TAMPERE UNIVERSITY OF TECHNOLOGY

SAMULI PARKKINEN

GESTURE-BASED INTERACTION WITH MODERN INTERACTION DEVICES IN DIGITAL MANUFACTURING SOFTWARE

Master of Science Thesis

Examiners: Professor Seppo Kuikka,
Professor Kaisa Väänänen-Vainio-Mattila
Examiner and topic approved in
the Faculty of Automation,
Mechanical and Materials
Engineering Council
meeting on October 3, 2012

ABSTRACT

TAMPERE UNIVERSITY OF TECHNOLOGY
Master's Degree Programme in Automation Technology
PARKKINEN, SAMULI: Gesture-based interaction with modern interaction devices in digital manufacturing software
Master of Science Thesis, 93 pages, 9 Appendix pages
October 2012
Major: Software Engineering in Automation
Examiner: Professor Seppo Kuikka, Professor Kaisa Väänänen-Vainio-Mattila
Keywords: Human-Computer Interaction, 3D world, Digital Manufacturing, Gestures, Gesture-Based Control, User Interface, Touch Screen, Microsoft Kinect, 3DConnexion SpacePilot PRO, 3D Mouse

Traditionally, equipment for human-computer interaction (HCI) has been a keyboard and a mouse, but in the last two decades, the advances in technology have brought completely new methods for the HCI available. Among others, digital manufacturing software 3D world has been controlled with the keyboard and mouse combination. Modern interaction devices enable more natural HCI in the form of gesture-based interaction. Touch screens are already a familiar method for interacting with computer environments, but HCI methods that utilize vision-based technologies are still quite unknown for a lot of people. The possibility of using these new methods when interacting with 3D world has never been studied before.

The main research question of this MSc. thesis was how the modern interaction devices, namely touch screen, Microsoft Kinect and 3DConnexion SpacePilot PRO, can be used in interacting with 3D world. The other research question was how the gesture-based control should be utilized with these devices. As a part of this thesis work, interfaces between 3D world and each of the devices were built.

This thesis is divided into two main parts. The first background section deals with the interaction devices, 3D world, and also gives the necessary information that is needed in fully utilizing the possibilities of these interaction devices. The second part of the thesis is about building the interfaces for each the above-mentioned devices.

The study indicates that the gesture-based control with these interaction devices cannot replace the functionality of a keyboard and a mouse, but each of the devices can be used for certain use cases in particular use scenarios. Two dimensional gesture-based control on touch screen suits well for using camera controls as well as doing the basic manipulation tasks. Three dimensional gesture-based control when using Kinect is applicable when it is used in specially developed first person mode. Kinect interface requires a calm background and quite a large space around the user to be able to be used correctly. Suitable use scenario for this interface is doing a presentation to audience in front of an audience in a conference room. The interface for SpacePilot PRO suits well either for controlling the camera or manipulating object positions and rotations in 3D world.

TIIVISTELMÄ

Tampereen Teknillinen Yliopisto
Automaatiotekniikan koulutusohjelma
PARKKINEN, SAMULI: Elepohjainen vuorovaikutus moderneilla interaktiolaitteilla digitaalisen valmistuksen järjestelmässä
Diplomityö, 93 sivua, 9 liitesivua
Lokakuu 2012
Pääaine: Automaation ohjelmistotekniikka
Työn tarkastajat: Professori Seppo Kuikka ja Professori Kaisa Väänänen-Vainio-Mattila
Avainsanat: Ihminen-tietokone vuorovaikutus, 3D-maailma, digitaalinen valmistus, eleet, elekäyttö, käyttöliittymä, Microsoft Kinect, 3DConnexion SpacePilot PRO, 3D hiiri

Perinteisesti ihmisen ja koneen välinen vuorovaikutus on tapahtunut käyttäen näppäimistöä ja hiirtä, mutta tekniikan kehitys parin viimeisen vuosikymmenen aikana on tuonut täysin uudentyyppisiä menetelmiä tälle saralle. Myös digitaalisen valmistuksen sovellus 3D-maailmaa käytetään tyypillisesti näppäimistön ja hiiren avulla. Moderneista interaktiolaitteista kosketusnäytöt ovat tuttuja useimmille käyttäjille, mutta konenäköön perustuvat laitteet ovat vielä usealle aivan tuntemattomia. Nämä interaktiolaitteet mahdollistavat eleiden käyttämisen syötteen välittämiseen. Uusien interaktiolaitteiden mahdollisuuksia 3D-maailman käyttämisen yhteydessä ei ole aikaisemmin tutkittu.

