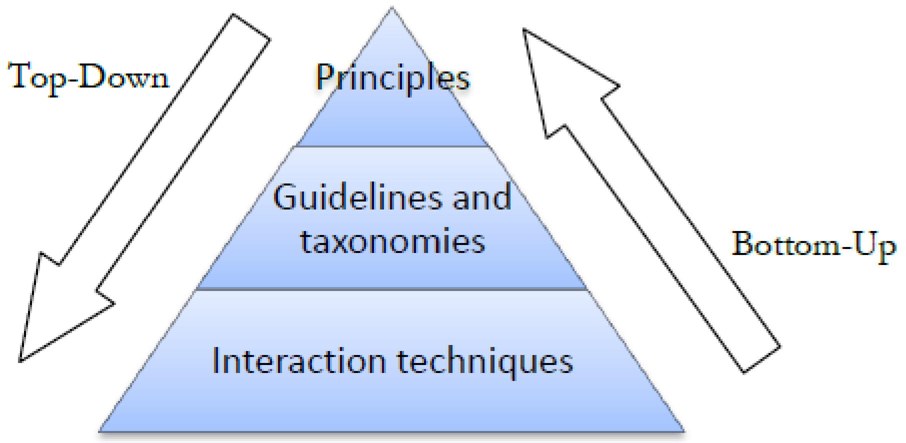


Figure 6 - Top-down vs bottom-up approach

2.7.1 PROCESS

For the design and development of new ITs we will consider two principal different approaches, which we refer to as *top-down* or *bottom-up* approaches (Figure 6). In general the top-down approach starts with a broad perspective and from that perspective the more concrete design concepts are developed. In practice this could be the formulation of design principles or metaphors from which ideas for ITs are developed. Often such approach is helpful for creating consistency in the design of the interface and the associated ITs. The disadvantage of such an approach is often the lack of formalism and structure (Bowman & Hodges, 1999), when compared to a bot-tom-up approach. The bottom-up approach tries to give structure and formalize in a systematic way all parts of a system. In this approach taxonomies are often used to create categories and to identify shortcomings in a design. Bowman and colleagues use such an approach throughout his work (Bowman, 1998) (Bowman & Hodges, 1999) (Bowman, 1999) etc., to formalize ITs for VEs.

Using a similar approach building on the existing research and findings will be used as a starting point in the following analysis chapter. This will help us to identify various possibilities of ITs for the selection tasks. Since we also want to incorporate consistent metaphors in the design of selection techniques we will synthesize principles, based on the research presented in this pre analysis.

These principles will be used in the practical design of new ITs for selection tasks.

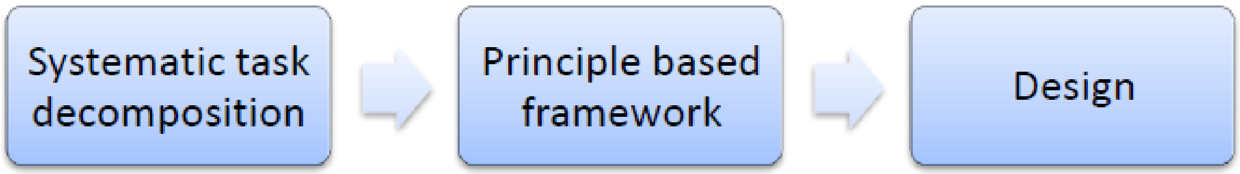


Figure 7 - Process structure analysis

Figure 7 shows an outline of the theoretical process for the analysis, which will be used as the bases for the design. The design process has the goal of designing ITs for selection, suited for various scenarios. Though the basis for this process is a theoretical developed framework, it is still a fundamentally creative process; ideas need to be developed, prototyped and tested. Our approach to this process will be based on Human centered design (Maguire, 2001), depicted in Figure 8. The overall process is based on an *active involvement of users, appropriate allocation of function between user and system* and *iteration of design solutions* (Maguire, 2001). The active involvement of users will in general let us evaluate the usability of the system, where an early involvement can identify weaknesses early in the process and thus lead to avoiding errors. The appropriate allocation of function between users and system is especially important, as this is used to determine which aspects of the task should be handled by the user and which by the system (Maguire, 2001). The iteration as overall structure is essential for the process; it allows constant reevaluation and refinement of proposed design solutions. In general the design starts with developing a low fidelity prototype, using paper drawings and photos in order to allow an early identification of problems and solutions. Later stages of the design include high fidelity prototypes, with a proper implementation of a system focused on selection tasks in VEs.

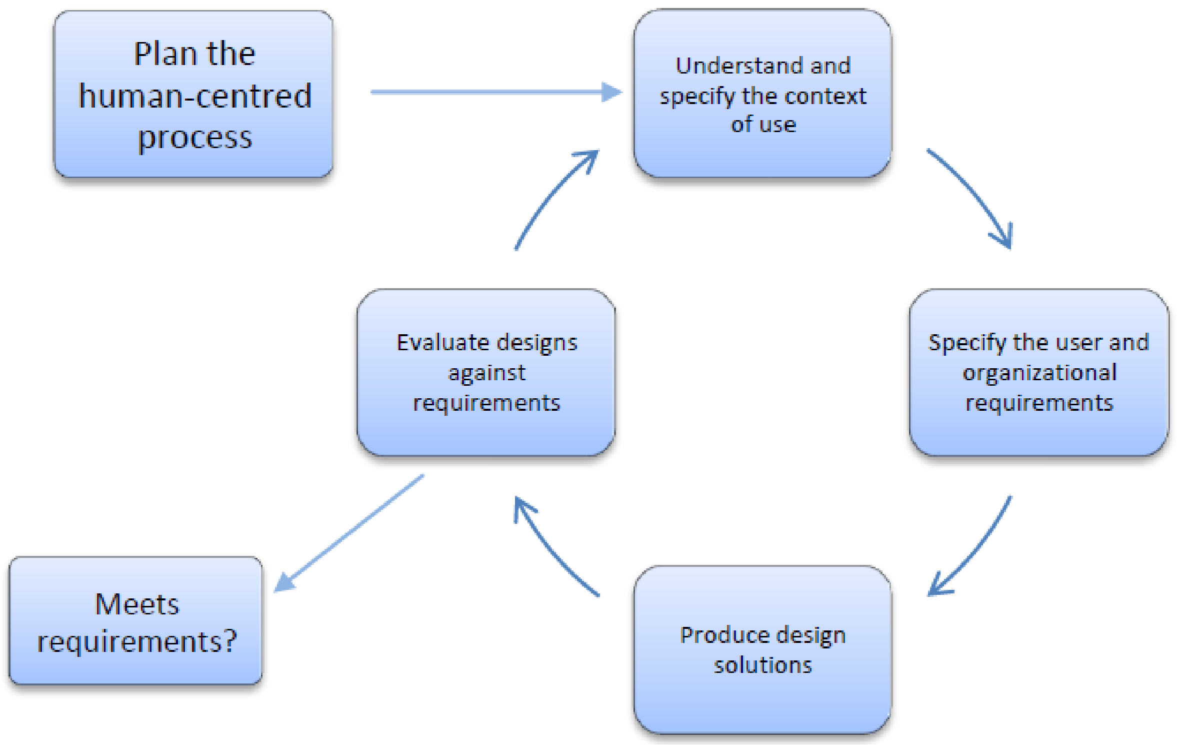

Plan the human-centred process
Understand and specify the context of use
Specify the user and organizational requirements
Produce design solutions
Evaluate designs against requirements
Meets requirements?

Figure 8 - Key human-centered design activities (from the ISO 13407 standard)

2.7.2 EVALUATION CRITERIA AND METRICS

With the goal of developing and verifying usable selection techniques, we need to define usability. Jakob Nielsen defined five usability attributes in his book "Usability Engineering" (Nielsen, 1993), which are:

- *Learnability*: The system should be easy to learn so that the user can rapidly start getting some work done with the system.
- *Efficiency*: The system should be efficient to use, so that once the user has learned the system, a high level of productivity is possible.
- *Memorability*: The system should be easy to remember, so that the casual user is able to return to the system after some period of not having used it, without having to learn everything all over again.

- *Errors*: The system should have a low error rate, so that users make few errors during the use of the system, and so that if they do make errors they can easily recover from them. Further, catastrophic errors must not occur.
- *Satisfaction*: The system should be pleasant to use, so that users are subjectively satisfied when using it; they like it.

Other authors choose a more compact definition into three major categories: *effectiveness, efficiency,* and *satisfaction* (Frøkjær et al., 2000). Here the effectiveness refers to the accuracy and completeness with which a user reaches a certain goal. Metrics for effectiveness can include qualitative measures meaning specific measures of the interaction between user and system and error rates. Efficiency refers to the relation between the accuracy and completeness with which users reaches certain goals as well as the resources expended in achieving them. Metrics for efficiency include task completion times and learning time. Satisfaction refers to the comfort and subjective attitude of users towards the system. Metrics can include different rating scales of the user comfort or attitude, commonly evaluated through interviews or questionnaires. In VEs the satisfaction could also include measures of presence, for which a range of questionnaires have been developed.

Besides metrics for evaluating usability which are focused on the user, the system performance should also be evaluated. The system performance gives indications on how usable a system is, since a system achieving low performance may hinder the user interaction significantly. For system performance general metrics include, average frame rates, average latency meaning input and output delay, variability in frame rate identifying situations with low frame rates, network delay, and distortion which in VEs for example can happen in graphical output using HMDs. All of these parameters can have an effect on task performance and usability.

2.7.3 EVALUATION METHODS

With the iterative human centered design used in this thesis practical work, a range of different methods for evaluating the different stages of prototypes will be used. Early prototypes are thus based on heuristic evaluations, whereas the later stages include formative and summative evaluations. In the following we will give a short introduction of these three evaluation forms, where more details can be read in the respective test chapter and the appendices.

Heuristic evaluations1 are used on the early low fidelity prototypes. These refer to an evaluation by interface experts, based on a set of guidelines. The interface and ITs can be examined visually, via written descriptions, or through actual use, in order to determine if the set criteria are met (LaViola et al., 2009). In our work we used heuristic evaluations to limit the possibilities of the numerous ITs for selection. As a starting point we defined a model describing possible ITs for the selection task (section 10.2). In order to find out the most meaningful interactions described by this model, a low fidelity prototype based on drawings and photos served for evaluating these possible ITs (see appendix I.b). As we intend to implement a system supporting whole body interactions the most obvious method for an early evaluation was to act like using this system, without having an actual implementation. Such technique was for example used by Jeff Hawkins, who invented the Palm Pilot, and carried around a block of wood in his everyday life which he used as a prop to imagine different situations of usage. The technique was later formalized and described as Playacting (Sato & Salvador, 1999). Results and more details on the used methodology can be found in the appendix, section I.

Formative evaluations2 are used in the medium stages of development, where early system implementations are evaluated. In practice these informal evaluations serve for refining parts of the implementation, design, or the metaphors on which the design was based (LaViola et al., 2009). In our work these evaluations involved observational and informal user studies using the implemented system, in order to identify usability issues with the implemented ITs. For the methodology observations of users interacting with the system and interviews was used. These included a "think out loud" approach, where the users commented on the implemented ITs while using them. Observations were used to find out and refine certain aspects of the implemented ITs. This formal evaluation was based on a formal test design, with the independent variables being ITs and dependent variables the task scenario, the environment and system. The methodology, ecological validity, results and a statistical analysis are included in the test chapter, where more details can be found.

CHAPTER 3

PROPOSED TECHNIQUE

3.1 ANALYSIS

Based on the conclusions from the previous discussion on interaction techniques in VEs, the focus of this investigation will be on selection techniques. Hence the following chapter will first focus on a task decomposition of selection techniques, here the main inspiration will derive from Dough Bowman's taxonomy, and therefore a presentation of his taxonomy will first be given. From this foundation our contribution to a potential interaction selection model will be presented and discussed. Furthermore the considerations made in the pre analysis concerning design principles will form the bases of our effort towards a unified model of reality and imagination based principles. Finally a requirement specification for selection techniques will be constructed from the knowledge gained in the pre analysis and analysis, setting out the consideration needed for the design.

3.1.1 SELECTION TAXONOMY

One of the key figures in the area of 3D user interfaces is Doug Bowman; he and his colleagues (2005) have investigated different existing interaction techniques and have decomposed them into a taxonomy an approach that also will be addressed in the current context.
Taxonomies have originally been used to classify organisms, creating a structure where one can have a set of higher and lower levels of categories within a species (Encyclopedia Britannica, 2011). This notion has been adapted over to 3D User Interface by Bowman and colleagues (2005) who calls it *Generic technique-decomposition.*

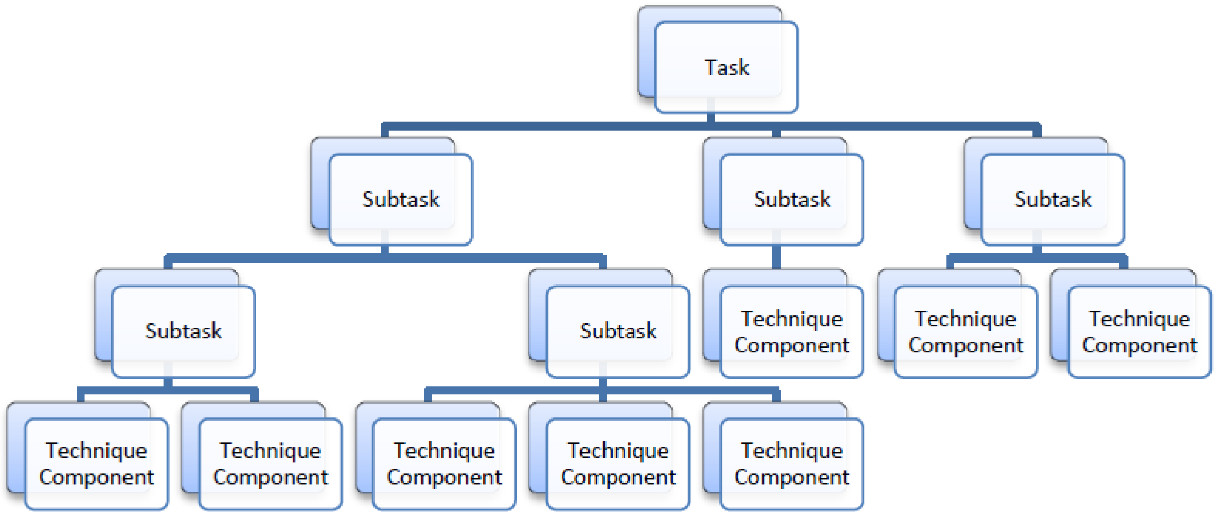


Figure 9 - Shows how each task can have a subtask, and each subtask can have another subtask of technique component. The figure derives from (Bowman et al., 2005, p.353).

Our approach is inspired from Bowman and colleagues (2005) and we make use of a taxonomy to define the selection techniques in order to fully understand all the elements that have to be taken into consideration when designing a selection technique for a VE.

3.1.1.1 PREVIOUS TAXONOMIES

Lindeman (1999) constructed a taxonomy of manipulation techniques. In the realm of manipulation there is also selection, meaning that a manipulation technique is a combination of selection and manipulation. Lindeman (1999) divides manipulation techniques in to two different categories, direct and indirect manipulation. His direct manipulation is determined of how closely user movements are mapped to objects movement, and indirect manipulation uses symbology or mediated tools for manipulation.

Lindeman's (1999) device oriented taxonomy focuses on input/output devices, forming the final three axes of this taxonomy which are: the *manipulation parameter* (the manipulation is direct or indirect), the *action type* (the action that is required to perform the technique is discrete or continuous), and lastly the *degrees of freedom* the user can use the technique to physically manipulate any given object.

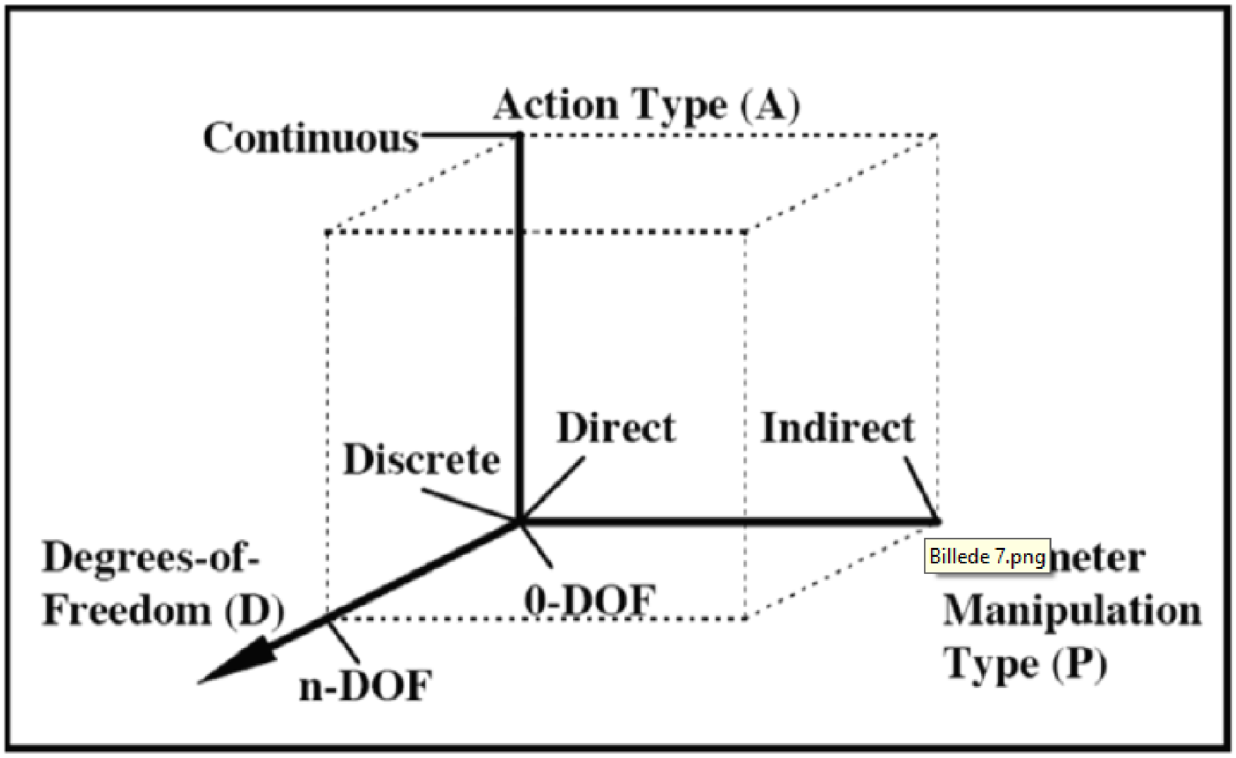


Figure 10 - Lindeman's IVE Interaction Taxonomy, having three axis, Parameter of Manipulation type "Direct - Indirect ", Degrees of Freedom "0 DOF - n DOF" and lastly Action Type "Discrete - Continuous". Source: (Lindeman, 1999)

Although Lindeman only focuses on manipulation techniques, his taxonomy points out some of the dimensions to be taken into account, when designing ITs.

Jian Chen and Doug Bowman (2009) have a different approach in decomposing interaction techniques into a series of tasks. As already introduced in the pre-analysis their description of ITs consist of four universal tasks "Navigation, Selection, Manipulation and System Control". This builds upon Bowman and colleagues' (2005) work within 3D user interfaces.

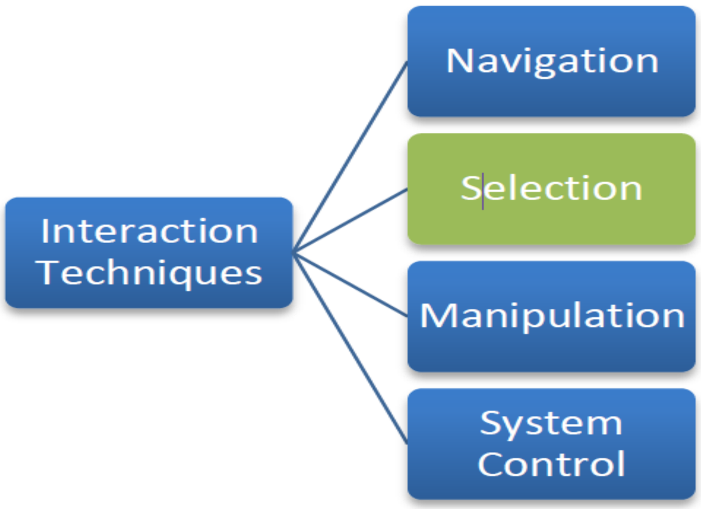


Figure 11 - The four universal tasks that according to Chen and Bowman are within the domain of interaction techniques. Selection will be the main focus of current investigation of interaction techniques. The figure derives from the article (Chen & Bowman, 2009).

Each of the interaction techniques subtasks has other subtasks. In order to construct a single interaction technique the lowest level of subtasks are combined.

Bowman (1999) constructed a structure to selection/manipulation techniques, applying a method of combining the lowest level subtasks in selection and manipulation, concluding that a total amount of plausible techniques were 4608, reducing the number with dependencies and constraints in the design space to 667. This example gives a good indication of the possible interaction techniques by combining the different lowlevel subtasks.

Since the focus of our work is on the task of selection, we need to fully understand selection tasks within VEs. In order to do so it is necessary to define and deconstruct the term selection. Doug Bowman and colleagues (2005) define selection, as "*Selection is the task of acquiring or identifying a particular object from the entire set of objects available… The real-world counterpart of the selection task is picking an object with a hand*" (Bowman et al., 2005, p.142) and from this definition they constructed a taxonomy of selection techniques. Bowman & colleagues (2005) stated that selection techniques have three subtasks *indication of object, confirmation of selection* and *Feedback.* Indication of object has four subtasks and three of these subtasks have technique components. Confirmation of selection and feedback has each four technical components.

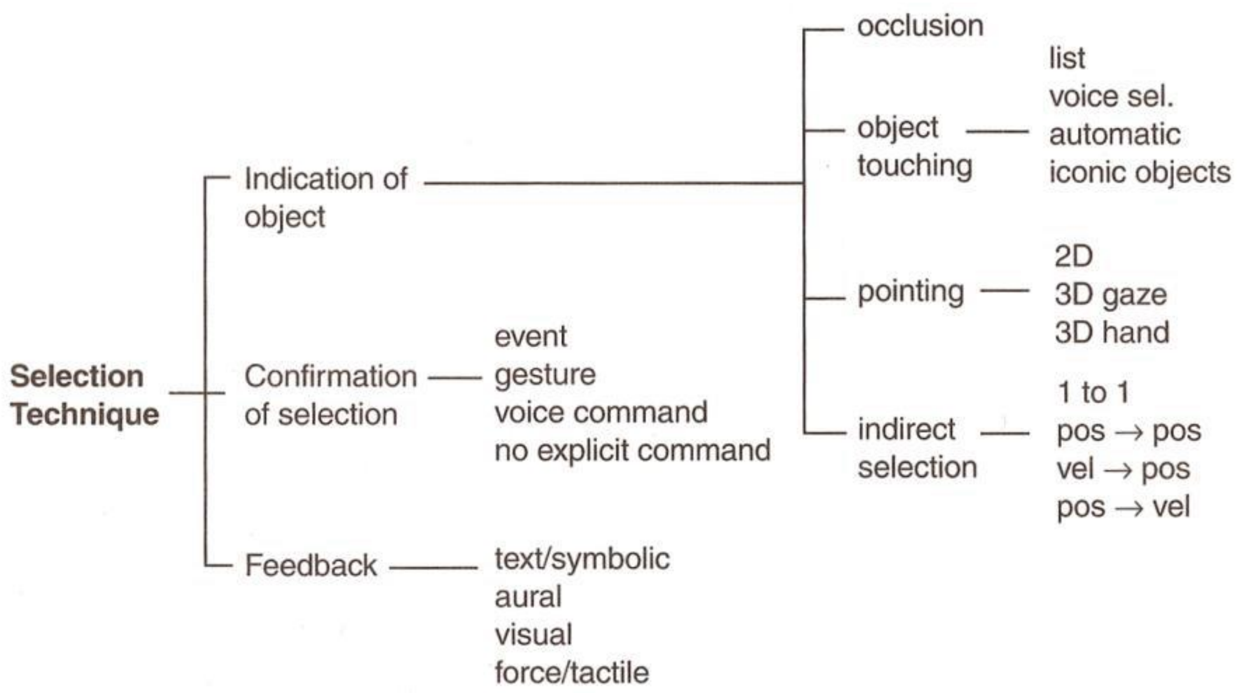


Figure 12 - Bowman and colleagues taxonomy of selection techniques, image source (Bowman et al., 2005, p.149).

Lindeman and Bowman have two different approaches towards describing interaction techniques. Lindeman's focusing on the input/output devices to describe the techniques, the drawback of his taxonomy is that it gives a set of parameters to be taken into consideration in each technique, but it does not give a detailed description of each element in the technique, elements that can be implemented in a software environment.

Bowman and colleagues on the other hand have decomposed each interaction technique into task elements; this approach allows a good understanding of the technique and assists when implementing the technique into a software environment. Further on, this method supports the notion of reusing elements from one technique to another technique that may be needed in a specific situation or application, an important aspect that is often seen in software developments.

Although agreeing with Bowman and colleagues task oriented approach, our critique is that their model is only based on the known techniques applied into the taxonomy, resulting in a taxonomy that becomes a static model for such ITs. Thus the development of novel techniques from this taxonomy only happens by combining existing techniques.

Bowman & colleagues selection taxonomy does not pay much attention to design principles; we argue that this is an area that should be paid attention to when defining selection techniques. They touch upon the subject when they define manipulation techniques, mentioning isomorphism in the regards of having real and magical actions corresponding to the user's actions in the VE. They correlate this subject to mapping of the users physical actions to system feedback.

We agree that this is a part of the answer, but we see these mappings as a product of a set of design principles. We argue that having an interaction taxonomy with design principles as a point of origin will leave a model that will be more generic and resistant for technical developments within the field of VE. The correlation between these two will be described in details in the principles for VE & interaction techniques section.
Furthermore the results of their decomposition are lacking a more detailed description of the system responses to the user's actions. Bowman and colleagues (2005) state that interaction in a selection task should "*provide the user with the means to indicate an object to select and confirm the selection; and provide …feedback while performing the task*" (Bowman et al., 2005). It becomes clear from this statement that in order for a selection task to be possible, there are two main actors in the scenario: first the user that is indicating and confirming the selection and secondly the system is providing the user with feedback on his actions; hence there is an input/output relationship between the user and system. Bowman and colleagues (2005) taxonomy is lacking a more detailed description of these actors, e.g. what are the constraints and feedback options of both parties? These missing links will be addressed in our taxonomy that will be further described in the following section.

3.1.1.2 PROPOSED TAXONOMY

The proposed taxonomy takes its foundation in Bowman & colleagues selection task taxonomy, assisted by our investigation of current selection techniques and selection task decomposition of selection techniques. Results from this approach indicated that a selection technique was a product of a more complex structure than earlier researchers have demonstrated, a structure that will be scrutinized in the following section.
We have reached the conclusion that the structure of a selection technique is as following: the lowest level is human physical and VE hardware (software) input to the system, this aspect has been introduced in the pre analysis, and therefore not repeated in this chapter. Next level is where the

main tasks of selection technique *Indication, Confirmation and feedback* take place. Higher levels are interaction techniques and finally principles of design. Further description of the different levels and relational description between the different levels will be addressed in the following sections.

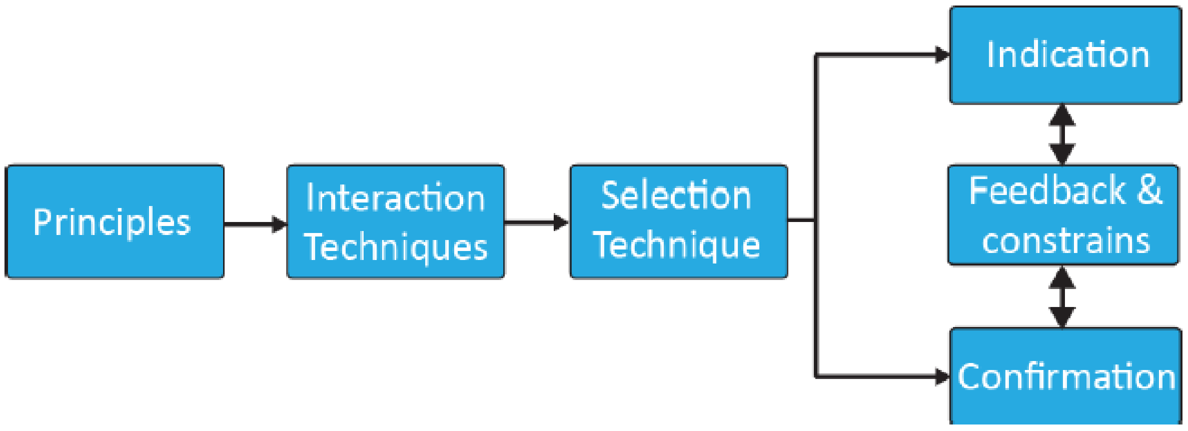


Figure 13 - The figure depicts how selection techniques can be described.

The description of the different elements that are situated in correlation with the selection techniques will be divided *low level* (e.g. indication in selection technique) and *high level* (e.g. interaction techniques) related subjects. Lastly a final relation description of how all the elements are related to each other.

3.1.2 SELECTION TECHNIQUE

Our proposed taxonomy is task oriented. It differs from Bowman and colleagues' in the respect that we clearly acknowledge the actors included in the different selection tasks in a Virtual Environment, whereas Bowman (1999) when defining selection "*Selection involves the specification of one or more virtual objects by the user for some purpose*" seems to have neglected the system that responds to the users actions and provides the user with proper feedback. Bowman (1999) mentions feedback as a subcomponent of selection, but regards it purely as an interaction issue, which does not correspond to the actual users' goal. We do not agree with this statement, building on the Human Computer Interaction (HCI) notion having a system of user and computer actions. We are bound to the fact that each users action has an impact on the system and the system has an impact on the user, a symbiosis that needs to be taken into account and cannot be regarded as purely feedback of actions; thus the computer system that the VE is a part of needs to be considered when describing selection techniques.

The consequence of this notion can be seen in how the subcategories of selection techniques are related to each other. The user's *indication* and *confirmation* are closely connected to the system *VE Properties*. Furthermore *indications* and *confirmations* subcategories are dependent on user and computer system constraints.

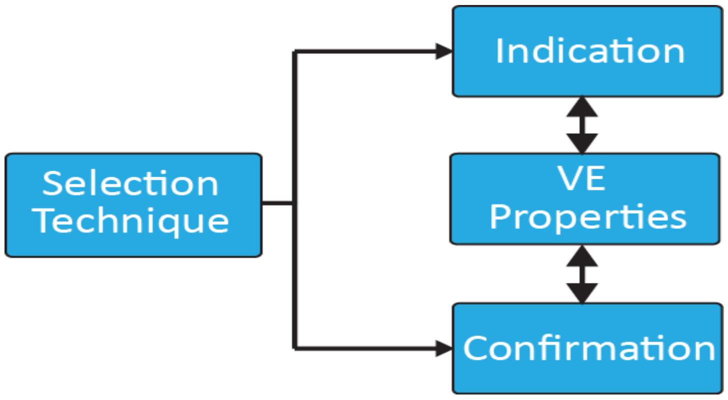


Figure 14 - The sub categories of selection technique and how they are related to each other.

Indication has two subcategories, *direct* and *indirect* selection; further details can be seen in Figure 14. We define direct selection as the task where the user needs to touch the object in order to select it. The technical component of direct selection is *object touch*, which should be considered in regard to the users' virtual attributes. If the users are able to virtually touch the object, we regard it as direct selection. That could be considered as pointing gesture from a programming perspective an indirect selection, but the main difference here is that the ray is used to touch the object directly, opposite to an indirect pointing technique. In this context the virtual arm is considered as an extension of the users arm. Thus the direct selection also requires no further confirmation action but selects objects directly upon touch. There are two technique components and one subtask in indirect touch. These are *Head Up Display* (*HUD) pointing* and *Gesture*. The technique component HUD is defined as a mediated method of choosing an object, here selection is possible through lists or iconic representation of the object. Gesture can be divided into two technique components. First is posture, a static gesture that does not change position over time. An example can be pointing and the main difference between pointing in a direct and indirect selection is the feedback and constraints relation between the user and system. Dynamic gesture is considered as a posture that has a dynamic movement over time. One example of such dynamic gesture is lasso tool selection in Photoshop1, where the user is drawing a line that is circling all the objects he wants selected.

Pointing can normally be categorized as a static gesture, but due to the vast amount of implementations versions of this technique component, it will remain as independent category. An example of pointing technique is the ray casting technique.

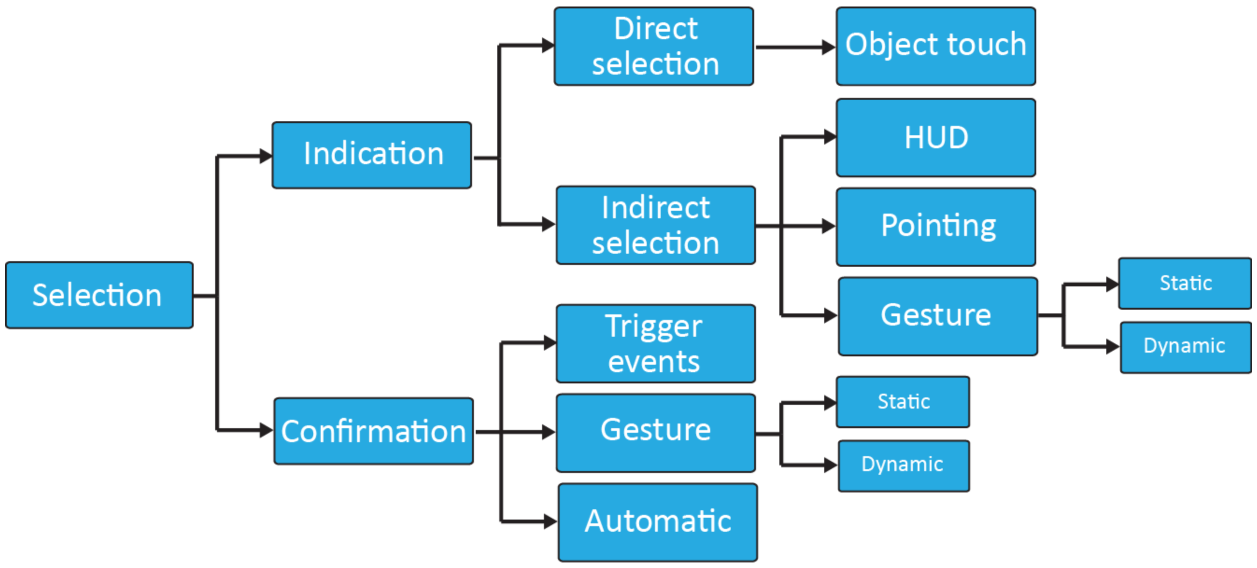


Figure 15 - Shows our proposed taxonomy of selection techniques

The task of confirmation consists of three technique components: *Trigger event, Gesture and Automatic.* Trigger event is to be considered as the discrete action taken to confirm the selection. One example can be the push of a mouse button. Gesture of confirmation has the same properties, as in indication, hence is the division of static and dynamic gesture; these can also be regarded as discrete and continuous actions. Automatic confirmation is either to be considered as a direct confirmation from object touch or the confirmation through a time threshold, where the user needs to indicate an object for a certain amount of time in order to select it.

As stated earlier a selection technique is a relationship between Indication, Confirmation and VE Properties. Feedback is a part of VE Properties and is regarded as the communication link between the user and computer system. A link which according to Donald Norman (1990) can be considered to be a part of design principles. Recognizing the individual designer's preferences when dealing with feedback, our taxonomy of feedback will only give subtasks of the subject.

Feedback has four subtasks: *Text/Symbolic, Aural, and Visual* and *Force/Tactile feedback*. Most of the subtasks are self explanatory by their names. Aural is audio feedback the user gets when confirming or indicating a selection, such as for example found in Windows when double clicking on a folder. Visual feedback can be used in different aspects, such as a guiding tool for the user to navigate in the scene with his selection tool; one example can be the ray extending from the user hand when using the ray casting selection technique. Force/Tactile feedback is the physical feedback the user gets when interacting with the system; one example can be the Nintendo Wii remote controller that is vibrating when the user is performing various tasks. Text/symbolic subtask could be regarded as a visual technical component, but we regard it as an independent subtask due to the semantically content of this task. One example of textual feedback can be a list produced to identify objects in the scene.

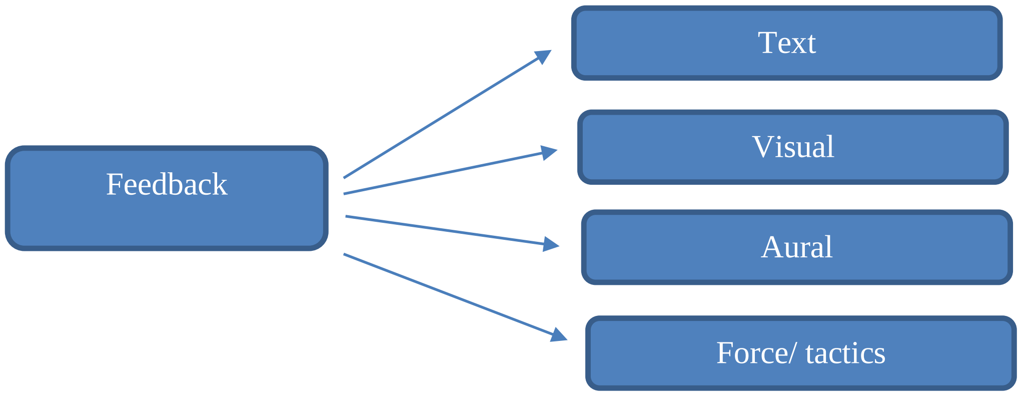


Figure 16 - The different feedback options available for ITs

System constrains are also considered as a part of VE Properties. System constrains finds it roots in Donald Normans design principles (Norman, 1990). Constrains is to be regarded as a guiding tool the system gives to the user in order for the user to more easily understand the system and to be more capable to perform actions that are a part of the designed system. System constrains have three sub elements: *Motion*, *Visual* and *Aural*.

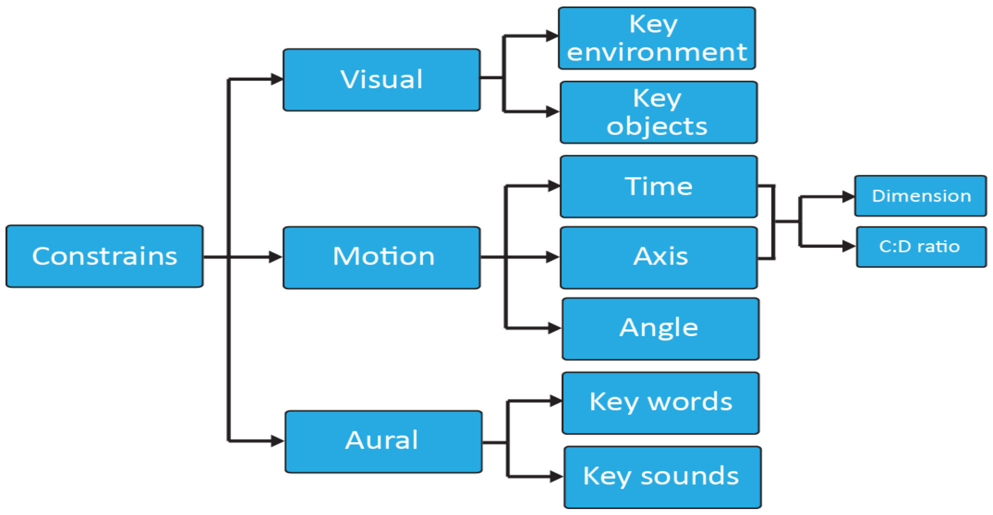


Figure 17 - Depicts the constrains that can be added to the interaction technique

Motion has three sub elements: *Time*, *Axis* and *Angle*. Time is considered as sub element of motion, due to the factor of movement, hence time can be a constrain factor in determining velocity of a movement. Another time related issue is when the user is confirming his action; time can be a determining factor in regards to respond of action.

Axis can be defined by its technical components *dimension* and *control display ratio*. In an axis point of view, a dimension is the describing factor of the virtual environment space the user is operating in. The number of dimensions describes the minimum number of coordinates that are needed to describe a point within the Virtual Environment. One example can be three-dimensional virtual environments here a point can be described by the xyz coordinates. Control display ratio is how user actions in one dimension are mapped over the virtual environments dimensions. An example of such mapping can be the users three-dimensional actions are mapped to a two dimensional environment.

Angle has two sub-categories: *dimension* and *control display ratio*. Dimension is describing to which degree it is possible to rotate a point on each of the axis. Three-dimensional spaces have the three axis that a point can rotate around. Control display ratio is how the user's rotational actions can be constrained by the system; one example can be that the user's 360-degree movement on the x-axis is mapped to have a span of 180 degree in the VE.

Aural and Visual constraints are regarded to be the systems internal infrastructure of audio signals. An unconstrained audio environment within VE can be considered as a simulation of the real world. But due to the complexity of such a system, a united understanding of system commands across all cultures is unlikely to be achieved. Therefore to guide the user to understand the system, aural constraints are divided into keywords and key-sounds, while visual constraints are divided into key environments and objects.

Having described how all the sub tasks in a selection technique, it should be clear that in order to create a selection technique, elements in the indication, confirmation and feedback/constraints are needed. Furthermore the selection technique is also dependent on the hardware and user input to the system. The overall relationship is described in Figure 18.

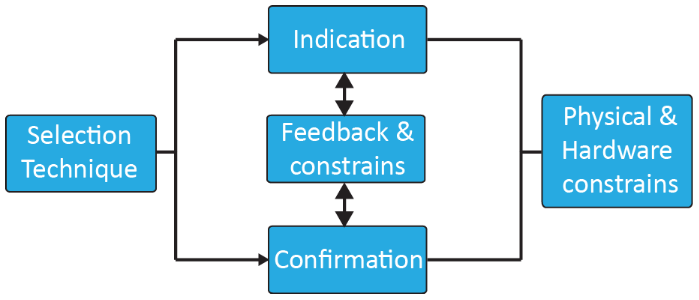


Figure 18 - The relationship between the elements that constitute a selection technique

Further consideration are needed when creating a selection technique, and that is design principles, a subject that has been touched upon in this section but will be further described in the following section.

3.1.3 PRINCIPLES FOR VES & INTERACTIONS

In this section we will present our idea based on a combined model of reality and imagination based principles. The goal of our model is to describe the principles that are particularly suited for body controlled interaction in VEs or more generally in 3D interfaces. As already described in the pre analysis there are two opposing set of principles: reality based (RBI) and imagination based (IBI) principles. The source of our model is the RBI framework (Jacob et al., 2008), which is described through the four main categories: *naïve physics*, *body awareness*, *environment awareness* and *social awareness*. From these RBI principles we derived the imagination based principles (IBI), which fits to each of the RBI categories and can be considered as an opposition to the RBI principles. Our proposed model can be seen in Figure 19. In the following paragraphs we will explain this model and give some practical examples on how to utilize such principles in the design of ITs and VEs.

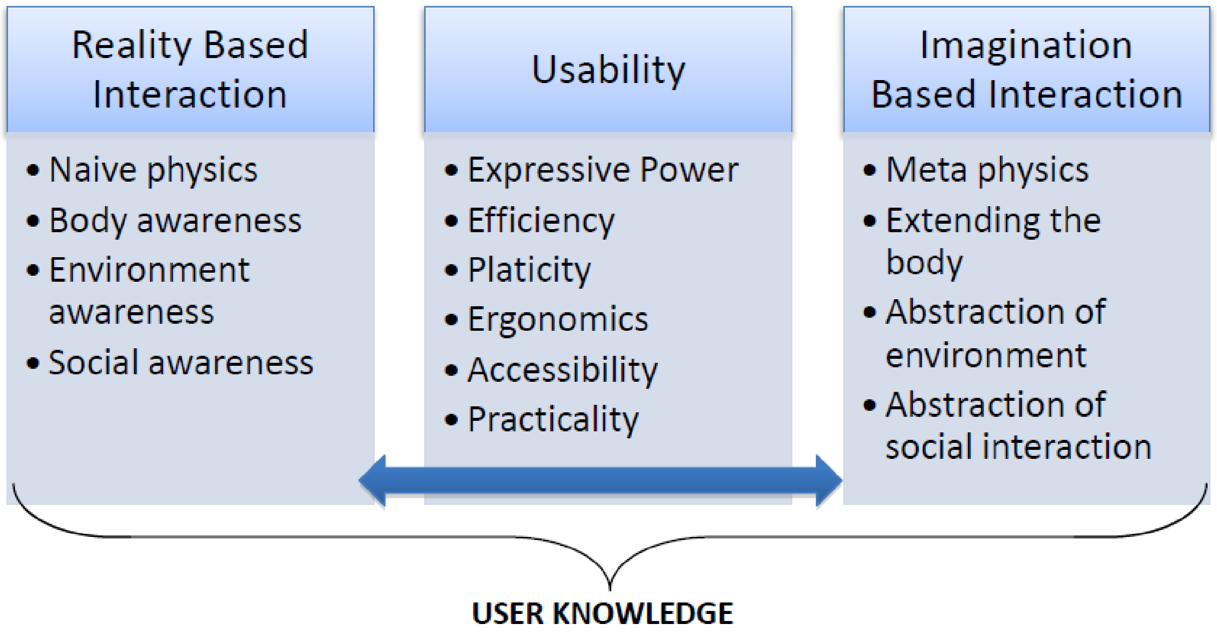


Figure 19 - The relationship between reality and imagination based principles and their relation to usability. (Source SimpleOpenNI.com)

All principles in our model utilize the users pre-existing knowledge. The RBI principles build on user's everyday knowledge and it can be considered as expectations towards the system of what the user normally would encounter in the real world. The IBI principles on the other hand build upon the users' knowledge from various other sources than the real world, such as movies, games, books, dreams or in general imagination. It is important to note that this knowledge is highly subjective and thus the utilization of principles should always be considered in a cultural context. The RBI principles may though provide a greater common cross cultural understanding than the IBI principles, as they build on the more direct perceivable phenomena than principles rooted in imagination. We placed the usability between the IBI and the RBI principles, since we consider the usability as the tradeoff between the two opposing set of principles. In this context we want to highlight that basing a system solely and strictly on either of the two opposing principles may result in a loss of usability. This will be further explained throughout the following sections.

3.1.3.1 NAIVE PHYSICS AND META PHYSICS

Basing ITs on the naïve physics principle should be considered as mimicking the expected real world behavior. For example could an interaction with a virtual object result in a physical response of the object. Considering manipulation techniques, the object could for example be rotated by direct touch and should act under the influence of gravity, friction and the users input forces. For selection techniques the principle could be utilized by providing physical response as feedback on a selection action. For example could a push result in a physical corresponding action of the object. By this the direct selection, which we describe in our taxonomy, is highly connected to the naïve physics principle, since it relates to the corresponding real world action of selecting objects which is carried out by simply grabbing or touching a real object in order to manipulate it.

3.1.3.2 BODY AWARENESS AND EXTENDING THE BODY

The principle of body awareness refers to familiarity and understanding that users have of their own bodies. Drawing its strength from proprioception, the principle is especially used in many of the ITs found in VEs. For selection techniques, the principle is present by using arms and hands to point at objects, in contrast to the mediated (through the mouse) style of interaction found in the WIMP paradigm. The principle can be supported by providing the user with feedback of his/her body, which in VEs often is utilized through displaying an avatar of the user. The use of full-body motion tracking further supports this concept, where users' body motion can be directly translated into the system. We consider the RBI principle of body awareness as building strongly on user expectations of his/hers own body and its virtual counterpart. Utilizing this principle to its full extend would be to represent the user one to one in the VE, meaning that the avatar by which the user is represented actually matches the user.

The principle of body awareness can be bent and extended, and the user may accept a variety of representations to be him/her-self. Thus the opposing principle to body awareness can be considered as an abstraction of the more realistic representation of the user. By this the principle of "extending the body" builds on the concepts found in ecological perception and embodiment. As almost anything is possible, in a VE the principle can take various forms, were simple forms of utilizing the principle may be to extend the arm of a user to allow him/her to reach a distant object.

More abstract forms could even be to replace the users' representation entirely, which for example could allow the user to "be" an Orc, a Worm, a Tiger or almost any other object thinkable. By such the principle still draws its strength from proprioception, but translates it to an entirely different representation. Here it is especially important to map the input so that the user still can make sense of body actions and the corresponding actions (feedback) of the representation. Practical considerations of the body awareness and extending of the body principle should always include mapping and feedback considerations. For example the users hand position can be directly mapped to the position of a cursor for selecting objects. In such example the users hand could be represented by a cursor, which can be represented through the classical mouse courser icon or through a more matching representation of a "hand" icon. Also in many situations it may not be necessary to actually represent the user through an avatar, but the body parts used for an interaction should always be taken into consideration when choosing feedback, so the user can easily relate his actions to what is going on in the system.