Tämän diplomityön päätutkimusongelma oli kuinka interaktiolaitteita (kosketusnäyttö, Microsoft Kinect ja 3DConnexion SpacePilot PRO) voidaan käyttää vuorovaikutukseen 3D maailman kanssa. Toinen tutkimusongelma oli kuinka elepohjaista vuorovaikutusta tulisi hyödyntää näitä laitteita käytettäessä. Osana työtä luotiin sekä ohjelmalliset rajapinnat että käyttöliittymät 3D maailman ja jokaisen laitteen välille.

Tämä opinnäytetyö on jaettu kahteen pääosioon. Ensimmäisessä taustatieto-osio esittelee interaktiolaitteet, 3D maailma ja muu rajapintojen luomisessa tarvittava tieto. Toisessa osiossa keskitytään rajapintojen määrittelyyn ja toteutukseen.

Työ osoittaa, että elekäyttöliittymästä tai mainituista interaktiolaitteista ei ole perinteisen näppäimistö-hiiri -yhdistelmän korvaajaksi, mutta jokaista laitetta voidaan käyttää tiettyjen toimintojen suorittamiseen määrätyissä olosuhteissa. Kaksiulotteinen, elepohjainen käyttö kosketusnäytöllä sopii hyvin sekä kameran ohjaukseen, että perus objektien muokkaustoimenpiteisiin. Kolmiulotteinen, elepohjainen käyttö Kinectillä sopii hyvin käytettäväksi erityisesti tätä varten suunnitellussa ihmisperspektiivitilassa. Kinect-rajapinnan käyttö vaatii staattisen taustan ja melko ison alueen käyttäjän ympärille. Siksi rajapinta soveltuu käytettäväksi vain esimerkiksi 3D mallien esittelyyn. SpacePilot PRO rajapinta soveltuu hyvin sekä kameran kontrollointiin, että objektien sijainnin ja rotaation muokkaamiseen.

PREFACE

This MSc. thesis was written for Visual Components. I would like to thank people at the company Visual Components for this chance to make my thesis on such an interesting subject. I would especially like to thank Ricardo Velez for tutoring me through this thesis and giving really good advices along the way, Mika Anttila and Antto Rossi for guidance on 3D world and using the COM interface, and Juha Renfors for giving this this possibility in general.

Special thanks go also to my guiding professors Seppo Kuikka and Kaisa Väänänen-Vainio-Mattila from Tampere University of Technology for their valuable advice and expertise for this thesis.

I would like to express my gratitude to my parents Mikko and Tiina Parkkinen for their interest towards my studies and guidance throughout my life. I would like to thank Minna Salomaa for checking the grammar. I would also like to thank my mental assistants Henri Tikkanen and Vincent Arcis for their support along the way.

Last but not least, I would like to thank my lovely fiancée Elli-Maija Martikainen for her support and understanding during this thesis work.

CONTENTS

ABBREVATIONS, TERMS AND DEFINITIONS

.NET	.NET Framework. Software framework that runs on Microsoft Windows.
2D	Two dimensional.
3D	Three dimensional.
3D mouse	Six degree of freedom movement controller that is used in navigating in 3D spaces.
6DoF	Six Degrees of Freedom.
ActiveX	Framework for defining reusable software controls.
API	Application Programming Interface.
ASP.NET	Web application framework.
ATM	Automatic Teller Machine.
C++	Programming language.
C#	Programming language.
CAD	Computer Aided Design.
Chordic Manipulation	Manipulation performed with fingers on touch screen.
CLI	Common Language Infrastructure.
CLR	Common Language Runtime.
CMOS	Monochrome Complementary Metaloxide Semiconductor.
COM	Component Object Model.
CTS	Common Type System.
DirectX	Collection of Application Programming Interfaces.
DLL	Dynamic Linked Library.
EXE	Executable file.
Forms	Windows Forms. Graphical application programming interface for .NET framework.
FSM	Finite State Machine.
Gesture	Form of non-verbal communication.