3.1.4 REQUIREMENT SPECIFICATION

The intention with our taxonomy and the described principles in this analysis chapter is to be used in the practical design of selection techniques. As already described in the problem statement and methodology (section 8 & 9) our goal is to implement and evaluate a set of ITs. Since there are many possibilities for creating techniques we define a set of requirements for the design and implementation of these techniques. This is also helpful to limit the amount of techniques and to focus on the design.

- The selection techniques should reflect all major aspects of our taxonomy, which are:
  - ➢ A technique using direct selection – by object touch;
  - ➢ A technique using indirect selection – by a HUD element;
  - ➢ A technique using indirect selection – by a pointing method;
  - ➢ A technique using indirect selection – by gestures.
- Principles should be applied in the design of the techniques whenever possible.
  - ➢ The selection techniques should be usable for both single and multi-selection scenarios.

Requirements to the implemented techniques are that they meet a set of usability criteria in order to actually be usable in an evaluation context. The general usability criteria to the techniques are:

- The techniques should be easy to learn.
- The techniques should be easy to remember.
- The techniques should be effective to use.
- Using the techniques should possibly not result in making selection errors.
- The techniques should be pleasant to use and comfortable.

With the goal to develop techniques for 3D interfaces and VEs it is required to design and implement a 3D environment. As we do not work with a specific context or a specific focus with the environment, the requirements to it are broad. The goal of the environment is in general to support depth perception and to allow easy selection of objects placed in this environment. With our focus on interaction and the evaluation of these, we set out to keep this environment simple. This has the advantages of not requiring an extensive amount of cognitive resources from the user and also the intention was to keep the environment as the dependent variable in the test design. The requirements to the environment are the following:

- The environment should support depth perception.
- The environment should be simple to not confuse the user.
- The objects which users are able to select should be easy identifiable.

Besides the above described requirements to the design, another set of requirements to the hardware and implementation are described. These requirements are based on the input and output method used in the prototype and describe the more general requirements to the prototype developed during this thesis. Since we are using optical marker less motion capture with the Kinect camera as input, requirements to the prototype are that:

The input delay (response time of input or ping) should be kept as small as possible.

- At least below 500ms.
    - ➢ It should be designed to work well with the method of tracking.

The input delay is important to keep low since high response time could cause users to be confused. Executing a gesture or an action should in this way always have an immediate result in the system. Requirements to the prototypes output method are:

- Providing an adequate real-time frame rate (~30+FPS).
- Supporting a stereoscopic screen-display method.

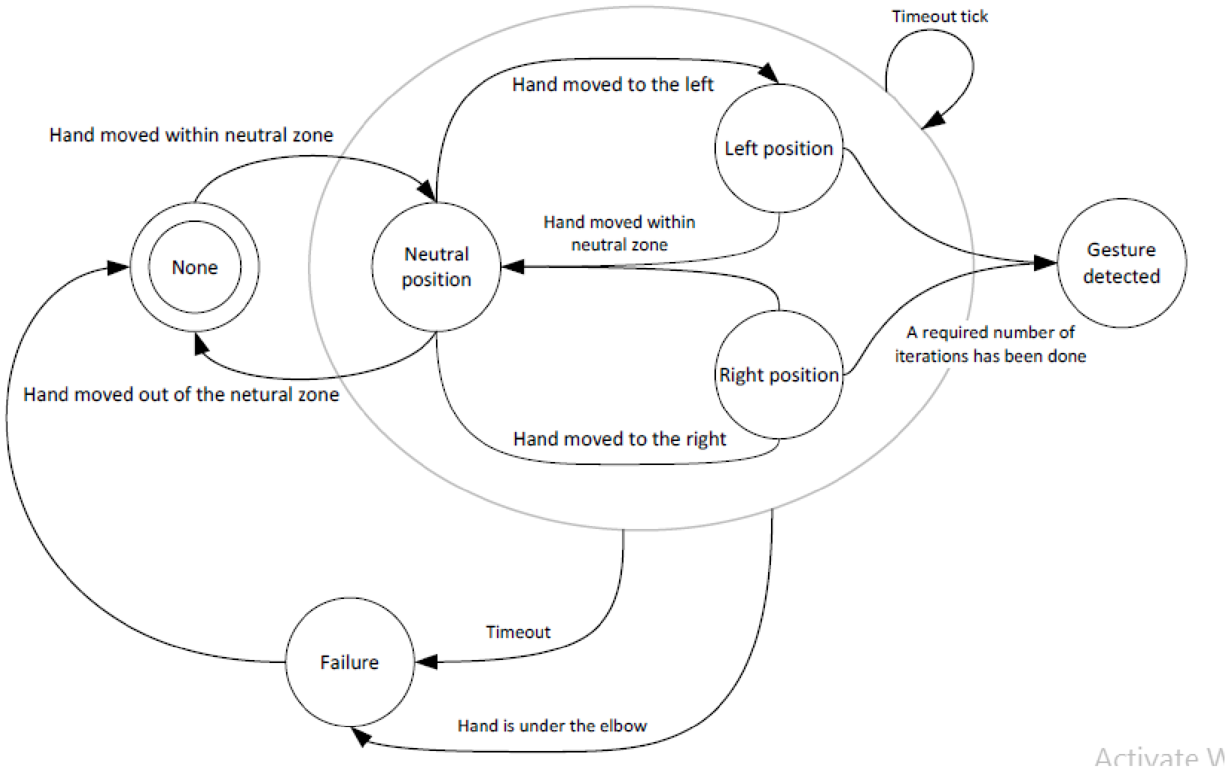


Figure 20: simple state diagram of wave gesture recognition

3.2 DESIGN

The structure of the design process used for creating selection ITs was based on an iterative process, as described in the methodology (section 9). In this process we used several creative methods to create the ITs. For the initial ideas brainstorming was used, whereas later concepts were refined through playacting and drawings. Throughout this process we created concepts for around 12 different techniques. Some of the techniques we include are already existing techniques or based on them, where others are new techniques. In the iterative process we cut down the number of techniques and evaluated the most usable and feasible for implementation. The outcome was 5 techniques which were implemented and later tested in a user study. All of the techniques were created in the scope of our taxonomy and principles were applied in most of them. In this chapter the theoretical design of these five techniques are described, along with a range of design considerations discovered when creating the concepts of these techniques.

3.2.1 DESIGN FRAMEWORK

With the described taxonomy selection techniques were separated into *indication* and *confirmation* actions. The goal of the techniques was to cover each of the elements found in indication: *object touch, HUD, pointing* and *gesture* (Figure 20). The confirmation actions were designed separately and were assigned to the different techniques based on a usability evaluation with real users. For the confirmation actions we excluded trigger events, since they would require a discrete input device such as a button.

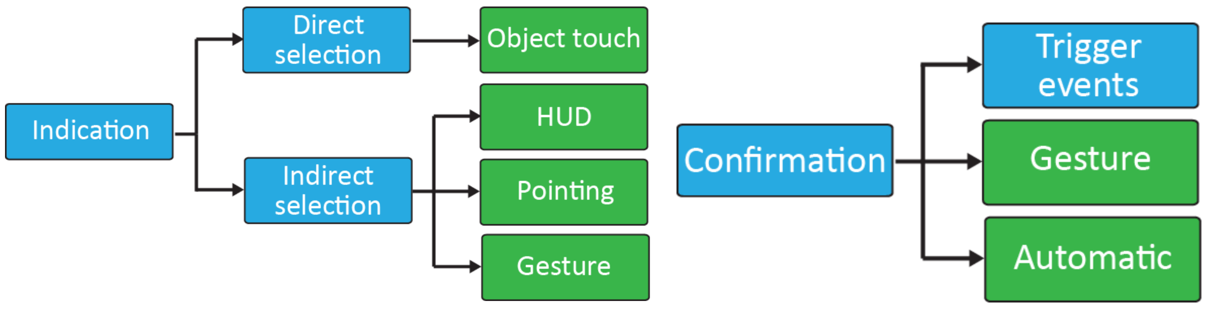


Figure 21 - Targeted indication and confirmation actions

For each of the indication methods constraints can be added, in order to guide the user into action. Also each IT require some form of feedback. In our design we included only visual feed-back, knowing that the use of multi-model feedback possibly might enhance the overall usability.

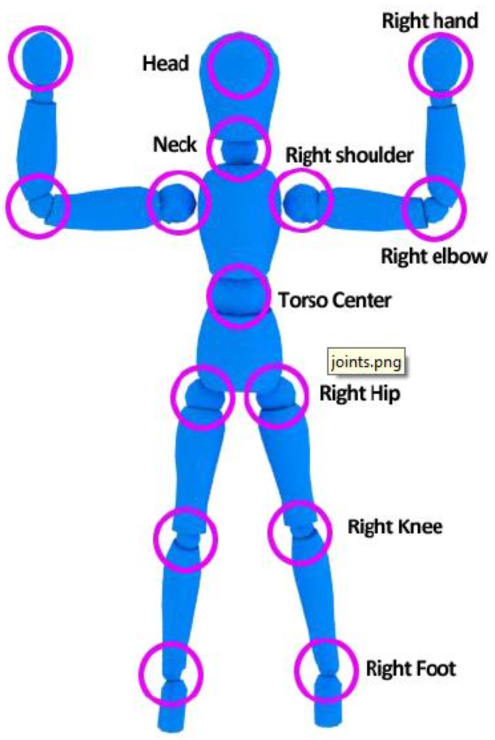


Figure 22 - Input joints available through the NITE skeleton tracking framework (simpleOpenNi)

3.2.2 INPUT AND OUTPUT

With the motivation of designing for controller free VE interaction, we optimized the ITs for screen display and for the input we limited the design space to the available skeleton tracking method (see pre-analysis section and Figure 21) for the Kinect camera. It is important to notice that a better tracking method could include tracking on more detailed joint parts, such as finger position or hand rotation etc., which in turn would give more possibilities in the design of interactions. Nevertheless are the most important body parts included in the tracking, mainly the hands and arms as well as the head position and rotation. Throughout the design cycles we tried out different body parts for indication actions, such as e.g. the head rotation. We came to the conclusion that using other body parts than the hands and arms (e.g. the head rotation) was not very usable and lacked of accuracy for fine tuned selection tasks. The different techniques described throughout this chapter therefore mainly use the arm position, rotation and hand position for interactions.

3.2.3 CONFIRMATION ACTION

The confirmation actions are thought of as the actions users need to perform in order to confirm the indication of an object to consequently select it. This is similar to the actions users perform in desktop computers, where indication is carried out by placing the mouse curser on top of an object (often icons) and the confirmation is carried out by pushing the mouse button.

The two categories for confirmation actions are gesture and automatic. With the automatic actions, either direct object touch or confirmation through a time threshold is considered. The gesture confirmation however can include numerous possible actions; in the following a short explanation of the designed and implemented actions as well as some other possibilities for gestural confirmation is given.

We divided confirmation gestures in two rough categories: simple and advanced gestures. The simple gestures include actions like raising a hand, pushing, pulling and swiping. The advanced gestures include more complex actions like e.g. drawing a question mark, a circle or a confirmation sign in the air. The swipe gestures are well known from touch enabled mobile phones, which are defined by a sideways movement of a finger. In our case the swipe gesture is carried out by moving the hand to the right or to the left from its original position. The push gesture is carried out by moving the hand forward. During the design we tried out different sizes of motion required for this gesture, and most users preferred to have only a small movement or tilt of the hand as the push gesture. Bigger motions of for example moving the whole arm forward showed to be not feasible in connection with most indication actions. The pull gesture is similar to the push gesture carried out by moving the hand backwards.

The advanced gestures that were discussed during the design of techniques were all based on a motion trajectory or path, meaning that these are actions where the user should draw some kind of symbol in the air or do a defined kind of motion. The motions that were considered for confirming a selection were to draw a check sign and circling the selection. These were however disregarded because they showed to not work well together with the indication actions.

3.2.4 VIRTUAL ENVIRONMENT

The design of the virtual environment had the goal of being used in the usability study. With this goal the design of the VE was intended to not place unnecessary cognitive load, nor confuse the user. By this the design of the VE was tried to be kept as minimalistic as possible, with the requirements of supporting depth perception. Our choice was to use a simple quadratic room as the environment. The depth perception was supported by adding parallel lines to the wall texture used as monocular cues for perspective and convergence. Shadows were also added, which serve as an important cue for depth perception. Through the placement of light and the used shading the depth perception was further enhanced. Another important feature for depth perception that was added was the use of motion parallax. This was done by mapping the virtual camera to the tracked head position of the user, thus giving motion parallax when the user moved.

Another important factor of the VE was to design the objects which users could select, through the different ITs. These objects were designed, similar to the environment, as cubes. This choice was made in relation to have easy identifiable objects. Since the goal of the environment was to be used in a test scenario, the choice of simplicity was more important than visual fidelity. In a real application this VE could be much more complex, an element which could add to the ecological validity of the test. Also the principles described could be fully utilized in the design of a complex world in conjunction with the techniques, which we see as an important point for future work

3.2.5 DISCUSSION OF TECHNIQUES

The design of the selection techniques was part of an iterative process based on the proposed taxonomy. The techniques therefore represent the two main categories of this taxonomy: direct selection and indirect selection with corresponding subcategories: object touch, HUD, pointing and gesture. Although the taxonomy was used as the framework for designing the ITs it still is very broad not suggesting particular details for the different techniques leaving room for creativity.

We consider the proposed selection technique model as support tool to the creative process of creating techniques for VEs. This model could be further developed to give the designer suggestions toward which elements a selection technique should contain in order to fit a specific purpose. In our case a selection techniques that is usable and preferred in single and multiple selection tasks. These recommendations will be supported by research done in regard to that specific subject, meaning that any potential findings in our final evaluation can be used to further define the model in regards to design recommendations for general selection techniques in a VE.

It should be noted that the design of the selection techniques is in respect to the final summative evaluation; hence the design will contain elements that in normal situations would be regarded as bad design. One example is that the current design of the techniques does not contain an option for the user to revert from a wrong selection. The reason for this is due to the test design, where selection errors are recorded without being able to revert from them. Having theses design considerations in mind the next step will be to implement them.

3.3 IMPLEMENTATION

Based on the design consideration done in previous chapter, the implementation will be presented in regards to how the selection techniques were implemented in order to be a part of a final evaluation. First an overview of the implementation structure will be presented, depicting the input sources, input data handling and output rendering. Then the OpenNI and NITE framework will be outlined in regards to the data provided and its usage for the selection techniques. Followed by the gesture section, which describes the confirmation actions used throughout the different selection techniques.

In the selection techniques sections, the implementation of previous design considerations in regards to selection techniques and user interaction with scene content will be described. The final implementation, containing considerations to how the 3D scene is organized and how the techniques are presented to the users during the evaluation will be described in the test implementation section. The output hardware will be described in the stereoscopic rendering section.

Finally a discussion section will reflect on all the decisions taken in respect to the implementation elements, detecting any potential weaknesses that should be taken into consideration when analyzing the evaluation results.

Code examples are given throughout this chapter, showing details of our implementation where relevant. All the code for this implementation can be found on the CD.

3.3.1 STRUCTURE

3.3.1.1 FIELD OF VIEW

Because the sensor works in many ways similarly to a camera, it also can see only a limited part of the scene facing it. This part of the scene that is visible for the sensor, or camera generally, are called *Field of View* (FOV) [13]. The sensor's FOV for both depth and color camera is described by the following vertical and horizontal angles in [14]. The horizontal angle is 57.5 degrees and the vertical angle is 43.5 degrees. The vertical angle can be moved within range from -27 to +27 degrees up and down by using the sensor tilt. Additionally, the depth camera is limited in its view distance. It can see within range from 0.4 meters to 8 meters, but for the practical use there are recommended values within 1.2 meters to 3.5 meters. In this range the objects are captured with minimal distortion and minimal noise. The sensor's FOV is illustrated in the Figure 23.

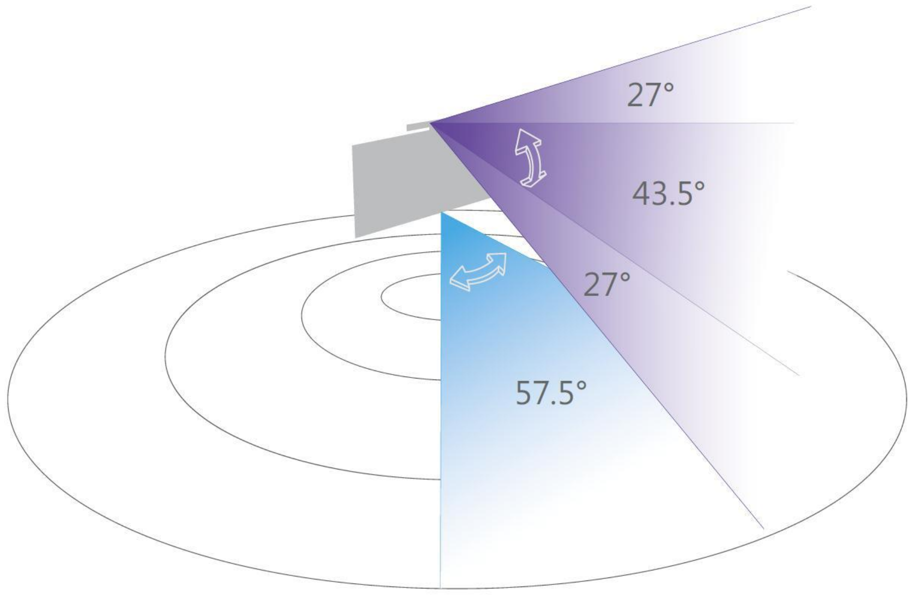


Figure 23 – Kinect for Windows sensor field of view (developer.microsoft.com)

3.3.1.2 SOFTWARE DEVELOPMENT KITS

There are several *Software Development Kits* (SDK) available for enabling a custom application development for the Kinect device. The first one is a *libfreenect* library which was created as a result of the hacking effort in 2010, at the time when Microsoft had not published public drivers and held back by providing any development kits for PC. The library includes Kinect drivers and supports a reading of a depth and color stream from the device. It also supports a reading of accelerometer state and interface for controlling motorized tilt.

Another SDK, available before the official one, is *OpenNI* released in 2010, a month after the launch of *Kinect for Xbox 360*. The *OpenNI* library was published by *the Prime Sense Company*, the author of the depth sensing technology used by Kinect. The SDK supports all standard inputs and in addition includes a *Skeletal Tracking*. Since its release an *OpenNI* community has grown and developed a number of interesting projects, including 3D scanning and reconstruction or 3D finger tracking.

The Microsoft's official SDK for Kinect was unveiled in its beta version in July 2011 and its first release was on February 2012 as the *Kinect for Windows SDK* version 1.0. Currently, there is available the newest version of the SDK, a version 1.7. An evolution and features of the SDK are described in the following chapter.

3.3.1.3 DEPTH STREAM

Information from the Kinect's depth camera are provided by the depth stream. The depth data are presented as a human body made up of picture elements that contain the distance in millimeters from the camera plane to the nearest object as is exemplified in the Figure 24

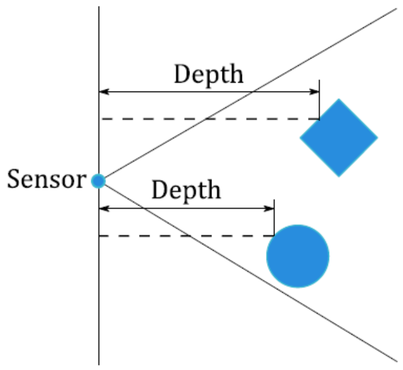


Figure 24 – depth stream values.

The pixel merges the distance and player segmentation data. The player segmentation data stores information about a relation to the tracked skeleton that enables to associate the tracked skeleton with the depth information used for its tracking.

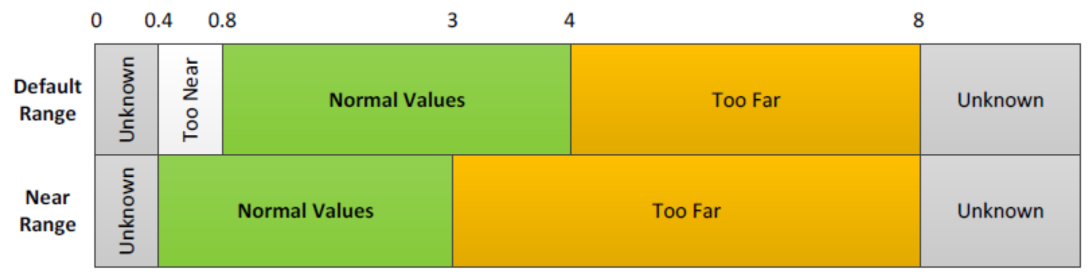


Figure 25 – depth space range.

The depth data are represented as 16–bit unsigned integer value where the first 3 bits are reserved for the player segmentation data and the rest 13 bits for the aloofness. It implies that the maximal distance stored in the depth data can be upwards to 8m. Distance from sensor in meter. The depth frame is available in different settlements. The maximum resolution is 640 480 pixels and there are also available resolutions 320 240 and 80 60 pixels. Depth frames are captured at 30 frames per seconds for all results. The depth camera of the Kinect for Windows sensor can go steady in two range modes, the default and the penny-pinching mode. If the orbit model is set to default value the sensor captures depth values in scope from 0.8 meters to 4.0 meters, otherwise when the orbit model is set to near value the sensor captures depth values in scope from 0.4 meters to 3.0 measures. Granting to the description of the depth space range described in the maximal captured depth value may be upwards to 8.0 m in both range modes. Nevertheless, the caliber of the depth value exceeding a limit value of 4.0 meters in default mode and value of 3.0 meters in near mode may be degraded with distance.

3.3.1.4 COLOR STREAM

Color data available in different resolutions and formats are provided through the color stream. The color image's format determines whether color information are encoded as RGB, YUV or Bayer.

The RGB format represents the color image as the 32–bit, linear X8R8G8B8–formatted color bitmap. A color image in RGB format is updated at up to 30 frames per seconds at 640 480 resolution and at 12 frames per second in high–definition 1280 960 resolution. The YUV format represents the color image as 16–bit, gamma–corrected linear UYVY–formatted color bitmap, where the gamma correction in YUV space is equivalent to standard RGB gamma in RGB space.

According to the 16–bit pixel representation, the YUV format uses less computer storage to hold bitmap data and allocates less buffer memory. The color image in YUV format is useable only at the 640 480 resolution and only at 15 fps.

The Bayer format includes more green pixel values than blue or red and that makes it closer to the physiology of the human eye [20]. The format represents the color image as the 32–bit, linear X8R8G8B8–formatted color bitmap in standard RGB color space. Color image in Bayer format is updated at 30 frames per seconds at 640 480 resolution and at 12 frames per second in high–definition 1280 960 resolution.

Since the SDK version 1.6, custom camera settings that allow optimizing the color camera for actual environmental conditions have been available. These backgrounds can help in scenarios with low illumination or a brightly lit setting and allow adjusting hue, brightness or contrast in order to improve visual clarity.

Additionally, the color stream can be applied as an Infrared stream by specifying the color image format to the Infrared format. It allows reading the Kinect's IR camera's image. The main purpose of the IR stream is to improve external camera calibration using a test pattern observed from both the RGB and IR camera to more accurately define how to map coordinates from one camera to another. As well, the IR data can be utilized for getting an IR image in darkness with a provided IR light source.

3.3.2 SKELETAL TRACKING

The crucial functionality provided by the *Kinect for Windows SDK* is the *Skeletal Tracking*. The skeletal tracking allows the Kinect to recognize people and observe their natural processes. It can discern up to six users in the field of sentiment of the sensor, and of these, up to two users can be tracked as the frame consisted of 20 joints that represent the locations of the key components of the user's body (Figure 27). The joint locations are actually coordinates relative to the sensor and values of X, Y, Z coordinates are in m. The Figure 26 illustrates the skeleton space.

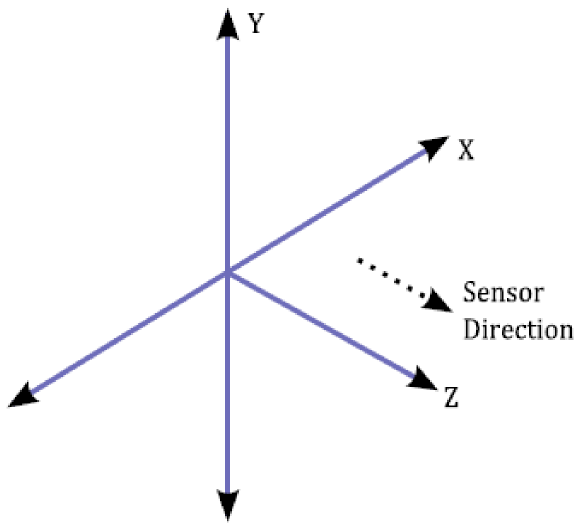


Figure 26 – skeleton space (source Kinect imagery)

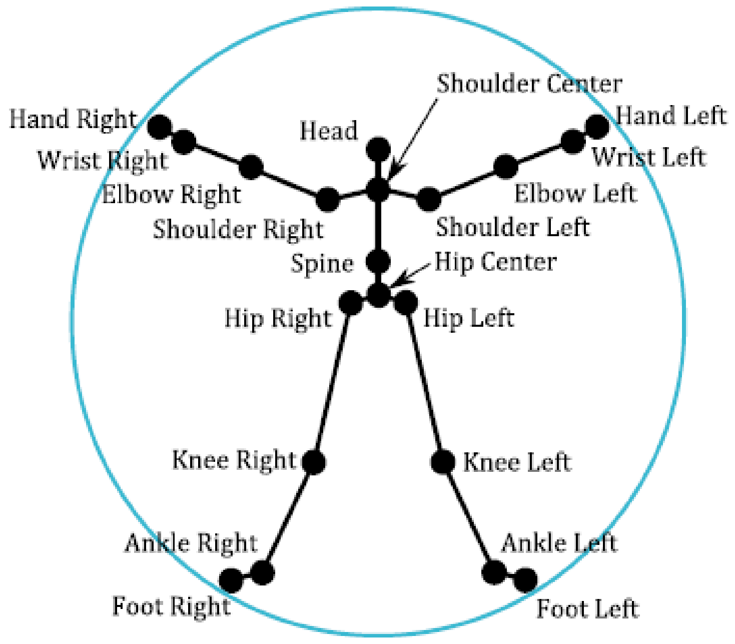


Figure 27 – Tracked skeleton joints overview (source Kinect imagery)

The tracking algorithm is designed to recognize users facing the sensor and in the standing or sitting position. The tracking sideways pass is challenging as part of the user is not visible for the detector. The users are distinguished when they are in front of the sensing element and their principal and upper body is visible to the detector. No specific pose or calibration action needs to be leased for a user to be passed over.
The skeletal tracking can be utilized in both range modes of the depth camera. By utilizing the default range mode, users are tracked in the length between 0.8 and 4.0 meters away, but a practical range is between 1.2 to 3.5 meters due to a circumscribed sphere of opinion. In event of near range mode, the user can be tracked between 0.4 and 3.0 meters away, but it possesses a practical range of 0.8 to 3 meter.

The tracking algorithm provides two ways of tracking .The *default mode* is designed for tracking all twenty skeletal joints of the user in a standing pose. The *seated mode* is intended for tracking the user in a seated pose. The seated mode tracks only ten joints of the upper torso. Each of these modes uses different line for the tracking. The default mode detects the user based on the distance of the subject from the backdrop. The seated mode uses movement to find the user and distinguish him or her from the background, such as a sofa or a chairman. The seated mode uses more resources than the default mode and returns a lower throughput on the same picture. Nonetheless, the seated mode provides the best way to know a skeleton when the depth camera is in the near range mode. In practice, only one tracking mode can be applied at a time and then it is not possible to track one user in seated mode and the other one in default style using single detector.

The skeletal tracking joint information may be strained due to the noise and inaccuracies caused by physical limitations of the detector. To minimize guttering and stabilize the joint positions over time, the skeletal tracking can be set across different frames by setting the Smoothing Parameters. The skeletal tracking uses the smoothing filter based on the Holt Double Exponential Smoothing method used for statistical analysis of economic information. The filter provides smoothing with less reaction time than other smoothing filter algorithms.

3.3.3 OPENNI AND NITE (INPUT)

The input to our system happens through the OpenNI framework (OpenNI, 2011). This frame-work is a standard interface for 3D sensor data and it is open sourced. The purpose of this framework is to define data types like depth maps, color maps and a module that generates them from the hardware, for 3rd party developers. The NITE framework (PrimeSense, 2011), is a middleware component utilizing the data from the OpenNI framework in order to output tracked positions of the user. NITE also provides gesture detection, which however was not used in our project, since we created our own.

The implementation in Unity3D is based on a wrapper provided by OpenNI. This wrapper1 has been written by Shlomo Zippel. For the purpose of this project a range of changes has been applied and functionality added to the wrapper.

The starting point for this wrapper is a static implementation of the external C functions provided by NITE, here is an example of one important function (in C#):

```
[DllImport("UnityInterface.dll")]

public static extern bool GetJointTransformation(uint userID, SkeletonJoint joint, ref
SkeletonJointTransformation pTransformation);
```

The wrapper handles in this way all functionality with the Kinect: it starts the camera, shuts it down, displays the depth map and most importantly tracks the user. We structured the functionality of the wrapper further by adding a general skeleton class (MasterUser), which holds all the tracked data points and added access functionality to this class.

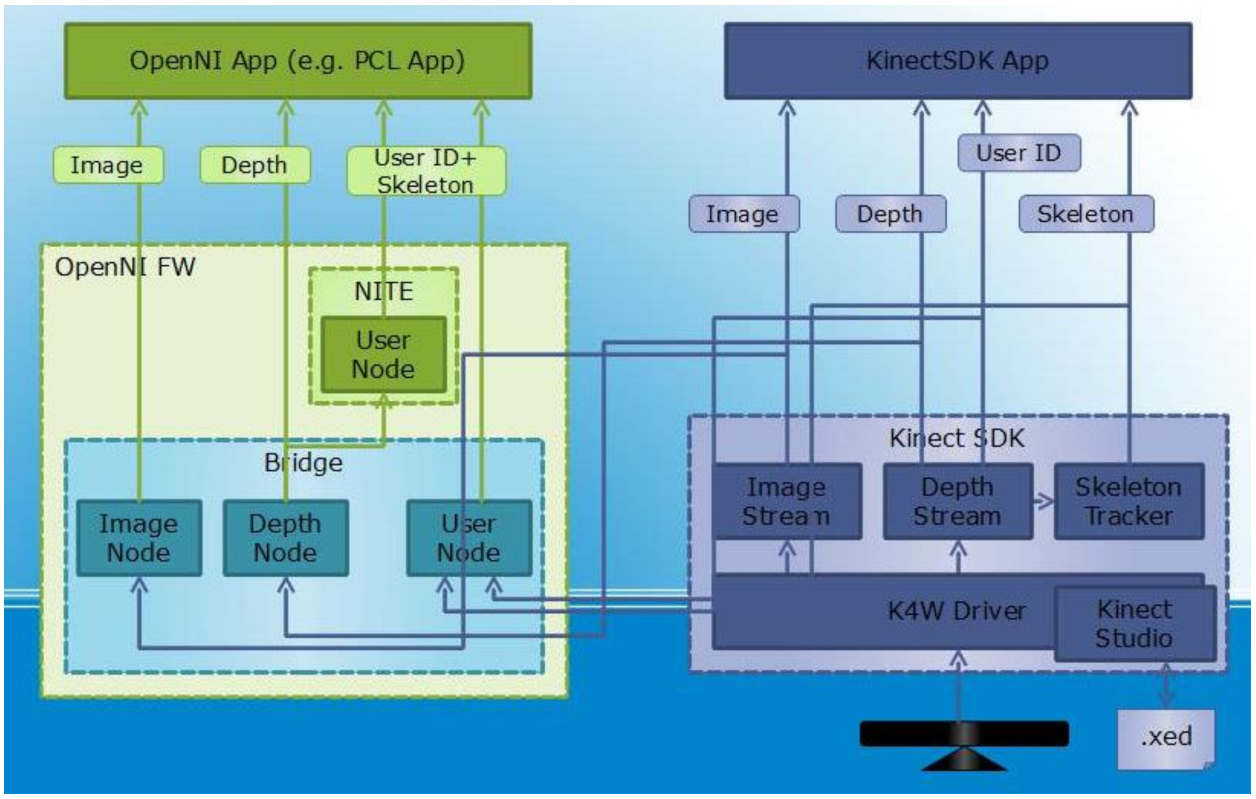


Figure 28 - OpenNi structure explaining the data flow (Source SDK Kinect)

Overall the functionality is laid out in the way that when the camera is running it starts looking for users and when it finds a user calibration can be started. The calibration requires the user to hold a “psi-pose” (Figure 30) for around 3seconds, after which the user is calibrated and tracked.

With a calibrated user, tracking is running, which updates all joint positions on every frame. From here the different ITs can be added in order to allow interaction with the scene content. In order to fulfill its goal, OpenNI released an open source framework called OpenNI Framework. It provides an API and high-level middleware called NITE for implementing hand/skeleton tracking and gesture recognition Kinect was the first implementation of the PrimeSense reference design; its optics and microchip were developed entirely by PrimeSense. Then Microsoft added a motor and a three-axis accelerometer to the design. This is why OpenNI doesn’t provide access to the motor or the accelerometer; they are specific to the Kinect implementation.

PrimeSense is a fabless (fabrication-less) semiconductor company. It makes its revenue by selling hardware and semiconductor chips while outsourcing fabrication. It’s mainly a B2B (business to business): it sells solutions to manufacturers who insert these solutions into consumer products. This is exactly the way in which it was involved in the development of the Kinect with Microsoft. PrimeSense then sells its technology to manufacturers such as ASUS and other computer or television manufacturers. But for this market to be developed, there needs to exist an ecosystem of people creating natural interaction based content and applications. PrimeSense created OpenNI as a way to empower developers to add natural interaction to their software and applications so this ecosystem would flourish.

3.3.4 STRUCTURED-LIGHT 3D SCANNING

Most structured-light scanners are based on the projection of a narrow strip of light onto a 3D object, using the deformation of the stripe when seen from a point of view different from the source to measure the distance from each point to the camera and thus reconstitute the 3D volume. This method can be extended to the projection of many stripes of light at the same time, which provides a high number of samples simultaneously (Figure 29).

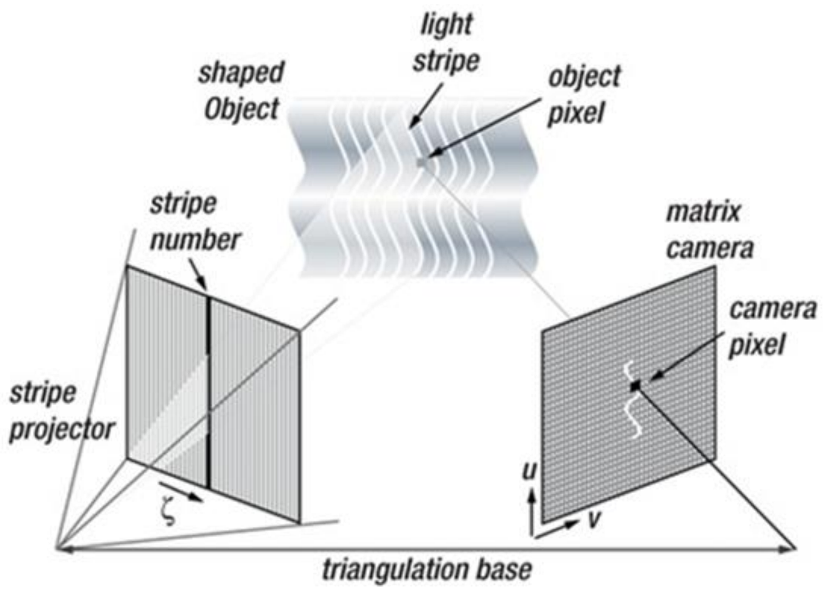


Figure 29. Triangulation principles for structured-light 3D scanning(image courtesy of PrimeSense)

Converting the Light Coding Image to a Depth Map Once the light coding infrared pattern is received, PrimeSense’s PS1080 chip (Figure 29) compares that image to a reference image stored in the chip’s memory as the result of a calibration routine performed on each device during the production process. The comparison of the “flat” reference image and the incoming infrared pattern is translated by the chip into a VGA-sized depth image of the scene that you can access through the OpenNI API

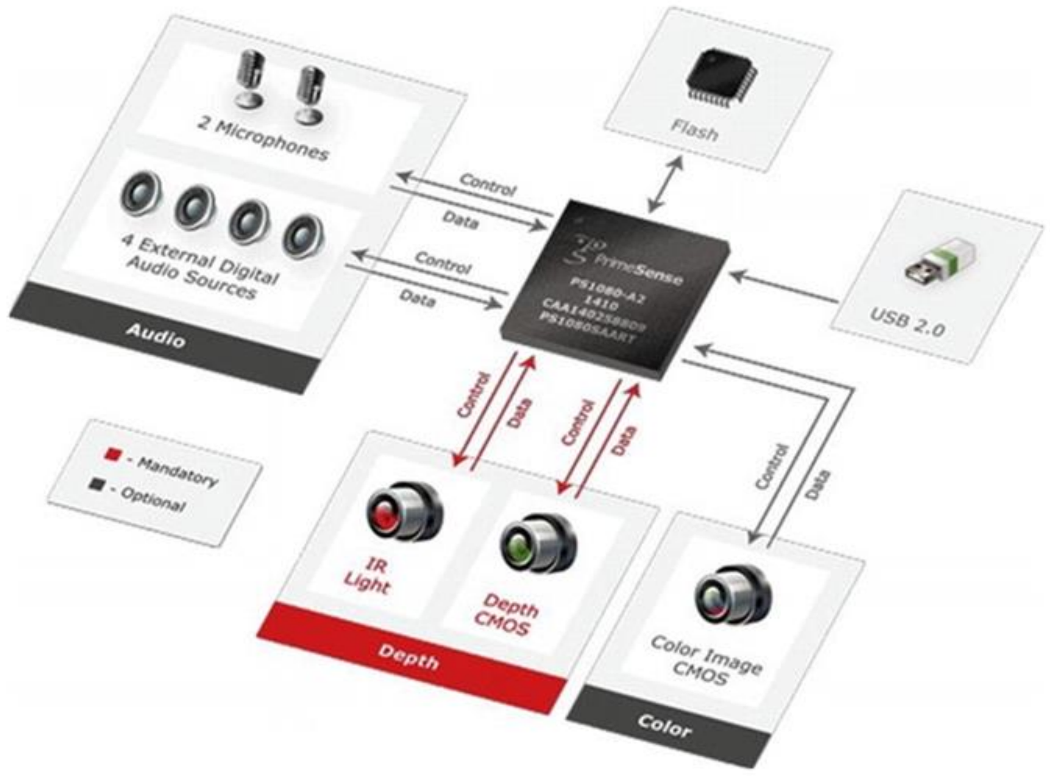


Figure 30. PrimeSense PS1080 system on chip (image courtesy of PrimeSense)

3.3.5 ACCESSING THE DEPTH MAP AND RGB IMAGE

First, import the Simple-OpenNI library, and declare the variable kinect that will contain the Simple-OpenNI object, like so:

```
import SimpleOpenNI.*;
```

SimpleOpenNI kinect;

Within setup(), initialize the kinect object, passing this (your PApplet) as an argument. the Kinect has a standard RGB camera and an infrared camera on board, used in combination with an infrared projector to generate the depth image for 3D scanning. Enable the depth map and RGB images in your kinect object and the mirroring capability that will mirror the Kinect data so it’s easier for you to relate to your image on screen. Then, set your sketch size to the total size of the RGB and depth images placed side by side, so we can fit both in IDE frame.

```
void setup() {

kinect = new SimpleOpenNI(this);

// enable depthMap and RGB image

kinect.enableDepth();

kinect.enableRGB();

// enable mirror

kinect.setMirror(true);

size(kinect.depthWidth()+kinect.rgbWidth(), kinect.depthHeight());

}

void draw() {

kinect.update();

// draw depthImageMap and RGB images

image(kinect.depthImage(), 0, 0);

image(kinect.rgbImage(),kinect.depthWidth(),0);

}
```

By writing that code we can acess the kinect device streaming . so now we can deside wheather we are doing thing right or wrong.

3.3.5.1 THE THIRD DIMENSION

This might sound less intriguing than talking about the fourth or the fifth dimension, but it is still a subject that provokes many a headache to anybody getting into the world of computer graphics, so let’s take a close look at what we mean when we talk about 3D. We all live in a three-dimensional world, though our screens and human interfaces have Traditionally been two-dimensional. With the advent of real-time 3D scanners like the Kinect, this Limitation is somewhat left behind. We can interact with the computer using the three dimensions of physical space (or at least the three we are aware of, if you ask some physicists!). Our screens are still 2D, though, so representing the three dimensions of space on the display

3.3.5.2 KINECT SPACE

You have already accessed Kinect’s depth map in the “Accessing the Depth Map and RGB Image” section and wondered if you can get that map translated back into the real-scale 3D coordinates of the objects. The following sketch does that for you. It starts pretty much like the depth map example but includes a function that shows the real coordinates of the points. First, import the appropriate libraries, declare Simple-OpenNI and KinectOrbit objects, and initialize them in the setup() function, enabling the depth map only.

```
import processing.opengl.*;

import SimpleOpenNI.*;

import kinectOrbit.*;

KinectOrbit myOrbit;

SimpleOpenNI kinect;

void setup()
```

```
{

size(800, 600, OPENGL);

myOrbit = new KinectOrbit(this, 0);

kinect = new SimpleOpenNI(this);

// enable depthMap generation

kinect.enableDepth();

}
```

In the draw() function, include a pushOrbit/popOrbit loop; inside it, add the drawPointCloud() function and draw the Kinect frustum with the method drawCamFrustum(), already included in SimpleOpenNI.requires some nasty math dealing with perspective and projections—which luckily we don’t have to work out by ourselves because somebody did for us already. Processing has full 3D support and two different 3D renderers, P3D and OpenGL. We are going to focus on OpenGL because it is the more powerful of the two.

For drawing Puppet we are using SVG because, they provide flexibility over various preexisted methods

3.3.6 INITIALIZATION OF PUPPET

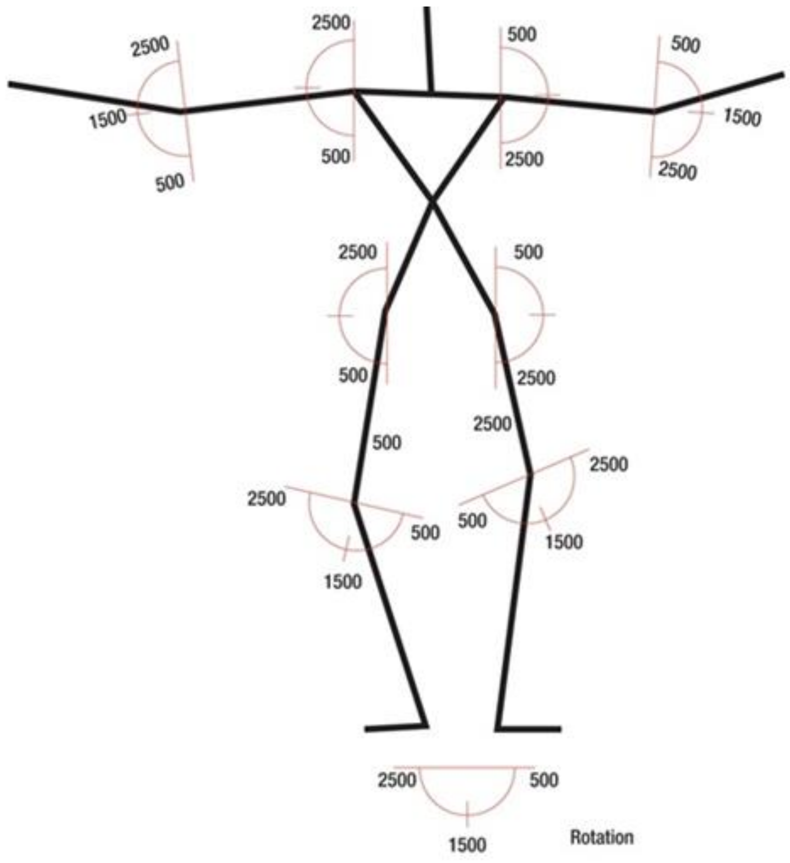


Figure 31. Puppet virtual servo angles

- updateServo(servo3Pin, servoPulse);//lef Shoulder
- updateServo(servo4Pin, servoPulse2);//left Elbow
- updateServo(servo5Pin, servoPulse);//left Hip
- updateServo(servo6Pin, servoPulse2);//left Knee
- updateServo(servo7Pin, servoPulse3); //right Shoulder
- updateServo(servo8Pin, servoPulse2);//rigt Elbow
- updateServo(servo9Pin, servoPulse3);// right Hip
- updateServo(servo10Pin, servoPulse2);//right Knee

- updateServo(servo11Pin, servoPulse2);//move it to the central position

That virtual servo position will be the starting position of the puppet. Once this position has been fixed, you know the orientation of all limbs so you can run simulation.

3.3.6.1 SKELETON TRACKING ON-SCREEN

The first thing we need to get sorted for this project is tracking target user skeleton with Kinect. by tracking the skeleton on-screen, and then we will attach the virtual puppet to simulator . SimpleOpenNI provides a nice interface for skeleton tracking and a couple of callback functions to which you can add your own code. Start by importing and initializing

```
Simple-OpenNI.

import SimpleOpenNI.*;

SimpleOpenNI kinect;

public void setup() {

kinect = new SimpleOpenNI(this);

kinect.setMirror(true);

kinect.enableDepth();

kinect.enableUser(SimpleOpenNI.SKEL_PROFILE_ALL);

          size(kinect.depthWidth(), kinect.depthHeight());

 }
```

Set the mirroring to On so it's easy for you to relate to your onscreen image, and enable the depth image and the User. Pass the parameter SimpleOpenNI.SKEL_PROFILE_ALL to this function to enable all the joints. Set your sketch size to the depth map dimensions.