HCI	Human-Computer Interaction.
HTTP	HyperText Transfer Protocol.
IL	Intermediate Language.
IPT	Integrated Product Team.
IR	Infra-Red.
JAVA	Programming language.
JIT	Just-In-Time.
Kinect	Microsoft Kinect.
LED	Light Emitting Diode.
LCD	Liquid Crystal Display.
MGS	Motion Generation from Semantics.
MTS	Multi Touch Surface.
MVC	Model-View-Controller.
MVP	Model-View-Presenter.
MVVM	Model-View-ViewModel.
NGWS	Next Generation Windows Services.
NUI	Natural User Interface.
OLE	Microsoft OLE.
PAC	Presentation-Abstraction-Control.
PACT	People Activities Contexts Technologies.
PDF	Portable Document Format.
PnP	Plug and Play.
RGB	Red-Green-Blue.
SAW	Surface Acoustic Wave.
SDK	Software Developer Kit.
SpacePilot PRO	3DConnexion SpacePilot PRO. Movement controller.
TIO	Indium Tin Oxide.

UI	User Interface.
VB	Visual Basic.
Wii	Nintendo Wii.
Windows	Microsoft Windows operating system.
WPF	Windows Presentation Foundation. Next generation graphical application programming interface for .NET framework.
XAML	Extensible Application Markup Language.
Xbox 360	Microsoft Xbox 360 gaming console.
XML	Extensible Markup Language.

1 INTRODUCTION

The success story of smart phones and tablets has brought touch screens to workplaces and homes. Touch screen enables a new kind of user interface (UI), where user can interact with the computer by using more natural actions. Selecting the action by pointing a spot on the screen is really intuitive, but the screen also enables new type of control where the normal human actions are mimicked on the screen. These actions are called gestures and they are used in controlling the applications. An example of a gesture that is used with touch screen is the pinch gesture, in which the user mimics compressing something between fingers into smaller space. It is used in zooming out.

Gestures and gesture-based control have been under study for the last two decades. Previously the study focused on two dimensional gestures on touch screens, but in the last decade as the equipment has been developing and the tracking of human movement in three dimensions has become possible, the focus of the research has been moving towards gestures in three dimensions. Tracking of human movement can be done by using handheld sensors or by placing sensors all over the body and using the sensing equipment to find out the positions of sensors. Vision-based technology can also be used in tracking human movement. Previously, equipment in the field of computer vision has been really expensive and only available to large industries and nations. However, the completely new and innovative interaction devices, such as Microsoft Kinect (Kinect), have been fairly affordable to consumers, thanks to the recent advances in technology. This has been a real kick start for research in the area of gestures in three dimensions.

New technologies bring new challenges: How to make the best use of these technologies? What kind of interaction should be used with them? What are gestures? How to recognize them? How to use them in the best possible way? As vision-based technology has been used previously only in the gaming industry, most of the former three dimensional gesture-based applications are games. Researchers have been developing various methods for gesture-based interaction and gesture-recognition, but the research is still in the very early stages, and the best practices and standards are yet to be found

In the digital manufacturing field, the human-computer-interaction (HCI) is typically done with a traditional keyboard and a mouse combination. The use of these interaction devices is very developed and macro-based. It requires a lot of knowledge and a long time to learn all the necessary keyboard button and mouse button combinations and use them effectively in everyday work. There is a definitely a need for a better interaction method, but currently there is no actual replacement for keyboard and mouse.

There are hundreds of functions available in digital manufacturing software, which are used through the keyboard and mouse. New interaction devices cannot completely replace this traditional interaction, but in some cases, different kind of interaction might work better or supplement the working solutions, making the use of software easier,

faster, and more fun. This thesis aims to find proper solutions for using interaction devices in digital manufacturing environment. The interaction devices addressed in this thesis are Kinect, touch screens and 3DConnexion SpacePilot PRO (SpacePilot PRO) 3D mouse.

1.1 Aims of the study

The main research question of this study is:

- How the interaction devices; Microsoft Kinect, 3DConnexion SpacePilot PRO and touch screen can be used in interacting with digital manufacturing software 3D world?

Gesture-based control is tightly integrated especially with Kinect and touch screen. The second main research question of this study is:

- How the gesture-based control can be used with these interaction devices?

As a part of this research, gestures and gesture-based interaction both in 2D and 3D environments were studied.

1.2 Structure of the study

The structure of the study is as follows. The first chapter is an introduction chapter to this thesis. This chapter introduces the overall field and the goals for this thesis.

Chapter number two is the background chapter which provides the necessary background information which was used in developing interfaces for these interaction devices. The second chapter consists of four subchapters. The first subchapter introduces the field of digital manufacturing. The second subchapter discusses gestures and gesture-based interaction. The third subchapter introduces the presentation patterns that are relevant to this thesis. The fourth subchapter focuses on the technology and tools that are used in building the interfaces.

The third chapter is about building the interface for each of the interaction devices. The third chapter consists of three subchapters and it is about building the interface for each of the interaction devices. The first subchapter is about the touch screen interface, the next subchapter discusses the building of interface for Kinect, and last subchapter introduces the interface for SpacePilot PRO.

The last fourth chapter is the conclusion and discussions chapter.

2 BACKGROUND

The main goal of this thesis work was to find out how the interaction devices Microsoft Kinect, touch screen and 3DConnexion SpacePilot PRO can interact with digital manufacturing software 3D world. This chapter presents the necessary background information that is relevant to this thesis. This chapter consists of four subchapters which are digital manufacturing; which introduces the application field of this thesis, gestures; which gives information about the gestures and gesture-based interaction that is used as main interaction method with Kinect and touch screen interface, design patterns; which focuses on the design patterns that are used in the development of the interfaces, and technology and tools; which presents the main technologies and tools that are used in building the interfaces.

2.1 Digital manufacturing

This section introduces the application field of digital manufacturing. Digital manufacturing is the use of a 3D computer environment that is used to simulate, visualize in three dimensions and analyze the creation of products and manufacturing processes simultaneously. [1] Brown breaks the digital manufacturing into ten methodologies:

1. Define all constraints and objectives of the production system.
2. Define the best process to build the product and its variants according to targeted constraints and objectives.
3. Define and refine the production system process resources and architecture, and measure its anticipated performance.
4. Define, simulate and optimize the production flow.
5. Define and refine the production layout.
6. Develop and validate the control and monitor functions of the production system. Execute the schedule.
7. Balance the line, calculate the costs and efficiencies of the complete production system and select the appropriate solution.
8. Download valid simulation results to generate executable shop-floor instructions.
9. Upload, accumulate and analyze performance data from actual production system operations to continuously optimize the production process.
10. Support field operations with maintenance instructions and monitor maintenance industry. [2]

By using the digital manufacturing system and following these principles, the entire enterprise can maintain complete control of process planning, product data, process data, manufacturing resources and deployment to the shop floor.

The Term digital manufacturing holds information about the whole manufacturing process. Seino, Ikeda, Kinoshita, Suzuki, and Atsumi extracted five modes of digital manufacturing for application of digital technology to manufacturing.