In the draw() loop, simply update the Kinect data, draw the depth map, and, if Kinect is tracking a skeleton, call the function drawSkeleton() to print your skeleton on screen.

```
public void draw() {

kinect.update();

image(kinect.depthImage(), 0, 0);

if (kinect.isTrackingSkeleton(1)) {

        drawSkeleton(1);

}

}
```

We took this drawSkeleton() function from one of the examples in SimpleOpenNI. It takes an integer as a parameter, which is the user ID (it will be always 1 if you are just tracking a single skeleton). This function runs through all the necessary steps to link the skeleton joints with lines, defining the skeleton that you will see on screen. The function makes use of the public method drawLimb() from the SimpleOpenNI class.

```
void drawSkeleton(int userId) {

pushStyle();

stroke(255,0,0);

strokeWeight(3);
kinect.drawLimb(userId, SimpleOpenNI.SKEL_HEAD, SimpleOpenNI.SKEL_NECK);

kinect.drawLimb(userId, SimpleOpenNI.SKEL_NECK, SimpleOpenNI.SKEL_LEFT_SHOULDER);

kinect.drawLimb(userId,   SimpleOpenNI.SKEL_LEFT_SHOULDER,   SimpleOpenNI.SKEL_LEFT_ELBOW);
kinect.drawLimb(userId, SimpleOpenNI.SKEL_LEFT_ELBOW, SimpleOpenNI.SKEL_LEFT_HAND);

kinect.drawLimb(userId, SimpleOpenNI.SKEL_NECK, SimpleOpenNI.SKEL_RIGHT_SHOULDER);

kinect.drawLimb(userId, SimpleOpenNI.SKEL_RIGHT_SHOULDER, SimpleOpenNI.SKEL_RIGHT_ELBOW);

kinect.drawLimb(userId, SimpleOpenNI.SKEL_RIGHT_ELBOW, SimpleOpenNI.SKEL_RIGHT_HAND);

kinect.drawLimb(userId, SimpleOpenNI.SKEL_LEFT_SHOULDER, SimpleOpenNI.SKEL_TORSO);

kinect.drawLimb(userId, SimpleOpenNI.SKEL_RIGHT_SHOULDER, SimpleOpenNI.SKEL_TORSO);

kinect.drawLimb(userId, SimpleOpenNI.SKEL_TORSO, SimpleOpenNI.SKEL_LEFT_HIP);

kinect.drawLimb(userId, SimpleOpenNI.SKEL_LEFT_HIP, SimpleOpenNI.SKEL_LEFT_KNEE);

kinect.drawLimb(userId, SimpleOpenNI.SKEL_LEFT_KNEE, SimpleOpenNI.SKEL_LEFT_FOOT);

kinect.drawLimb(userId, SimpleOpenNI.SKEL_TORSO, SimpleOpenNI.SKEL_RIGHT_HIP);

kinect.drawLimb(userId, SimpleOpenNI.SKEL_RIGHT_HIP, SimpleOpenNI.SKEL_RIGHT_KNEE);

kinect.drawLimb(userId, SimpleOpenNI.SKEL_RIGHT_KNEE, SimpleOpenNI.SKEL_RIGHT_FOOT);

popStyle();

}
```

3.3.6.2 SIMPLE-OPENNI EVENTS

Use the SimpleOpenNI callback functions to trigger the pose detection and skeleton tracking capabilities. The function onNewUser() is called when a user is detected. You know the user is there, but you haven’t calibrated their skeleton yet, so you start the pose detection.

```
public void onNewUser(int userId) {

println("onNewUser - userId: " + userId);

if (kinect.isTrackingSkeleton(1))   return;

println("  start pose detection");

kinect.startPoseDetection("Psi", userId);

}
```

When the user is lost, you simply print it on the console.

```
public void onLostUser(int userId) {

println("onLostUser - userId: " + userId);

}
```

When you detect a pose, you can stop detecting poses (you already have one!) and request a skeleton calibration to NITE.

```
public void onStartPose(String pose, int userId) {

println("onStartPose - userId: " + userId + ", pose: " + pose); println(" stop pose detection");

kinect.stopPoseDetection(userId);

kinect.requestCalibrationSkeleton(userId, true);

}
```

You also display messages when the pose is finished and when the calibration has started.

```
public void onEndPose(String pose, int userId) {

println("onEndPose - userId: " + userId + ", pose: " + pose);}
```

```
public void onStartCalibration(int userId) {

println("onStartCalibration - userId: " + userId);  }
```

If the calibration is finished and it was successful, you start tracking the skeleton. If it wasn't, you restart the pose detection routine.

```
public void onEndCalibration(int userId, boolean successfull) {

println("onEndCalibration - userId: " + userId + ", successfull: " + successfull); if (successfull) {

println("   User calibrated !!!");

kinect.startTrackingSkeleton(userId);

}

else {

println("   Failed to calibrate user !!!");

            println("   Start pose detection");

            kinect.startPoseDetection("Psi", userId);

 }

}
```

3.3.7 ANGLE CALCULATION

With the previous program, I can track subject skeleton and display it on screen. That is the base of my puppet control. Now to implement a routine that translates poses to the angles of eight servos that will ultimately drive the puppet.

The eight servos are placed on the shoulders, elbows, hips, and knees of your puppet. If we can find out the current bending angles of your joints, you can map those angles to the servos and thus make the puppet move like you. You implement this with the cunning use of geometry.

Taking the previous code as a base, you include a series of routines that will use the skeleton to calculate all the joints' angles. First, add some PVectors at the beginning of the sketch (outside of any function) to store the position of all your joints.

```
// Left Arm Vectors

PVector lHand = new PVector();

PVector lElbow = new PVector();

PVector lShoulder = new PVector();

// Left Leg Vectors

PVector lFoot = new PVector();

PVector lKnee = new PVector();

PVector lHip = new PVector();

// Right Arm Vectors

PVector rHand = new PVector();

PVector rElbow = new PVector();

PVector rShoulder = new PVector();

// Right Leg Vectors

PVector rFoot = new PVector();

PVector rKnee = new PVector();

PVector rHip = new PVector();
```

You also need a PVector array to contain the angles of your joints. float[] angles = new float[9]; You’re interested in nine angles, one for each virtual servo on your puppet.

- angles [0] Rotation of the body
- angles [1] Left elbow
- angles [2] Left shoulder
- angles [3] Left knee
- angles [4] Left Hip
- angles [5] Right elbow
- angles [6] Right shoulder
- angles [7] Right knee
- angles [8] Right Hip

You will be wrapping all these calculations into a new function called updateAngles() that you call from your main draw() function only in the case you are tracking a skeleton. Add this function to the if statement at the end of the draw() function, and then print out the values of the angles array.

```
if (kinect.isTrackingSkeleton(1)) {

updateAngles();

println(angles);

drawSkeleton(1);  }
```

You need a function that calculates the angle described by the lines joining three points, so implement it next; call it the angle() function.

```
float angle(PVector a, PVector b, PVector c) { float angle01 = atan2(a.y - b.y, a.x - b.x);

float angle02 = atan2(b.y - c.y, b.x - c.x);

float ang = angle02 - angle01;

return ang;

}
```

The function updateAngles() performs the calculation of all these angles and stores them into the previously defined angles[] array. The first step is storing each joint position in one PVector. The

method getJointPosition() helps you with this process. It takes three parameters: the first one is the ID of the skeleton, the second one is the joint you want to extract the coordinates from, and the third one is the PVector that is set to those coordinates.

```
void updateAngles() {

// Left Arm

kinect.getJointPositionSkeleton(1, SimpleOpenNI.SKEL_LEFT_HAND, lHand);

kinect.getJointPositionSkeleton(1, SimpleOpenNI.SKEL_LEFT_ELBOW, lElbow);

kinect.getJointPositionSkeleton(1, SimpleOpenNI.SKEL_LEFT_SHOULDER, lShoulder); // Left Leg

kinect.getJointPositionSkeleton(1, SimpleOpenNI.SKEL_LEFT_FOOT, lFoot); kinect.getJointPositionSkeleton(1,
SimpleOpenNI.SKEL_LEFT_KNEE,lKnee);

kinect.getJointPositionSkeleton(1,SimpleOpenNI.SKEL_LEFT_HIP, lHip); // Right Arm
kinect.getJointPositionSkeleton(1,SimpleOpenNI.SKEL_RIGHT_HAND,rHand);
kinect.getJointPositionSkeleton(1,SimpleOpenNI.SKEL_RIGHT_ELBOW,rElbow);
kinect.getJointPositionSkeleton(1,SimpleOpenNI.SKEL_RIGHT_SHOULDER, rShoulder); // Right Leg
kinect.getJointPositionSkeleton(1,SimpleOpenNI.SKEL_RIGHT_FOOT,rFoot);
kinect.getJointPositionSkeleton(1,SimpleOpenNI.SKEL_RIGHT_KNEE,rKnee);
kinect.getJointPositionSkeleton(1, SimpleOpenNI.SKEL_RIGHT_HIP, rHip);
```

Now you have all your joints nicely stored in the PVectors you defined for that purpose. These are three-dimensional vectors of real-world coordinates. You transform them to projective coordinates so you get the second angles (remember your puppet is pretty flat!), but first you need to extract your body rotation angle (see Figure 6-25), which is the only rotation of the plane that you need.

```
angles[0] = atan2(PVector.sub(rShoulder, lShoulder).z,
PVector.sub(rShoulder, lShoulder).x);
```

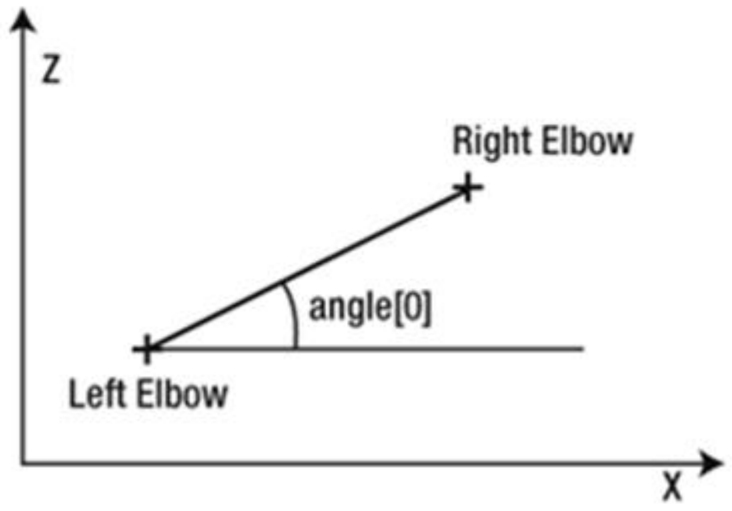


Figure 32. Rotation of the body

After this angle has been calculated, you can transform all the joints to projective coordinates (on screen coordinates).

```
kinect.convertRealWorldToProjective(rFoot, rFoot);

kinect.convertRealWorldToProjective(rKnee, rKnee);

kinect.convertRealWorldToProjective(rHip, rHip);

kinect.convertRealWorldToProjective(lFoot, lFoot);

kinect.convertRealWorldToProjective(lKnee, lKnee);

 kinect.convertRealWorldToProjective(lHip, lHip);

 kinect.convertRealWorldToProjective(lHand, lHand);

 kinect.convertRealWorldToProjective(lElbow, lElbow);

kinect.convertRealWorldToProjective(lShoulder, lShoulder);
```

```
 kinect.convertRealWorldToProjective(rHand, rHand);

kinect.convertRealWorldToProjective(rElbow, rElbow);

kinect.convertRealWorldToProjective(rShoulder, rShoulder);

//And finally, you use the angle function you implemented previously to compute all the //necessary angles for the control of the puppet.

// Left-Side Angles

angles[1] = angle(lShoulder, lElbow, lHand);

angles[2] = angle(rShoulder, lShoulder, lElbow); angles[3] = angle(lHip, lKnee, lFoot);

angles[4] = angle(new PVector(lHip.x, 0), lHip, lKnee); // Right-Side Angles

angles[5] = angle(rHand, rElbow, rShoulder);

angles[6] = angle(rElbow, rShoulder, lShoulder ); angles[7] = angle(rFoot, rKnee, rHip);

angles[8] = angle(rKnee, rHip, new PVector(rHip.x, 0));

}
```

Now I have all the data to control my puppet! By running the angle calculation sketch and start the skeleton tracking, I get the values of targets joints' angles appear on the console. I could now connect the serial cable to computer if I want an add on to this project. I want to be able to control the puppet remotely through the Internet, and to do this I need to make network infrastructure to communicate with our hosted server to translate client data and make a simulation model.

3.3.8 COMMUNICATING WITHIN A LOCAL NETWORK

To build up a communication connection between different nodes, here I am using unreserved network port.like 12345

In server skech, I have to add,

s = new Server(this, 12345);

And in the client sketch,

```
client. c = new Client(this, "127.0.0.1", 12345);
```

The client is trying to connect to the computer with IP address 127.0.0.1 through port 12345. Well, this address happens to be the localhost address of the computer. It worked well as long as we had both sketches running on the same computer, but if you want to make them work on different machines, we can provide the client with the IP address of the server. we can find the internal IP address of your computer easily. On a linux, open Network Preferences; it's on the top right. like "AirPort is connected to xxxxxx and has the IP address 192.168.0.101." On a PC, go to the Start menu. In "Search programs and files" write cmd and press Enter. You have opened a console. Now type ipconfig in the console and press Enter again. Many text lines will appear, one of which says something like "IPv4 Addres..:192.168.0.103". This is server IP address. If you go now to your client sketch and substitute the 127.0.0.1 with the IP address of the machine in which you are running the server (the other computer) and run both sketches (remember to run the server first), it should work smoothly.

```
c = new Client(this, "192.168.0.101", 12345);
```

Now you have two Processing sketches running on different computers talking to each other. But the ultimate goal is to be able to communicate over the Internet. Can this code make that happen? Can I go to my friend's house with the client sketch, run it, and be connected to my machine at home? Well, unfortunately not yet. The IP addresses we've been using are internal IP addresses.

3.3.8.1 SERVER APPLET: SENDING THE ANGLES OVER A NETWORK

So now that we are done with network and installation but to overcome various lagging issues and proxy configuration. I need to set port forwarding to make route path in network. Now we can communicate across networks, and check the functionality for angle calculation sketch so we can send the angles over the network for another program to retrieve them from a remote location. Now I am importing Network library and declare a Server object. afterword's initialize all the PVectors and the angles array as per previous code.

```
import processing.net.*;

import SimpleOpenNI.*;

SimpleOpenNI kinect;

Server s; // Server Object

// Left Arm Vectors

PVector lHand = new PVector();

PVector lElbow = new PVector();

PVector lShoulder = new PVector();

// Left Leg Vectors

PVector lFoot = new PVector();

PVector lKnee = new PVector();

PVector lHip = new PVector();

// Right Arm Vectors

PVector rHand = new PVector();

PVector rElbow = new PVector();

PVector rShoulder = new PVector();

// Right Leg Vectors

PVector rFoot = new PVector();

PVector rKnee = new PVector();

PVector rHip = new PVector();

// Articulation Angles

float[] angles = new float[9];
```

In the setup() function, we initialize the server object, passing the port you are communicating through as a parameter.

```
public void setup() {

kinect = new SimpleOpenNI(this);

kinect.setMirror(true);

kinect.enableDepth();

kinect.enableUser(SimpleOpenNI.SKEL_PROFILE_ALL);

frameRate(10);

size(kinect.depthWidth(), kinect.depthHeight());

s = new Server(this, 12345); // Start a simple server on a port

}
```

And finally, at the end of the draw() loop and only in case we are tracking a skeleton, write a long string of characters to server port. The string is composed of all the joint angles separated by white spaces, so we can transcode it on receiver side .

```
public void draw() {

kinect.update();    // update the Kinect

image(kinect.depthImage(), 0, 0);     // draw depthImageMap

if (kinect.isTrackingSkeleton(1)) {

            updateAngles();

drawSkeleton(1); // Write the angles to the socket

s.write(angles[0] + " " + angles[1] + " " + angles[2] + " "

            + angles[3] + " " + angles[4] + " " + angles[5] + " "

            + angles[6] + " " + angles[7] + " " + angles[8] + "\n");

 } }
```

CHAPTER 4

TEST & RESULT

FIRST RUN

4.1 CLIENT APPLET: CONTROLLING THE PUPPET

The server applet is finished and ready to start sending data over the Internet. To perform recognition we need a client sketch to receive the angle data and send it to simulated environment. Arduino.dll is used to string translate and decode into servo positions that will become movements of puppet. We’re going to implement a virtual puppet resembling, so check that everything is working properly before sending data to the virtual servos.

```
import processing.net.*;

import processing.serial.*;

Serial myPort;

boolean serial = true;

PFont font;
```

Now we need to declare a client object called c, declare a String to contain the message coming from the server applet, and a data[] array to store the angle values once you have extracted them from the incoming string. Afterwards we are going to use a PShape to draw a version of your physical puppet on screen so I can generate test environment for all the movements virtually first.

```
Client c;

String input;
```

```
float data[] = new float[9];

PShape s;
```

In the setup() function, initialize the client object. Remember to replace the IP address with the external IP of the server computer if you are communicating over the Internet or the internal IP if you are communicating within the same local network. we need to add the file Android.svg to your sketch folder so it can be loaded into the PShapes.

```
void setup()

{

size(640, 700);

background(255);

stroke(0);

frameRate(10);

// Connect to the server's IP address and port

c = new Client(this, "127.0.0.1", 12345); // Replace with your server's IP and port font = loadFont("SansSerif-
14.vlw");

textFont(font);

textAlign(CENTER);

s = loadShape("Android.svg");

shapeMode(CENTER);

smooth();

if (serial) {

String portName = Serial.list()[0]; // This gets the first port on your computer. myPort = new Serial(this, portName,
9600);

}
}
```

Within the draw() loop, check for incoming data from the server; in a positive case, read is as a string. Then trim off the newline character and split the resulting string into as many floats as there are sub-strings separated by white spaces in your incoming message. Store these floats (the angles of your limbs sent from the server sketch) into the data[] array.

Run a little validation test, making sure that all the values acquired fall within the acceptable range of -PI/2 to PI/2.

```
void draw()

{

background(0);

// Receive data from server

if (c.available() > 0) {

            input = c.readString();

input = input.substring(0, input.indexOf("\n")); // Only up to the newline data = float(split(input, ' ')); // Split values into an array

for (int i = 0 ; i < data.length; i++) {

if(data[i] >   PI/2) { data[i] = PI/2; }

if(data[i] < -PI/2) { data[i] = PI/2; }

}

}
```

Once the data is nicely stored in an array, draw your puppet on screen and add the limbs in the desired position. You first draw the shape on screen in a specific position, which you have figured out by testing. Then use the function drawLimb() to draw the robot's limbs using the angles in the data array as parameters.

```
shape(s, 300, 100, 400, 400);

drawLimb(150, 210, PI, data[2], data[1], 50); drawLimb(477, 210, 0, -data[6], -data[5], 50); drawLimb(228, 385, PI/2, data[4], data[3], 60); drawLimb(405, 385, PI/2, -data[8], -data[7], 60);
```

Then draw an array of circles showing the incoming angles in an abstract way so we can debug inconsistencies and check that client are receiving reasonable values.

```
stroke(200);

fill(200);

for (int i = 0; i < data.length; i++) {

            pushMatrix();

translate(50+i*65, height/1.2);

noFill();

ellipse(0, 0, 60, 60);

text("Servo " + i + "\n" + round(degrees(data[i])), 0, 55); rotate(data[i]);

line(0, 0, 30, 0);

popMatrix();

}
```

The drawLimb() function is designed to draw an arm or leg on screen starting at the position specified by the parameters x and y. The angle0 variable specifies the angle that you have to add to the servo angle to accurately display its movement on screen. angle1 and angle2 determine the servo angles of the first and second joint angles on the limb. Limbsize is used as the length of the limb.

```
void drawLimb(int x, int y, float angle0, float angle1, float angle2, float limbSize) {

        pushStyle();

strokeCap(ROUND);

strokeWeight(62);

stroke(134, 189, 66);

pushMatrix();

translate(x, y);

rotate(angle0);

rotate(angle1);

line(0, 0, limbSize, 0);

translate(limbSize, 0);

rotate(angle2);

line(0, 0, limbSize, 0);

popMatrix();

popStyle();

}
```

Now we can transfer our implemented part towards testing system. Where we simulate various gesture and test the output with respect to bandwidth consumed in various ideologies.

4.2 USER 3D

In this figure I am showing various capabilities of Kinect based system. Here I am tracing various position in 3D space. The complete rending and pixel positioning is done by data structure called point cloud.

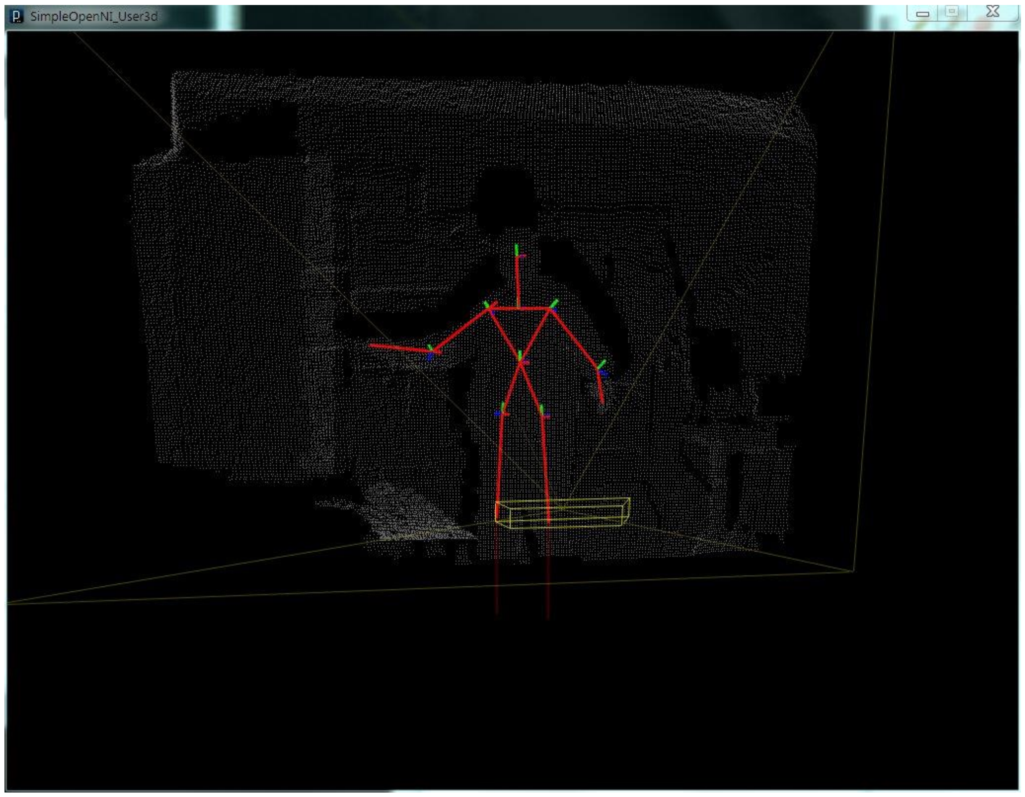


Figure 32: Point cloud as 3D space sklaton skech (Front View)

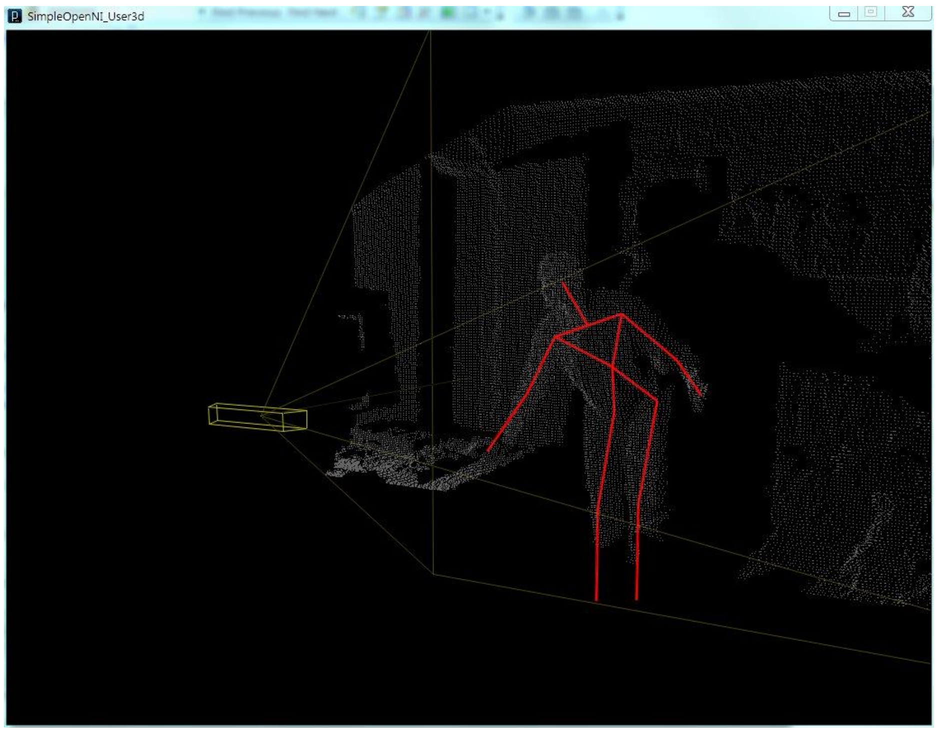


Figure 33: Point Cloud Formation (Side view)

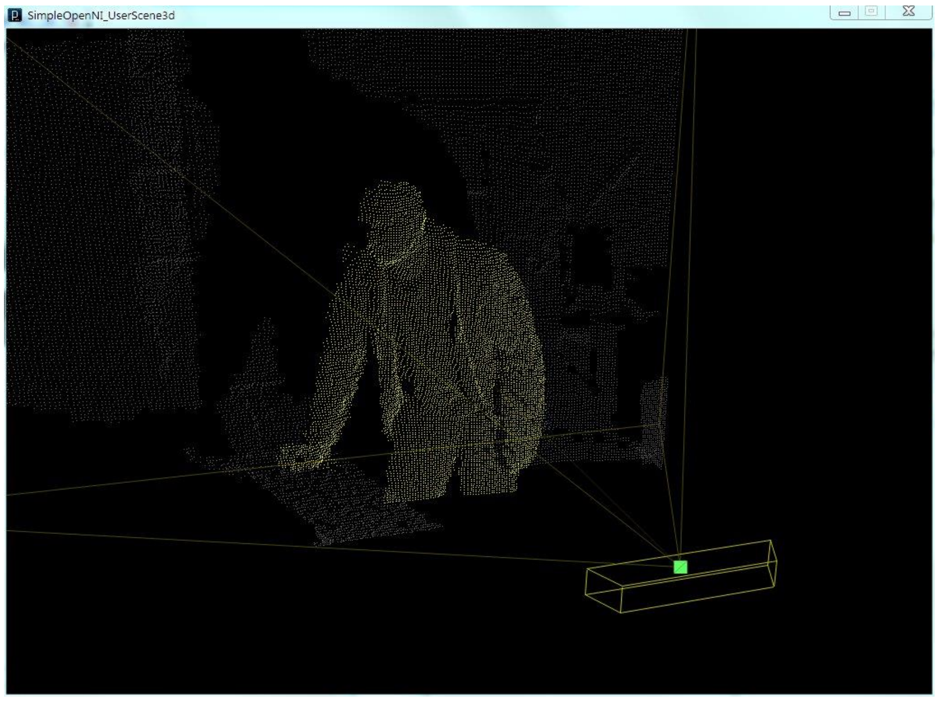


Figure 33: Point Cloud Formation (Side view)