1. Transforming technological and technical know-how into numerical data. The know-how from the processes can be transformed into tangible knowledge and used in establishing digital data for production process conditions and phenomena.
2. Virtual design and manufacturing. Use of digital manufacturing environment allows manufacturers to realize prototypeless product design and manufacturing.
3. Product data management; extracting the meaning from data, processing and utilizing it. Technologies can be used in gathering data on production and quality from the production line and transformed into meaningful product information.
4. Consistent and collective use of data. Digital manufacturing environment allows utilizing of product data and three dimensional CAD data from design stage to production and from product order to shipping.
5. Remote management. These technologies enable monitoring, diagnosing, controlling and managing the production conditions from a remote location. [3]

Digital manufacturing can be used as a decision support tool for engineering team or integrated product team (IPT). They can use it to optimize the combinations of products and processes by trying different variations of these. [2] By using the digital manufacturing systems engineers are able to create complete representation of a manufacturing process in a virtual environment. [1] They can change various properties of the manufacturing system such as:

- Assembly lines.
- Facility layout.
- Tooling.
- Work centers.
- Resources.
- Ergonomics. [1]

The use of a digital manufacturing system helps the users during the initial planning of a new system as well as in the updating of existing systems. Digital manufacturing environment can be used for various tasks, for instance:

- Designing facility layouts.
- Simulating material flow in 3D.
- Simulating manual assembly.
- Making geometry based process planning.
- Making cost estimation reports.
- Using as a decision support tool for process execution.

- Off-line programming. [2]

The use of digital manufacturing benefits most the industries in capital-intensive manufacturing and the industries with very complex products with very low production, down to single unit production. In capital-intensive manufacturing, the use of digital manufacturing reduces the time to market, product cost, engineering changes to product design and production tooling during launch. For companies with very complex products, the use of digital manufacturing enables them to learn more from the process before actual manufacturing, which increases productivity and helps them to avoid unforeseen problems. This is crucial with the production of only one or two units. [2]

2.2 Gestures

Gestures play an important role in human-to-human communication, and they can also be used in interaction with computer systems. This section focuses on gestures, gesture recognition and gesture-based interaction. General discussion is followed by more focused information about both two and three dimensional gestures. The next subchapter evaluates the applicability of gesture-based interaction to digital manufacturing. Final subchapter concludes the main points of gestures and gesture-based interaction.

As computers are responsible for an increasing number of tasks in the society, HCI is becoming more important. Typically HCI is done with keyboard and mouse, which provide a stable and familiar way to access computers. Nevertheless, in some cases they cannot be accessed or they do not provide the best possible way of HCI because the interaction with them is too slow. [4] Modern interaction devices, such as Microsoft Kinect, Nindendo Wii (Wii) and multi touch surfaces provide an interesting and more natural way for HCI. The interaction devices by themselves open a channel to access the computer, but the challenge is how to use them in the best possible way.

One goal of HCI research is to increase the naturalness of HCI. By increasing naturalness, researchers speak of using similar methods in HCI that humans use in communicating with each other. [4] The most natural way of human communication is the verbal communication. It is being increasingly used as a method in HCI. For example Microsoft Kinect provides speech recognition equipment, which consists of equipment for capturing audio and tools for processing voice. The second main human-to-human communication method is the nonverbal communication. Non-verbal communication concludes all of the other communication methods that humans use, such as facial expressions or hand gestures. In HCI the most common way of mimicking human-to-human nonverbal communication is the gesture-based interaction.

The gesture itself can involve an object or exist in isolation. In this subchapter only the isolated gestures are examined, since they are ones that are used in HCI. These types of gestures are called semiotic gestures. Semiotic gestures are used to deliver meaningful information. [5]

A gesture is defined by Kurtenbach and Hulteen as a motion that contains information. Waving goodbye is a gesture. Pressing a key on a keyboard is not a gesture,

because the motion of a finger on its way to hitting the key is neither observed nor significant. All that matters is which key was pressed. [6] The result of pressing that key would be the same regardless of the gesture that was used to press the key. The motions and the feelings that the user felt while pressing the key are not shown in the result. And they cannot be easily observed. [5]

Gestures can be divided into two groups of static and dynamic gestures. A static gesture is a single pose formed by a single image, while a dynamic gesture consists of movement and forms from a sequence of images. Gestures can be divided into conscious and non-conscious gestures, depending whether the gesture was intended to do. [7] Human gestures can be divided into five main types according to their properties:

- Affect displays, which are gestures that communicate emotions or communicator's intentions. For example, smiling communicates happiness.
- Adaptors, which are gestures that enable the release of the body tension. For example, head shaking. These gestures are not intentional.
- Illustrators, which are gestures that depict the verbal communication. For example, when the speaker tells about throwing a ball, he/she makes a throwing gesture with the hand.
- Emblems, which are gestures that can be translated into short verbal messages. For example, waving hand for goodbye in order to replace words is an emblem.
- Regulators, which are gestures that control interaction. For example, stopping someone with an open palm is a regulator gesture. [7]

Gestures can be divided into three main types according to the body parts that are used in the gesture. The main types are hand gestures, head and face gestures, and full body gestures. [7] Currently most of the studies are made from the area of hand gestures and facial gestures. Numerous different gesture frameworks have been developed to address these fields.

Gestures can be categorized into two different main types according to their primary goal. Gestures that aim to send information while using them are called communicative gestures. [8] An example of a gesture in this category is showing a thumb up, which means good luck in American culture [9] and is considered a positive sign. Communicative gestures are usually offline gestures, which are introduced in chapter 2.2.1, as the information is generally sent after the gesture is complete.

The second main type of gestures is manipulative gestures. These gestures aim to manipulate graphical objects in two or three dimensions. [8] For instance, the spreading of fingers on touch surface is usually interpreted as zooming. The manipulative gestures are usually online gestures, which are introduced in chapter 2.2.1, because the manipulation happens while the gesture is made.

2.2.1 Gesture recognition

The technique of capturing gestures by computer is called gesture recognition. Gesture recognition uses mathematical interpretations of human motions by a computing device.

[10] Gesture recognition is used to trigger actions on the computer. There are two main types of gestures in gesture recognition: online gestures and offline gestures. An online gesture is one where the user input directly manipulates the view, for example rotating or scaling. An offline gesture means that the processing happens after the interaction has been finished. For example, a user draws a triangle in the air, and after finishing the gesture the computer plays a sound. [11]

Technology enabling gesture recognition consists of two main types of devices: contact-based devices and vision-based devices. Contact-based devices include: multi-touch screens, accelerators, controllers and instrumented gloves. [7] From this category the touch screen technologies are introduced later. Vision-based technology is based on one or several cameras. The captured video sequence is observed in order to analyze and interpret the motion of it. Various vision-based sensors include: monocular cameras, body markers and infrared cameras. From this category the Microsoft Kinect is introduced later in this chapter. [7]

The methods for gesture recognition vary a lot. It can be done through feature extraction and statistical classification methods. These methods consist of two stages. In the learning stage, the extracted features are categorized. In the classification stage, the movement is compared to learned features. In model-based methods the recognition process happens in a single stage where the target's parameters are extracted and then filled to the adequate gesture model. In template matching methods, the whole gesture is considered a template, instead of using either feature extraction or a gesture model. Hybrid methods are combinations of these methods. For example, one hybrid method is using finite state machines (FSM) and posture recognition, and another one is using exemplar based technique. [7]

2.2.2 Gesture-based interaction

HCI done with gestures is called gesture-based interaction. According to Ishii, using gesture-based interaction aims to empower collaboration, learning, and design by using digital technology and at the same time taking advantage of human abilities to grasp and manipulate physical objects and materials. [12] Gestures allow direct, natural, and intuitive way of HCI. [13] One benefit of using gesture-based interaction is that it makes a wider range of actions available to manipulate the system compared to traditional interfaces. Other benefit is its interface's ability to change any time, allowing it to be more customizable for application's needs than traditional interfaces. [14]