4.3 DEPTH VIEW

In this sample I am showing various aspact of depth imagnary as a virtual envirnoment

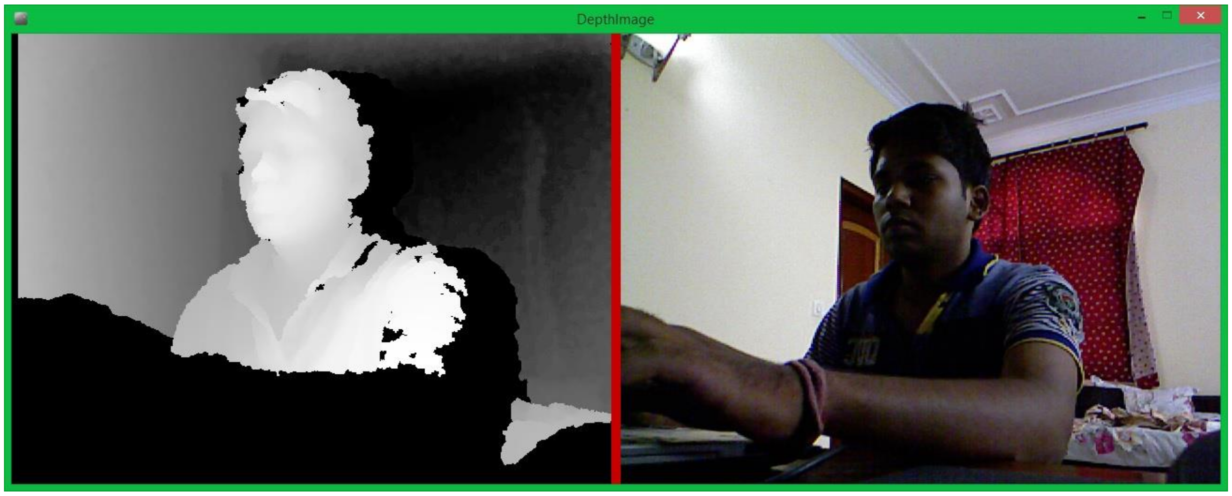


Figure 34: Depth Image in Florescent Light

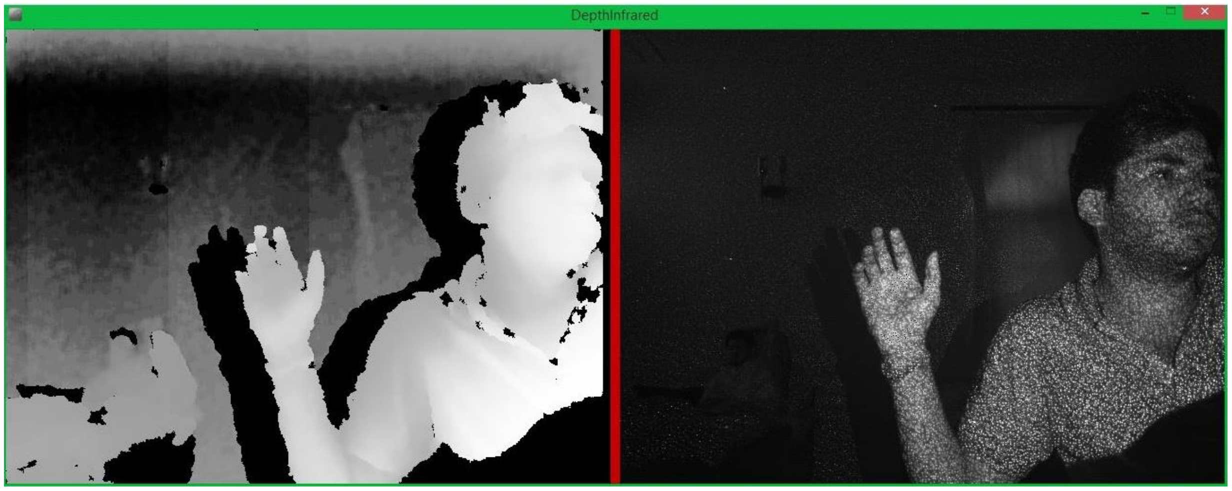


Figure 35: Depth Image in no Light (Infrared Image)

4.4 THESIS DEMONSTRATION AS A SIMULATOR (TELEPORTED CONTROL SYSTEM)

It’s a default overlay. If the server is down or not sending any signal.

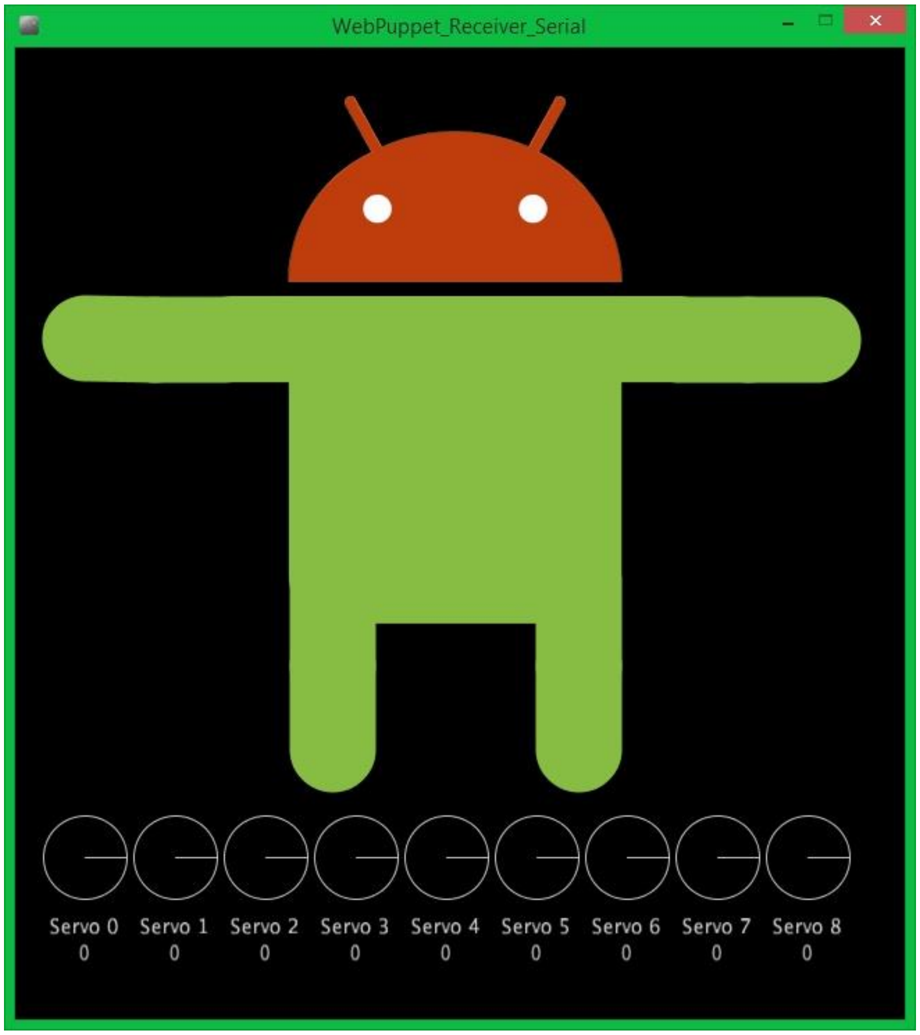
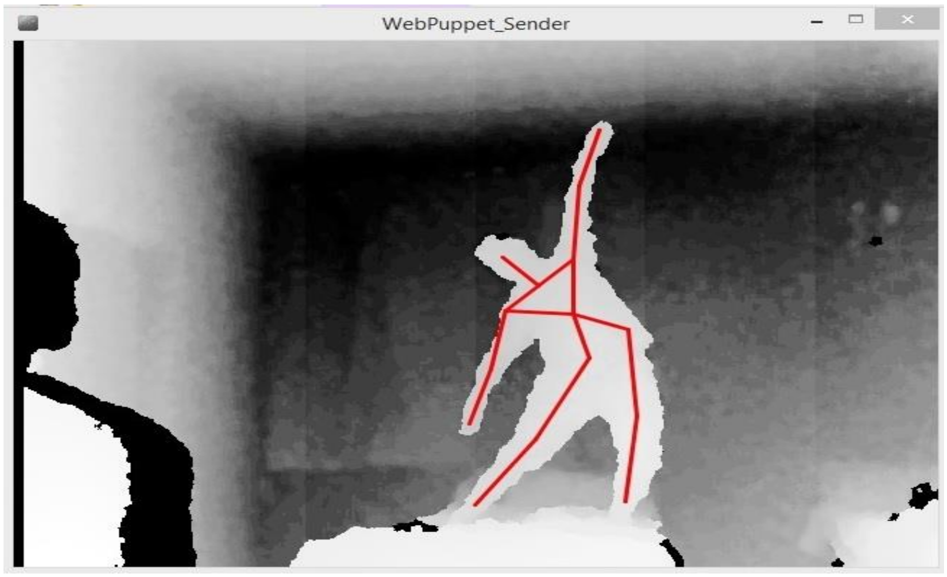


Figure 36: ideal puppet without I/O



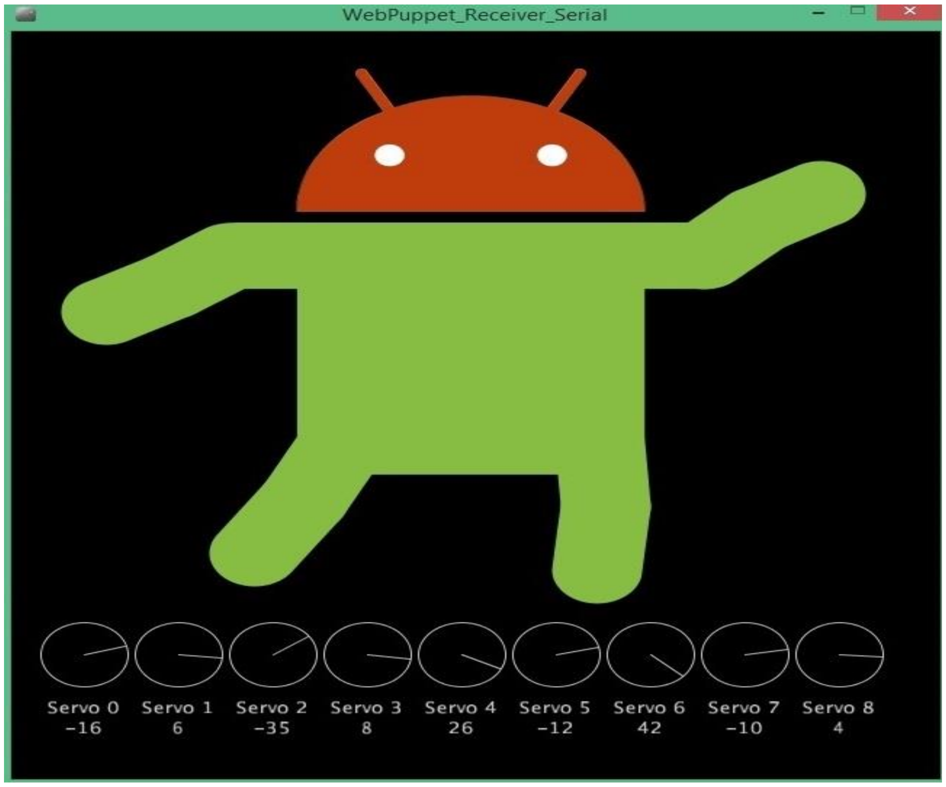
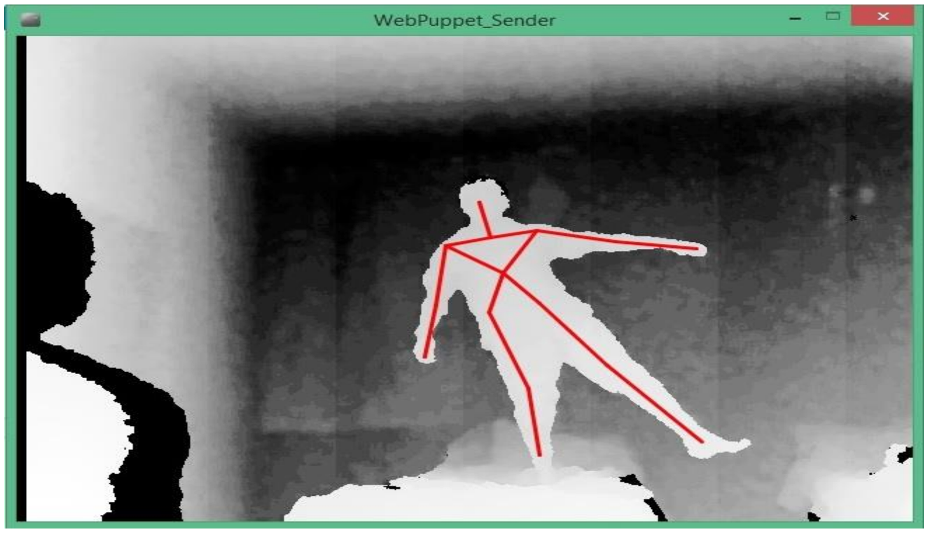


Figure 36: Test 1



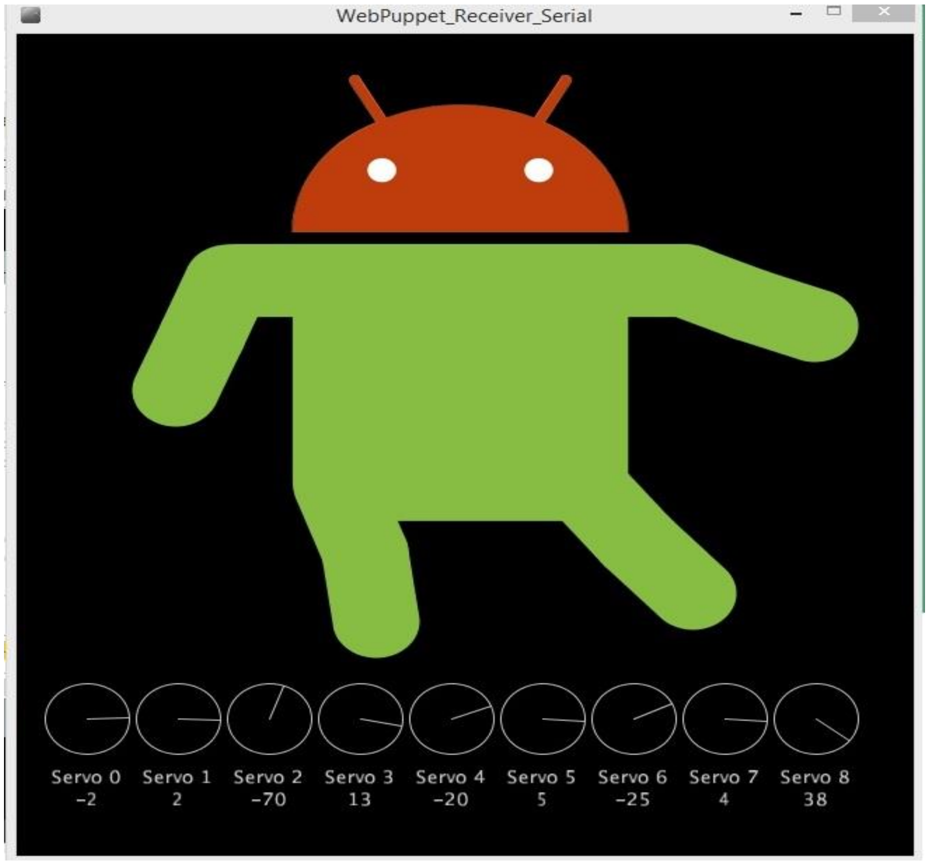
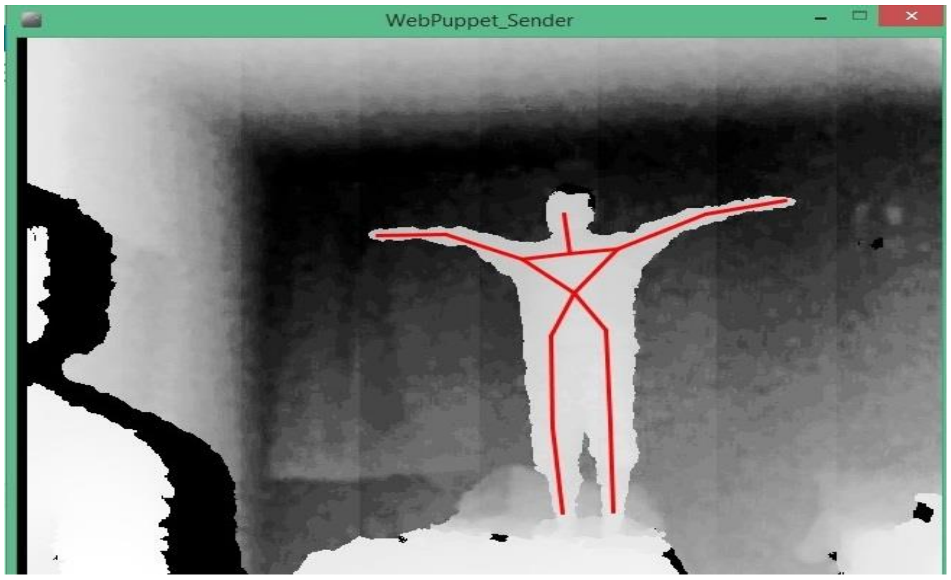


Figure 37: Test 2



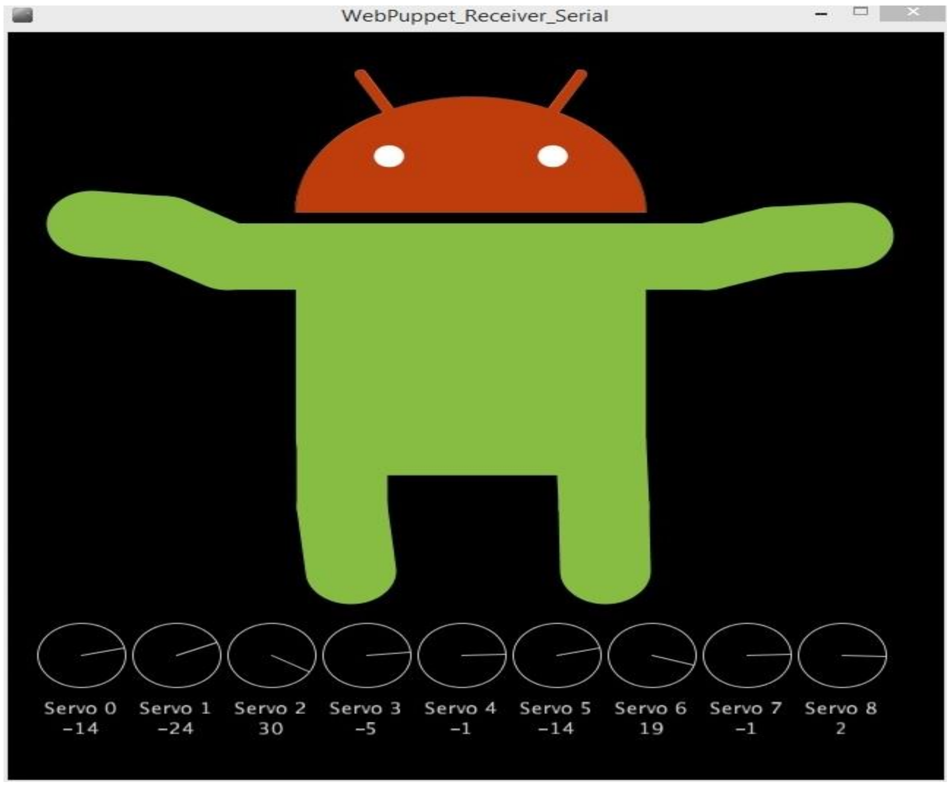
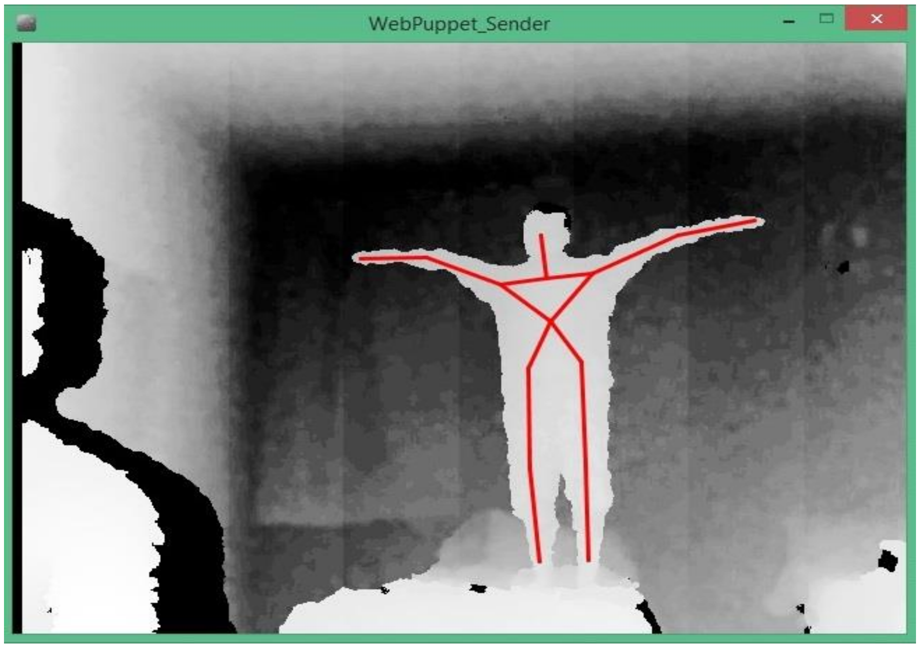


Figure 37: Test 3



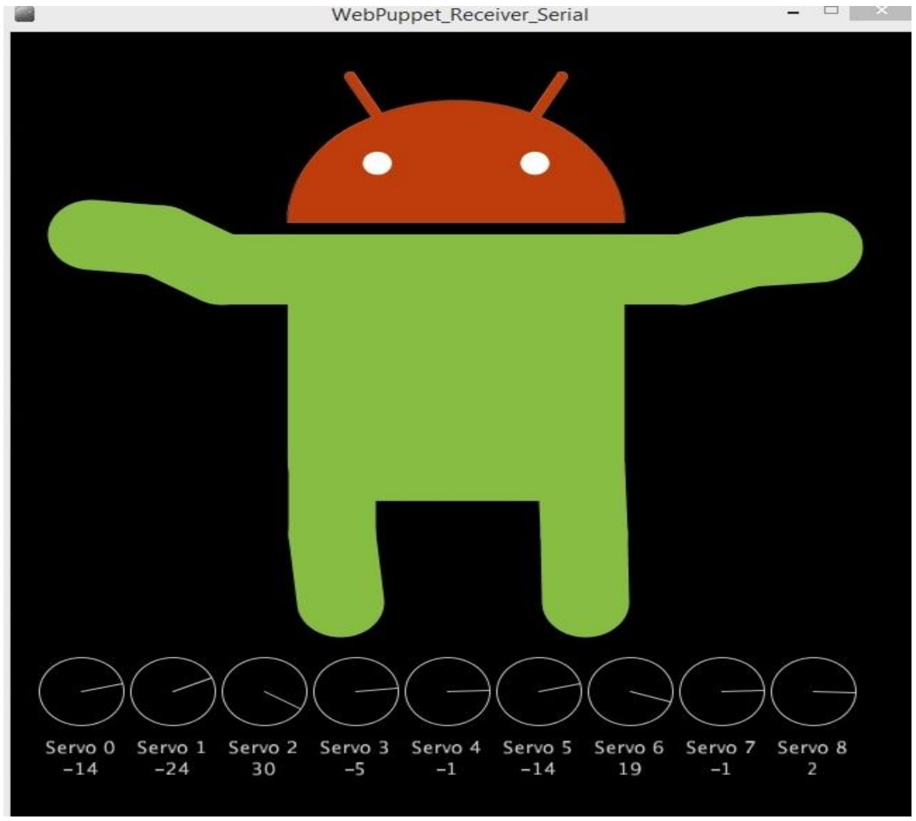
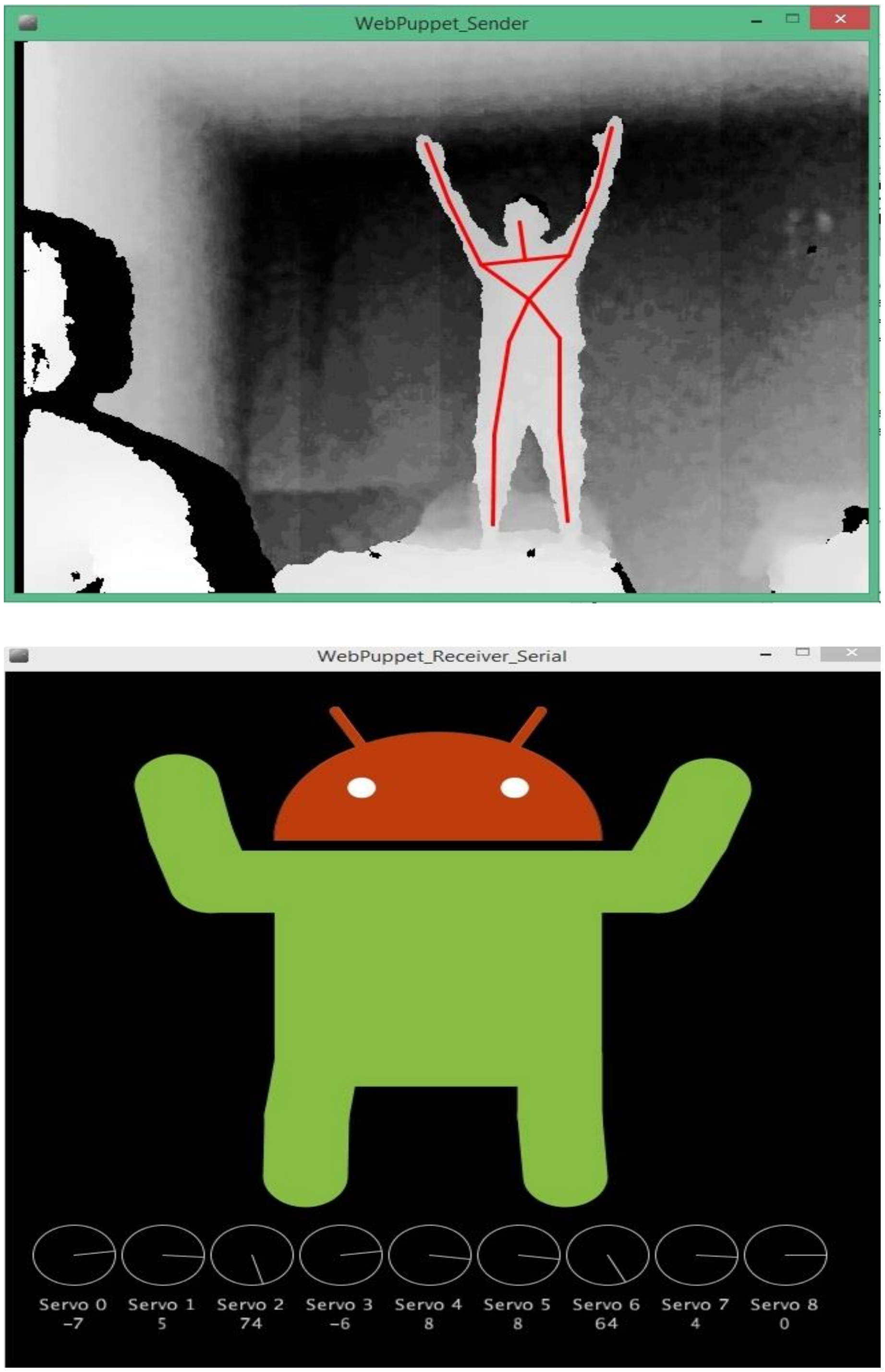


Figure 38: Test 4



```
Servo 0  Servo 1  Servo 2  Servo 3  Servo 4  Servo 5  Servo 6  Servo 7  Servo 8
  -7        5       74       -6        8        8       64        4        0
```

Figure 39: Test 5

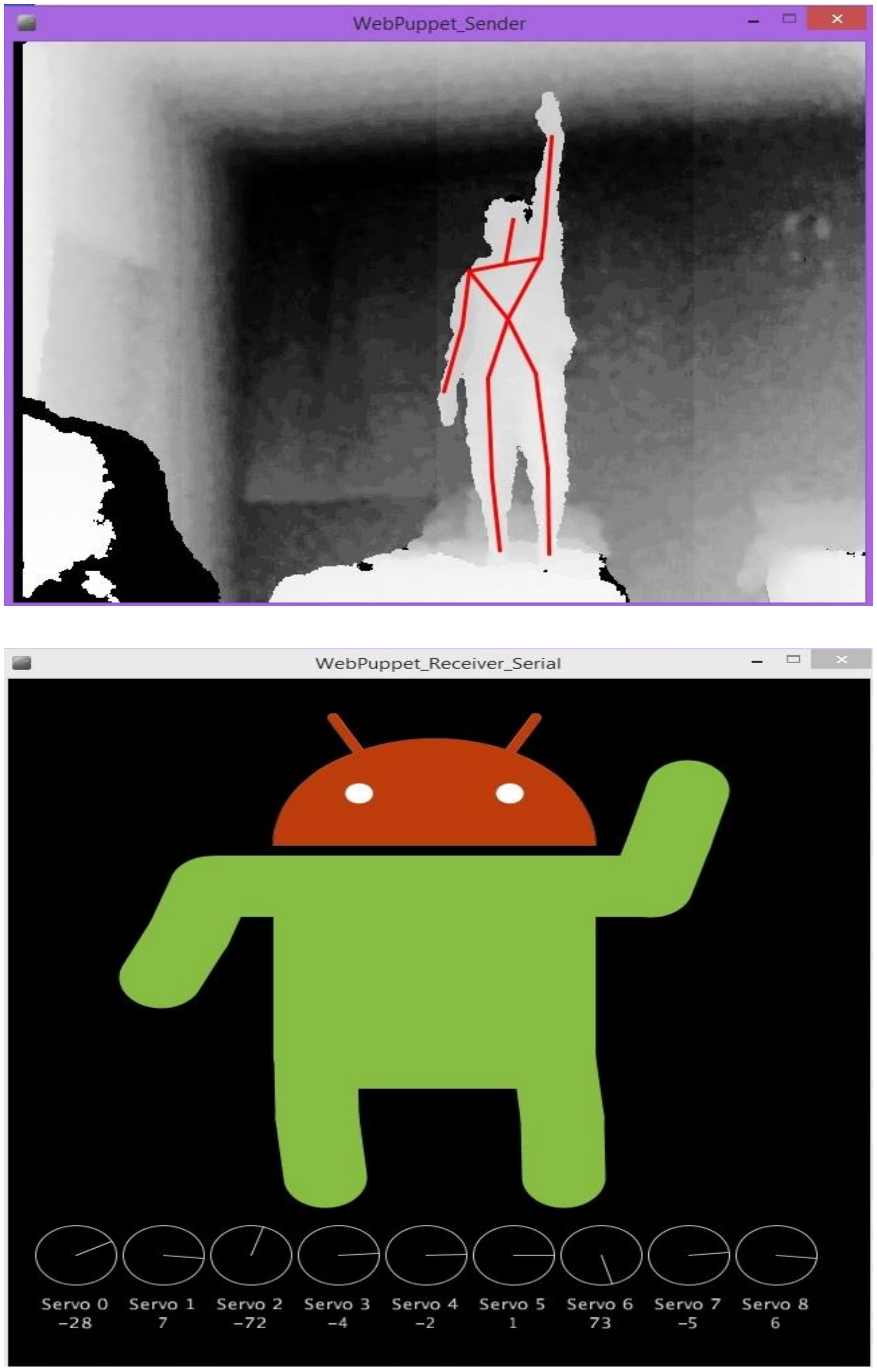
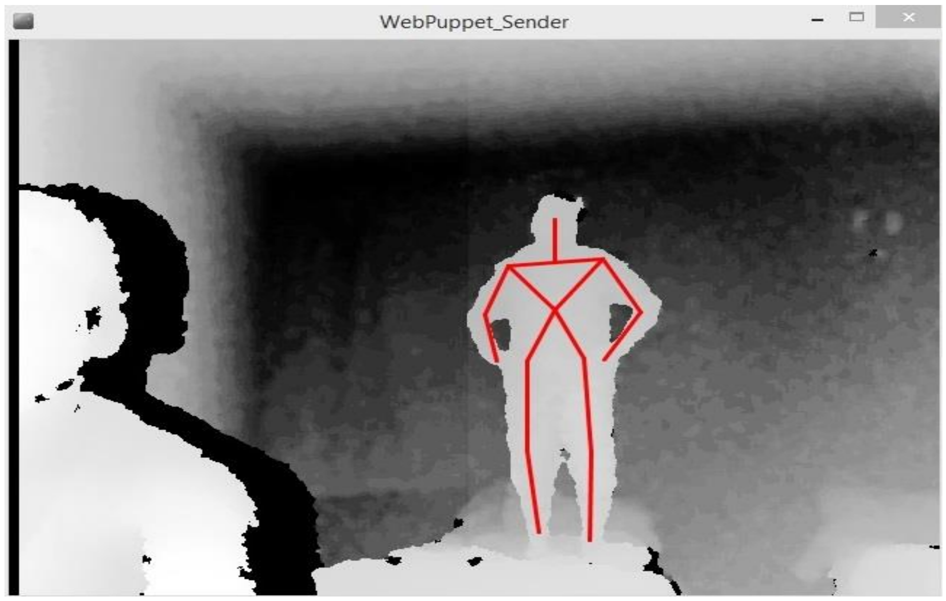
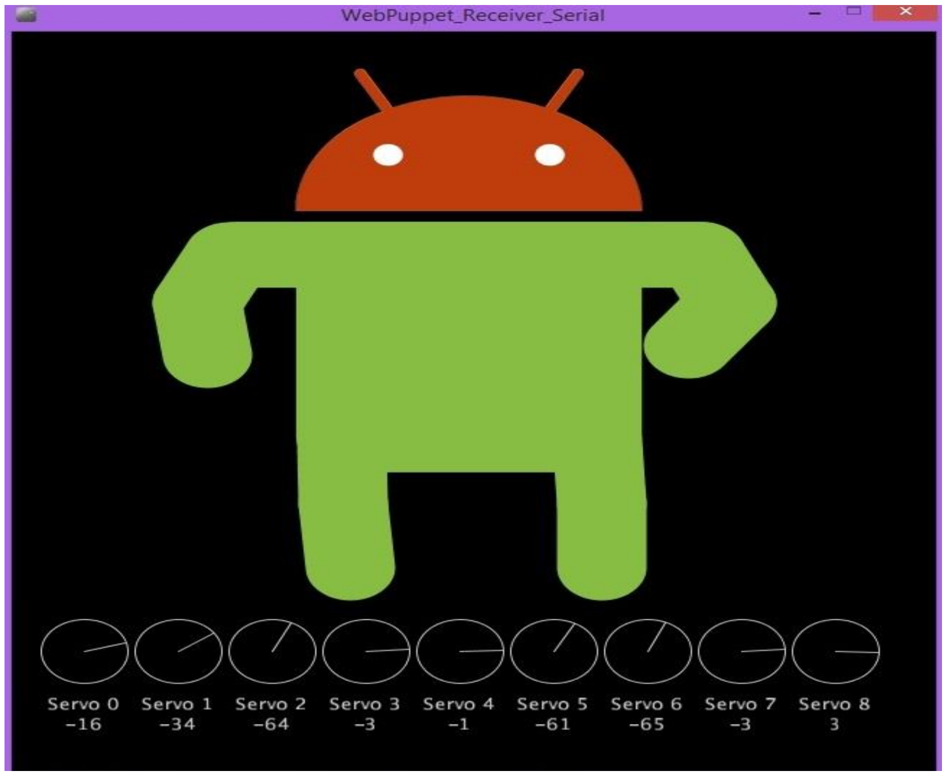


Figure 40: Test 6





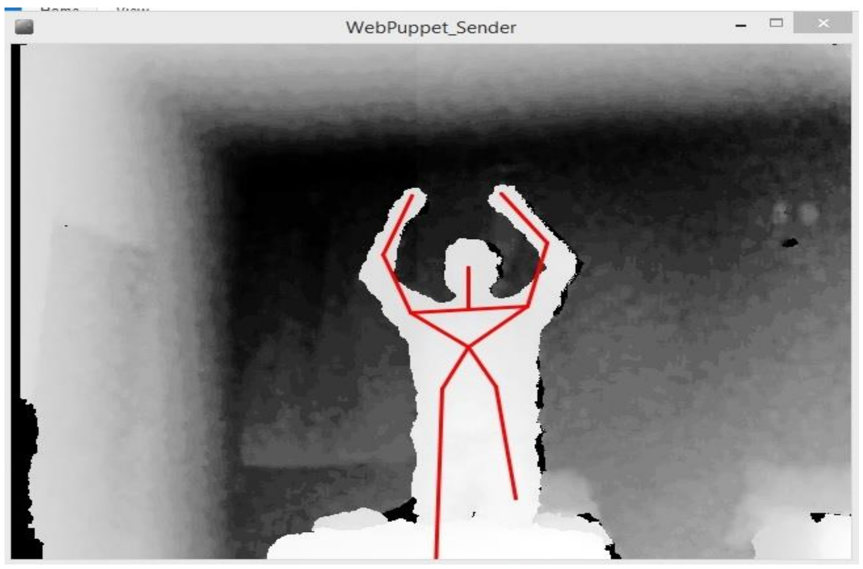
Servo 0 -16, Servo 1 -34, Servo 2 -64, Servo 3 -3, Servo 4 -1, Servo 5 -61, Servo 6 -65, Servo 7 -3, Servo 8 3

Figure 41: Test 7

Our all test case working fine if gesture is completely recognize by Kinect sensor

4.5 FAIL CASE

In this if we are taking incomplete gesture as an input, that's why there is a loss of information and we can't trace complete point cloud and therefore distortion is there, as shown in figure



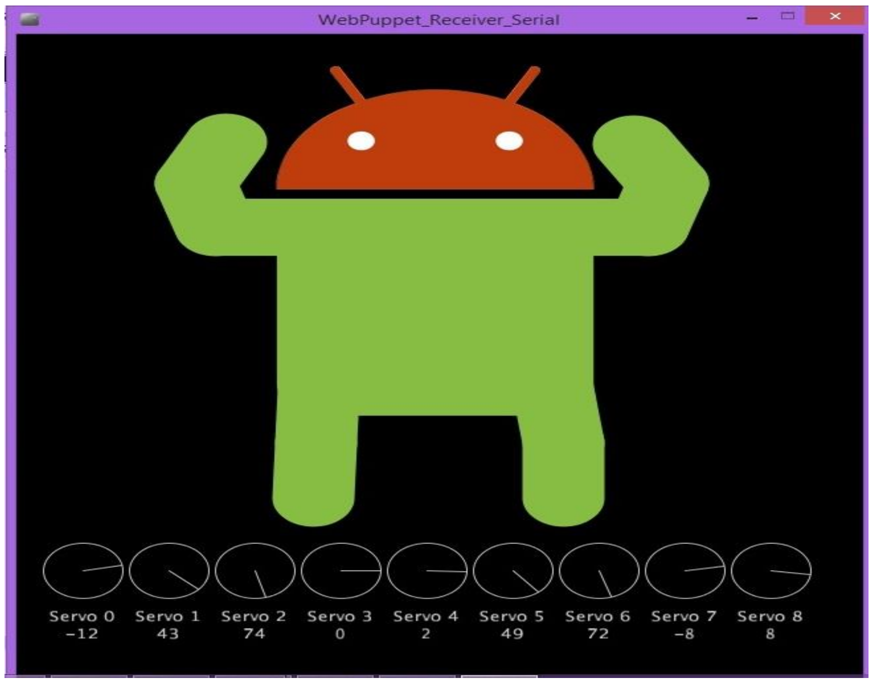
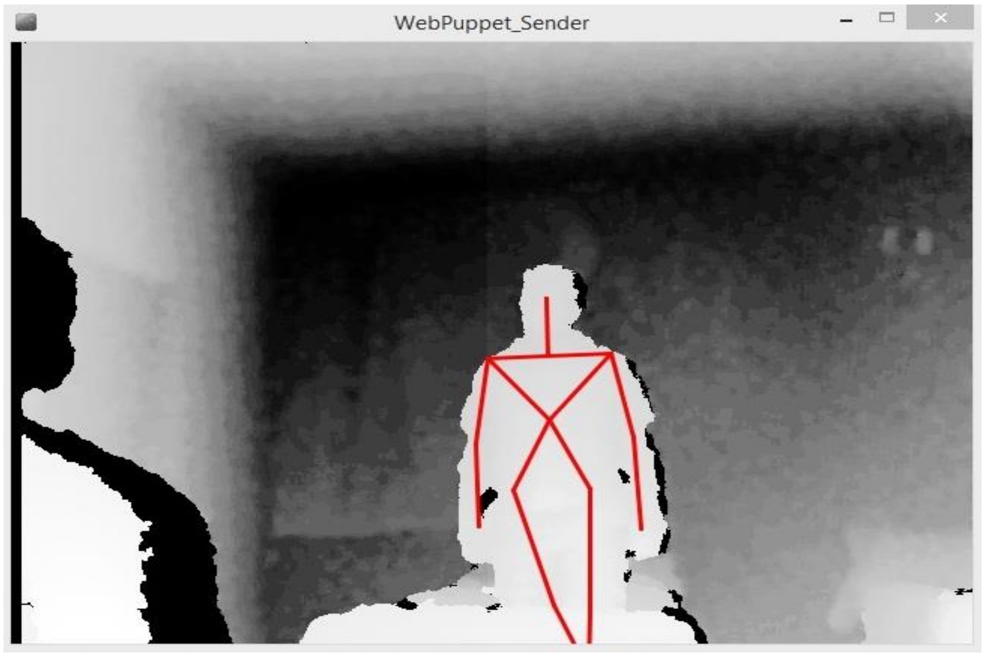


Figure 42: Error case1



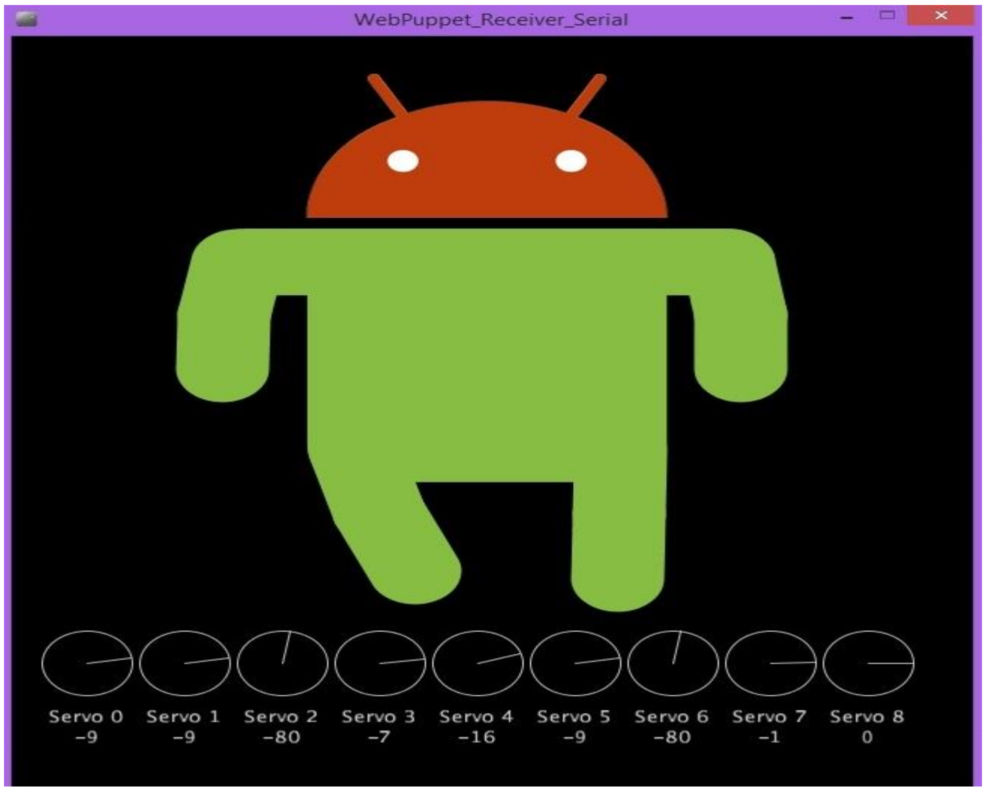


Figure 43: Error case 2

Inefficient lighting may cause problem but the sensor I am using here is night vision enable so, this problem not happen here but if care less operation is done like network bandwidth bottleneck or client port problem, these problem can directly reflected in our working modal, as I am implementing here is a virtual puppet we don’t have much problem but on further extension in hardware these small factor can cause serious damage to real servo and deal fluctuation in our puppet.

CHAPTER 5

CONCLUSION & FUTURE SCOPE

The following discussion will be divided into different parts. In the first part we will discuss the results implication on the usability; here the focus is on the findings from the evaluation in relation to the parts constituting to the usability. In the second part we will try to understand the results in relation to the target group. In the third part we will discuss the results implications for interaction techniques; here the focus is on the generalization of the results in relation to the theory and a discussion of which design elements constitute to the success of a technique

Test

1. **Memorability**
2. **Efficiency**
3. **Errors**
4. **Satisfaction & Comfort**

Result

5.1 TARGET GROUP AND USABILITY

In order to get closer to an answer to the problem statement of this thesis a discussion of the results importance within the usability is needed. Having described the obtained results in relation to the different parameters constituting to the usability, the important question of how to weight these parameters remains. It is also important to notice that the measured usability results are relative to the user group partaking in the evaluation.

The intention with the study was to evaluate the usability on novice users, which we defined as users without special experience in the use of 3D interaction techniques, especially without having used similar systems as the one tested on. Although it can be argued that the demographical data obtained shows that the user group had a fairly high experience in computer technology and alternative interaction devices (like the Nintendo Wii / Playstation Move), we still assume that the users partaking had no prior experience in the devised ITs or similar methods of interacting with screen content. This assumption is supported by the demographics showing that most participants had no prior experience in 3D tracking devices such as the Kinect.

5.2 GENERALIZATION AND DESIGN GUIDELINES

This last part of the discussion tries to explain which design factors are important for usable selection techniques using markerless motion tracking. In this part we try to gather all the knowledge learned throughout the project, including the theoretical research and the practical knowledge gained while working with the Kinect camera and the NITE tracking framework for creating the implementation. This knowledge forms the bases for a "best guess" heuristics concerning the parts that are especially important to include in the design of selection techniques using the described hardware; in the end of this section this will be finalized as a list of design guidelines. Though the discussion presented here is without direct proof, we think that the results indicate and the knowledge gained throughout the project justifies the elements presented in this part.

Having concluded that the mouse technique showed to be the overall most usable technique for the described target group, the important question to ask at this point is which of the design elements utilized in this technique constitute to its success? The goal with this techniques design was to resemble the behavior of a regular mouse pointer, as found in most operating systems. This design choice builds on the users knowledge, it allow users to form a mental model of a well-known form of interacting with computers. This choice constitutes to both the learnability and the memorability of the technique and it might even satisfy users by giving them what they already know.

In general we think that the number of actions required when using a selection technique, should be kept low when learnability is important. Our point of view is that many aspects concerning the memorability, can be explained by the general way the selection techniques are operated, by using body movements to control the techniques. In this way the techniques involve user's sensory memory, building on users motor skills and by this actively providing memorization through bodily action. It is hard to determine which design aspects constitute the satisfactory aspects of a technique. This subjective aspect might be influenced by several reasons. The possibly most valuable aspect to consider for the satisfaction is to know the targeted audience well. We also think that a part can be found in the aesthetics of a technique, the visual appeal and the comfort. On the other hand we believe that utilizing principles, like the ones described in the analysis section of this thesis, can play an important role in the user's satisfaction towards a technique. The magical principles could for example stimulate a user's sense of imagination, by that possibly contributing to the satisfaction. However, the techniques implemented in this project are not fully utilizing the possibilities of these magical principles. It would be interesting to further investigate these principles and to create techniques that encourage users' imagination.

The efficiency and errors are especially influenced by the input methodology. In order to design effective selection techniques for the Kinect and the used tracking framework, it showed to be important to address tracking inaccuracy through the design. One of the design elements important for allowing effective selection even with inaccurate tracking is the use of constrains and mapping. The circle-menu technique maps in this way the users hand position effectively by constraining the input axis. The outcome is that the technique only depends on the hands x and y position relative to a fixed point, which in turn eliminates a large part of the inaccuracy of the tracking. Similar are constrains utilized to restrict the input for the mouse technique. The other factor important is to consider the distance of motion needed to select objects, like described in Fitts' law. A general advice for overcoming tracking inaccuracy and also to lower the error rate is to design for coarse movement. Most results obtained showed a high error rate and lower efficiency in tasks where fine-tuned movement was required. This can largely be avoided by designing both the content (the VE) and the selection techniques to circumvent such movement. Especially the unconstrained techniques like the pointing technique are more prone to such inaccuracy, which contrary to design suggestions in the literature (e.g. (Bowman et al., 2005)) makes them less attractive for selection techniques using markerless motion tracking.

The following list summarizes the usability aspects discussed in a list of design guidelines for selection techniques using markerless motion tracking, like the Kinect and the NITE framework:

- **Learnability**
  - Selection techniques that require least possible actions in order to perform a selection are easy to learn
  - Utilizing the user knowledge (principles and metaphors) is important for the ease of learning
  - Selection techniques that do not require confirmation actions are especially easy to learn

- **Memorability**
  - Using the body as control (controller free input) allows users to easier remember selection techniques compared to mediated techniques (controller based input)

- **Satisfaction**
  - Detailed knowledge of the target group helps improving the satisfaction with selection techniques

- **Efficiency and errors**
  - Constrain the input to overcome tracking inaccuracy
  - Avoid fine movement and design for coarse movement to overcome tracking inaccuracy
  - Avoid timed confirmation methods when efficiency is more important than learnability
  - Utilize Fitts' law in the design of selection techniques
  - Avoid selection errors by constraining confirmation actions
  - Reliable gesture detection is of great importance to avoid errors

- **Comfort**
  - If possible allow users to rest their arms
  - Avoid techniques that require outstretched arms

5.3 CONCLUSION

In the work of this thesis we investigated the possibility of a cable free VE, by using the recently released Kinect as input together with the publicly available NITE tracking framework. The focus was set on interaction techniques allowing selection of screen content. The problem addressed was formulated as:

Which body controlled interaction techniques are best suited for single and multiple selection of screen content, using optical markerless motion capture (Kinect and NITE) and which design elements constitute the success of a technique?

Through the theoretical knowledge gained from the literature we developed a taxonomy describing the important aspects of selection techniques. Together with the principles important for VEs we used the theory to developed five interaction techniques representing the major aspects of our taxonomy: an object touch technique, two pointing techniques, a technique using HUD elements and a gesture based technique. These techniques were refined and implemented based on an iterative development process and finally evaluated in a user study addressing the important aspects of the usability, which are: learnability, memorability, efficiency, errors, satisfaction and comfort. Since these aspects are relative to the intended target group, we assessed and weighted the outcome of the study in relation to the evaluated user group, which we defined as novice users. For this target group the study showed that the implemented pointing technique, which we called mouse technique, was the best suited technique for both single and multiple selection of screen content. With this result being relative to our specific implementation and the target group, we tried to outline throughout the discussion which elements are important when designing for optical marker less tracking, as the Kinect and the NITE framework. The outcome was concluded by a list of design guidelines applying to such input.

- ✓ Make sure Proper lighting condition
- ✓ Lens should be clean
- ✓ Minimum distance is 0.4 and maximum is 4
- ✓ Use latest SDK provided by useful support communities
- ✓ Make sure good internet connection for latency free work

By concluding all these points we can sure to make good interaction system. So happy Kinect!

5.4 FUTURE

Scientists from MIT have divulged another manifestation of movement following that uses a three point framework to take after an individual's position, even though a completely dark divider. In spite of the fact that the saying "Kinect" has been tossed around generously for the purpose of openness, this is strictly a positional tracker that implies that it won't be deciphering gesture based communication or perusing lips whenever soon. Instead of being a control instrument, this gadget is only for watching clients as they move both inside and between rooms. At present the tracker is situated up directionally, so it can just transparent the single divider at which it is pointed, yet the evident finished objective is an omnidirectional tracker that could complete a client the entire house, upstairs and down.

The framework works utilizing three radio receiving wires divided around a meter separated to bob indicators off an individual's body. Indeed through the specialists' office divider, it can take after individuals with a precision of up to 10 centimeters (four inches), superior to Wifi restriction can presently give. Despite the fact that the gadget is blasted and sitting as part parts at present, one graduate learner taking a shot at the undertaking said they hope to have the capacity to gather it down to a last unit no bigger than Microsoft's Kinect sensor.There is much more we can develop from proposed work, now with this setup we can transfer information in form of gesture and if can be outreach as a very low bandwidth requirement.

In Microsoft research lab Kinect as a prime source to develop real time controller as a touch less interface.

REFERENCES

1. Antle, A.N., Corness, G. & Droumeva, M., 2009. What the body knows: Exploring the benefits of embodied metaphors in hybrid physical digital environments. *Interacting with Computers*, 21, pp.66-75. Special issue: Enactive Interfaces.
2. Baddeley, A., 1997. *Human Memory, Theory and Practice*. Revised Edition ed. East Essex, United Kingdom: Psychology Press Ltd.
3. Bowman, D.A., 1998. Interaction Techniques for Immersive Virtual Environments: Design, Evaluation, and Application. *Journal of Visual Languages and Computing*, 10, pp.37-53.
4. Bowman, D.A., 1999. *Interaction Techniques for Common Tasks in Immersive Virtual Environments*. PhD Thesis. United States of America: Georgia Institute of Technology. A Thesis Presented to The Academic Faculty.
5. Bowman, D.A. et al., 2006. New Directions in 3D User Interfaces. *International Journal of Virtual Reality*, 05(02), pp.3-14.
6. Bowman, D.A. & Hodges, L.F., 1999. Formalizing the Design, Evaluation, and Application of Interaction Techniques for Immersive Virtual Environments. *Journal of Visual Languages and Computing*, 10, pp.37-53.
7. Bowman, D.A., Koller, D. & Hodges, L.F., 1997. Travel in Immersive Virtual Environments: An Evaluation of Viewpoint Motion Control Techniques. In *Virtual Reality Annual International Symposium, 1997., IEEE*. Albuquerque, NM, 1997. IEEE.
8. Bowman, D.A., Kruijff, E., LaViola, J.J. & Poupyrev, I., 2005. *3D User Interfaces: Theory and Practice*. New York: Addison-Wesley.
9. Bruder, G., Steinicke, F. & Hinrichs, K.H., 2009. Arch-Explore: A Natural User Interface for Immersive Architectural Walkthroughs. In *3D User Interfaces, 2009. 3DUI 2009. IEEE Symposium on*. Lafayette, LA, 2009. IEEE.
10. Bullinger, H.-J., Bauer, W., Wenzel, G. & Blach, R., 2010. Towards user centred design (UCD) in architecture based on immersive virtual environments. *Computers in Industry*, 61(4), pp.372-79. Human-Centered Computing Systems in Industry - A Special Issue in Honor of Professor G. Salvendy.

11. Burmeister, O.K., 2000. Usability Testing: Revisiting Informed Consent procedures for testing Internet sites. In *CRPIT '00 Selected papers from the second Australian Institute conference on Computer ethics*. Canberra, 2000. Australian Computer Society.
12. Buxton, W. et al., 2011. *Human Input to Computer Systems: Theories, Techniques and Technology*. [Online] Available at: http://www.billbuxton.com/inputManuscript.html [Accessed 01 March 2011]. Chapter 7.
13. Chen, J. & Bowman, D., 2009. Domain-specific design of 3d interaction techniques: An approach for designing useful virtual environment applications. *Presence: Teleoper. Virtual Environ.*, 18(5), pp.370-86. 107
14. Chen, J., Pyla, P.S. & Bowman, D.A., 2004. Testbed Evaluation of Navigation and Text Display Techniques in an Information-Rich Virtual Environment. In *Virtual Reality, 2004. Proceedings. IEEE.*, 2004. IEEE.
15. Choumane, A., Casiez, G. & Grisoni, L., 2010. Buttonless Clicking: Intuitive Select and Pick-release Through Gesture Analysis. *Virtual Reality Conference (VR), 2010 IEEE* , pp.67-70.
16. Cohen, B.H. & Lea, R.B., 2004. *Essentials of statesitcs for the social and behavioral sciences*. Hoboken, New Jersey, United States of America: John Wiley & Sons.
17. Denzin, N.K., 2009. *The research act, a theoretical introduction to sociological methods*. First paperback printing ed. United States of America. Was originally printed in 1970.
18. Dominjon, L. et al., 2005. The "Bubble" Technique: Interacting with Large Virtual Environments Using Haptic Devices with Limited Workspace. In *Eurohaptics Conference, 2005 and Symposium on Haptic Interfaces for Virtual Environment and Teleoperator Systems, 2005. World Haptics 2005. First Joint.*, 2005. IEEE.
19. Drettakis, G., Roussou, M., Reche, A. & Tsingos, N., 2007. Design and Evaluation of a Real-World Virtual Environment for Architecture and Urban Planning. *Presence: Teleoper. Virtual Environ.*, 16(3), pp.318-32.
20. Encyclopædia Britannica, 2011. *biology, philosophy of*. [Online] Available at: http://www.britannica.com.zorac.aub.aau.dk/EBchecked/topic/681551/biology-philosophy-of/283549/Taxonomy [Accessed 21 April 2011].

21. Ericsson, K.A. & Kintsch, W., 1995. *Long-Term Working Memory*. technical report. University of Colorado, Boulder.

22. Eysenck, M.W. & Keane, M.T., 2005. *Cognitive Psychology, a student's handbook*. First Edition ed. East Sussex, United Kingdom: Psychology Press Ltd.

23. Forsberg, A.S. et al., 2000. Immersive Virtual Reality for Visualizing Flow Through an Artery. In *Visualization 2000. Proceedings*. Salt Lake City, UT, USA , 2000. Brown University, Providence.

24. Foxlin, E., 2002. Motion Tracking Requirements and Technologies. In K. Stanney, ed. *Handbook of Virtual Environment Technology*. Lawrence Erlbaum Associates. pp.163-210.

25. Froehlich, B. & Bowman, D., 2009. 3D User Interfaces. *IEEE Computer Graphics and Applications*, Nov./Dec. pp.20-21.

26. Frøkjær, E., Hertzum, M. & Hornbæk, K., 2000. Measuring Usability: Are Effectiveness, Efficiency, and satisfaction really correlated? In *Proceedings of the SIGCHI conference on Human factors in computing systems (CHI '00)*. The Hague, Netherlands, 2000. ACM Press.

27. Gerber, D. & Bechmann, D., 2005. The Spin Menu: a Menu System for Virtual Environments. In *Virtual Reality, 2005. Proceedings. VR 2005. IEEE.*, 2005. IEEE.

28. Gibson, J.J., 1979. *The ecological approach to visual perception*. Boston: Houghton Mifflin.

29. Gutiérrez, M., Thalmann, D. & Vexo, F., 2005. Semantic Virtual Environments with Adaptive Multimodal Interfaces. In *Multimedia Modelling Conference, 2005. MMM 2005. Proceedings of the 11th International*., 2005. IEEE. 108

30. Hirose, N., 2002. An ecological approach to embodiment and cognition. *Cognitive Systems Research*, 3, pp.289-99.

31. Hoang, T.N., Porter, S.R. & Thomas, B.H., 2009. Augmenting Image Plane AR 3D Interactions for Wearable Computers. In *AUIC '09: Proceedings of the Tenth Australasian Conference on User Interfaces*., 2009. Australian Computer Society, Inc..

32. Igarashi, T., Matsuoka, S. & Tanaka, H., 2007. Teddy: A Sketching Interface for 3D Freeform Design. In *SIGGRAPH '07 ACM SIGGRAPH 2007 courses*., 2007. ACM.

33. Jacob, R.J. et al., 2008. Reality-based interaction: a framework for post-WIMP interfaces. In *Proceeding of the twenty-sixth annual SIGCHI conference on Human factors in computing systems (CHI '08)*. Florence, Italy, 2008. ACM.
34. Kinect Games, X.3., 2010. *Xbox 360 Kinect Games*. [Online] Available at: http://www.xbox.com/en-US/Kinect/Games [Accessed May 2011].
35. Kinect, M., 2010. *Kinect*. [Online] Available at: http://www.xbox.com/en-US/kinect [Accessed May 2011].
36. Klatzky, R.L. & Lederman, S.j., 2003. Touch. *Expermental Psychology*, 4, pp.147-76.
37. Kulik, A., 2009. Building on realism and magic for designing 3D interaction techniques. *IEEE Comput. Graph. Appl.*, 29(6), pp.22-33.
38. LaViola, J. et al., 2009. CHI 2009 Course Notes: User Studies and 3D UIs. In *3D User Interfaces: Design, Implementation, Usability - CHI'09.*, 2009. ACM.
39. Lee, M. & Billinghurst, M., 2008. A Wizard of Oz Study for an AR Multimodal Interface. In *ICMI '08 Proceedings of the 10th international conference on Multimodal interfaces.*, 2008. ACM.
40. Liang, J. & Green, M., 1994. JDCAD: A Highly Interactive 3D Modeling System. *Computers & Graphics*, pp.499-506.
41. Lindeman, R.W., 1999. *Bimanual Interaction, Passive-Haptic Feedback, 3D Widget Representation, and Simulated Surface Constraints for Interaction in Immersive Virtual Environments*. The School of Engineering and Applied Science of The George Washington University. ph.d. thesis.
42. Lindeman, R.W., Sibert, J.L. & Hahn, J.K., 1999. Towards usable VR: an empirical study of user interfaces for immersive virtual environments. In *CHI '99: Proceedings of the SIGCHI conference on Human factors in computing systems: the CHI is the limit.*, 1999. ACM.
43. Liu, L., Martens, J.-B. & Liere, R.v., 2010. Revisiting Path Steering for 3D Manipulation Tasks. *International Journal of Human-Computer Studies*, 69(03), pp.170-81.
44. MacKenzie, I.S., 1992. Fitts' Law as a Research and Design Tool in Human-Computer Interaction. *Human-Computer Interaction*, 7(1), pp.91-139.

45. Maguire, M., 2001. Methods to support human-centred design. *International Journal of Human-Computer Studies*, 55(4), pp.587-634.
46. Maier, P. et al., 2010. What Do You Do When Two Hands Are Not Enough? Interactive Selection of Bonds Between Pairs of Tangible Molecules. *3D User Interfaces (3DUI), 2010 IEEE Symposium on* , pp.83-90. 109
47. Martens, J.-B. et al., 2004. Experiencing 3D Interactions in Virtual Reality and Augmented Reality. *EUSAI '04: Proceedings of the 2nd European Union symposium on Ambient intelligence* .
48. McGurk, H. & MacDonald, J., 1976. hearing lips and seing voices. *Nature,* 264, pp.746-48.
49. McTear, M.F., 2001. Spoken Dialogue Technology: Enabling the Conversational User Interface. *ACM Computing Surveys (CSUR),* 34(1), pp.90-169.
50. Miller, G.A., 1955. The Magical Number Seven, Plus or Minus Two Some Limits on Our Capacity for Processing Information. *Psychological Review*, 101(02), pp.343-52.
51. Mine, M.R., 1995. *Virtual Environment Interaction Techniques*. Report TR95-018. University of North Carolina. Computer Science Technical Report.
52. Nielsen, J., 1993. *Usability Engineering*. San Francisco: Morgen Kaufmann.
53. Nielsen, J., 2000. *Why You Only Need to Test With Five Users*. [Online] Available at: http://www.useit.com/alertbox/20000319.html [Accessed 21 April 2011].
54. Nielsen, J., 2005. *Ten Usability Heuristics*. [Online] Available at: http://www.useit.com/papers/heuristic/heuristic_list.html [Accessed 15 March 2011].
55. Norman, D.A., 1990. *The Design of Everyday Things*. New York: Basic Books.
56. Nvidia, 2011. *Nvidia 3D Vision*. [Online] Available at: http://www.nvidia.com/object/product_GeForce_3D_VisionKit_us.html [Accessed May 2011].
57. OpenNI, 2010. *Documentation*. [Online] Available at: http://www.openni.org/documentation [Accessed 12 May 2011].
58. OpenNI, 2011. *OpenNI*. [Online] Available at: http://www.openni.org [Accessed May 2011].

59. Pai, D.K., 2005. Multisensory Interaction: Real and Virtual. *Springer Tracts in Advanced Robotics*, 15, pp.489-98.

60. Peck, T.C., Fuchs, H. & Whitton, M.C., 2010. Improved Redirection with Distractors: A Large-Scale-Real-Walking Locomotion Interface and its Effect on Navigation in Virtual Environments. In *Virtual Reality Conference (VR), 2010 IEEE*. Waltham, MA, 2010. IEEE.

61. Phillips, L., Ries, B., Kaeding, M. & Interrante, V., 2010. Avatar self-embodiment enhances distance perception accuracy in non-photorealistic immersive virtual environments. In *Virtual Reality Conference (VR)*. Waltham, MA , 2010. IEEE.

62. Pierce, J.S. et al., 1997. Image Plane Interaction Techniques In 3D Immersive Environments. *I3D '97: Proceedings of the 1997 symposium on Interactive 3D graphics* , April. pp.39-43.

63. Pierce, J.S., Matthew, C., Dantzich, M.v. & Robertson, G., 1999. Toolspaces And Glances: Storing, Accessing, And Retrieving Objects In 3D Desktop Applications. In *I3D '99 Proceedings of the 1999 symposium on Interactive 3D graphics*., 1999. ACM.

64. Pierce, J.S., Steams, B.C. & Pausch, R., 1999. Voodoo Dolls: Seamless Interaction at Multiple Scales in Virtual Environments. In *I3D '99 Proceedings of the 1999 symposium on Interactive 3D graphics*., 1999. ACM. 110

65. Poupyrev, I., Billinghurst, M., Weghorst, S. & Ichikawa, T., 1996. The Go-Go Interaction Technique: Non-linear Mapping for Direct Manipulation in VR. *Proceedings of the ACM Symposium on User Interface Software and Technology*, pp.78-80.

66. PrimeSense Ltd., 2010. *Reference Design*. [Online] Available at: http://www.primesense.com/?p=514 [Accessed 25 February 2011].

67. PrimeSense, 2011. *NITE Algorithms 1.3 (User Manual)*. [Online] Available at: http://www.primesense.com [Accessed May 2011].

68. PrimeSense, 2011. *PrimeSense*. [Online] Available at: http://www.primesense.com/?p=515 [Accessed May 2011].

69. PrimeSensor, 2011. *PrimeSense - PrimeSensor*. [Online] Available at: http://www.primesense.com/?p=514 [Accessed May 2011].

70. Rademacher, P.M., 2002. *Measuring the Perceived Visual Realism of Images*. Ph.d. Chapel Hill: University of North Carolina.
71. Ray, A.A., 2008. *The Interaction Framework For Innovation: A Method to Create Reusable Three-Dimensional Interaction Techniques*. Blacksburg, VA: Virginia Polytechnic Institute and State University. ph.d. thesis.
72. Sarter, N.B., 2006. Multimodal information presentation: Design guidance and research challenges. *International Journal of Industrial Ergonomics* , 36, pp.439-45.
73. Sato, S. & Salvador, T., 1999. Playacting and Focus Troupes: Theater techniques for creating quick, Intense, immersive, and engaging focus Group sessions. *Interactions - New York*, 6(5), pp.35-41.
74. Schuchardt, P. & Bowman, D., 2007. The benefits of immersion for spatial understanding of complex underground cave systems. In *Proceedings of the 2007 ACM symposium on Virtual reality software and technology (VRST '07)*. Newport Beach, California, 2007. ACM.
75. Shneiderman, B., 1983. Direct manipulation: a step beyond programming languages. *IEEE Computer,* 16(8), pp.57-69.
76. Shneiderman, B., 2003. Why Not Make Interfaces Better than 3D Reality? *IEEE Comput. Graph. Appl.*, 23(6), pp.12-15.
77. Shneiderman, B., 2005. Leonardo's laptop: human needs and the new computing technologies. In *Proceedings of the 14th ACM international conference on Information and knowledge management*. Bremen, Germany, 2005. ACM.
78. Shotton, J. et al., 2011. *Microsoft Research*. [Online] Available at: http://research.microsoft.com/apps/pubs/default.aspx?id=145347 [Accessed 12 May 2011].
79. Slater, M. & Steed, A., 2000. A Virtual Presence Counter. *Presence: Teleoper. Virtual Environ.*, 9(5), pp.413-34.
80. Society for Technical Communication, 2011. *Usability Techniques Methods for Successful "Thinking Out Loud" Procedure*. [Online] Available at: http://www.stcsig.org/usability/topics/articles/tt-think_outloud_proc.html [Accessed 14 April 2011]. 111

81. Stoakley, R., Conway, M.J. & Pausch, R., 1995. Virtual Reality on a WIM: Interactive Worlds in Miniature. In *CHI '95 Proceedings of the SIGCHI conference on Human factors in computing systems*., 1995. ACM.
82. Suma, E.A. et al., 2010. Effects of Travel Technique and Gender on a Divided Attention Task in a Virtual Environment. In *3D User Interfaces (3DUI), 2010 IEEE Symposium on*., 2010. IEEE.
83. Terziman, L. et al., 2010. Shake-Your-Head: Revisiting Walking-In-Place for Desktop Virtual Reality. In *Proceedings of the 17th ACM Symposium on Virtual Reality Software and Technology*. Hong Kong, 2010. ACM.
84. Tullis, T. & Albert, B., 2008. *Measuring the user experience*. Burlington, United states of America: Morgan Kaufmann.
85. Unity3D, 2011. *http://unity3d.com*. [Online] [Accessed May 2011].
86. Valkov, D., Steinicke, F., Bruder, G. & Hinrichs, K., 2010. A multi-touch enabled human-transporter metaphor for virtual 3D traveling. In *3D User Interfaces (3DUI), 2010 IEEE Symposium on*. Waltham, MA , 2010. IEEE.
87. Watson, B., Spaulding, V., Walker, N. & Ribarsky, W., 1997. Evaluation of the effects of frame time variation on VR task performance. In *Proceedings of the 1997 Virtual Reality Annual International Symposium (VRAIS '97)*. Washinton DC, 1997. IEEE Computer Society.
88. Welch, G. & Foxlin, E., 2002. Motion Tracking: No Silver Bullet, but a Respectable Arsenal. *Computer Graphics and Applications, IEEE* , 22(6), pp.24 - 38.
89. Wendt, J.D., Whitton, M.C. & Brooks, F.P., 2010. GUD WIP: Gait-Understanding-Driven Walking-In-Place. In *Virtual Reality Conference (VR), 2010 IEEE*. Waltham, Massachusetts, USA, 2010. IEEE.
90. White, S., Feng, D. & Feiner, S., 2009. Interaction and Presentation Techniques for Shake Menus in Tangible Augmented Reality. In *Mixed and Augmented Reality, 2009. ISMAR 2009. 8th IEEE International Symposium on*., 2009. IEEE.
91. Wingrave, C.A., 2008. *Concept-Oriented Design in Chasm: Conversational Domain Language Inspired 3D User Interface Design and Development*. Blacksburg, VA: Virginia Polytechnic Institute and State University. ph.d. thesis.

92. Wingrave, C.A., Bowman, D.A. & Ramakrishnan, N., 2002. Towards Preferences in Virtual Environment Interfaces. In *EGVE '02 Proceedings of the workshop on Virtual environments*., 2002. Eurographics Association Aire-la-Ville.
93. Wixon, D., 2008. *UX Week - The Challenge of Emotional Innovation*. [Online video: http://vimeo.com/2893051] Available at: http://vimeo.com/2893051 [Accessed May 2011]. Citation can be found at 7.31 in the video.
94. Wyss, H.P., Blach, R. & Bues, M., 2006. iSith – Intersection-based Spatial Interaction for Two Hands. *3D User Interfaces, 2006. 3DUI 2006. IEEE Symposium on* , pp.59-61