Gesture-based interaction is a more natural way of interaction than the interaction with a traditional keyboard and mouse. Disadvantage of using gesture-based interaction is that it is generally a less precise form of interaction. Systems generally provide about 90% accuracy in gesture-recognition, which is significantly less than the near 100% accuracy when using keyboard and mouse. Therefore the gesture-based interaction is not usually a replacement for traditional input methods in existing software, but it suits well as an additional input method for most existing systems. [15] Various gesture-only interfaces exist, and they suit well for applications where the exact precision is not re-

quired. Applications range from using a hand as a pointer to interfaces where the commands are triggered by using single poses. [5]

Lorenz, Jentsch, Concolato and Rukzio studied the usefulness of different input methods for controlling a multimedia application from distance. In the study they analyzed how fast users can perform a task of five single steps with hardware buttons, software buttons, touch screen gestures, and gestures in the air. After using the system with these input methods the participants filled a system usability questionnaire. The results showed that the fastest completion time and the highest usability ratings were achieved with the hardware buttons, which every test person was familiar with. Software buttons, the same buttons in the touch screen software, were a bit slower to use and ranked a little bit lower in the usability questionnaire. The completion time, when using the touch screen gestures was significantly lower than with either of the buttons, but the gestures in the air took the most time to complete. The usability ratings followed the time spent on the task: the less time people had to spend on the task, the higher usability ratings it got. This research shows that because the gestures are a new way of interacting with computer, people feel uncomfortable using them due to the fact that they are not used to using them in HCI. It also indicates that the gesture-based control cannot currently replace the traditional interfaces without losses in usability. [16]

Despite the fact that gestures do not fit in all kinds of environments, there are areas that the gesture-based interaction suits well, for example:

- Games, where the input to the game is directly mimicked by the game, for example in tennis game hitting the ball with a racket.
- Touch-screen interfaces, where additional commands can be added trough two dimensional gestures on the screen.
- Healthcare, for example surgeons can view x-ray images in operating room by using hand gestures.
- Music, for example Theremin, an instrument that is played by changing hand positions in the air.
- Security, where the violent or threatening actions can be identified in video surveillance material in order to get an alarm in advance.
- Sign language recognition, in which the gestures are interpreted as words.
- Presentations, in which the gestures of the speaker can be used to identify more precisely the context that the speaker is focusing on currently. [5] [7]

2.2.3 2D gestures

2D gestures are performed in two dimensions, usually on top of some multi touch surface (MTS) device or on a touch pad controller. Touch pads are the main pointer control method on laptops, while MTS screens are the most common interaction technology on smart phones and tablets. Gesture-based control is the main benefit of using MTS devices or touch pad controller when compared to mouse and keyboard. [17] Basic two dimensional gestures consist of one or multiple fingers performing actions on the con-

troller surface. These kinds of gestures are called chordic manipulations. Chordic manipulations combine tapping with fingers, holding fingers in place and sliding the fingers in certain directions across the surface. There are four main types of operations that can be performed on the 2D surface. [8] These operations are:

- Hand translation, which is done by sliding all of the touching fingers in the same direction across the surface at the same speed.
- Hand scaling, which is done by pinching the thumb and other participating fingers together or flicking them apart, while touching the surface of the touch panel.
- Chord tap, which is done by lifting the fingers quickly from the surface after touching it.
- Hand rotation, which is done by moving all the touching fingers in the circular path clockwise or counterclockwise, similar to movement used in opening or closing of a bottle cap. [8]

The possible finger combinations that can be used on surface are called channels. For instance, pushing the screen with two fingers selects the two-finger manipulation channel. Channels allow more commands to be available to be used with gestures. Up to eight commands can be applied to each channel in MTS by using the operators and their opposites. Most of the basic gestures require at least two fingers to be able to be performed correctly. For example in some cases one-finger rotation and one-finger translation can be exactly the same movement. Usually only one and two finger channels are used, as the basic gestures are done with one or two fingers. Using too many channels can be confusing. If the amount of channels increases, the mappings usually become hard to remember, which removes the main benefits of using gesture-based control: naturalness and intuitiveness. [8] [17]

Chordic manipulations and channels are combined to make basic gestures. The most common gestures are supported by basic Windows 7 touch input. They are illustrated in figure 2.1.

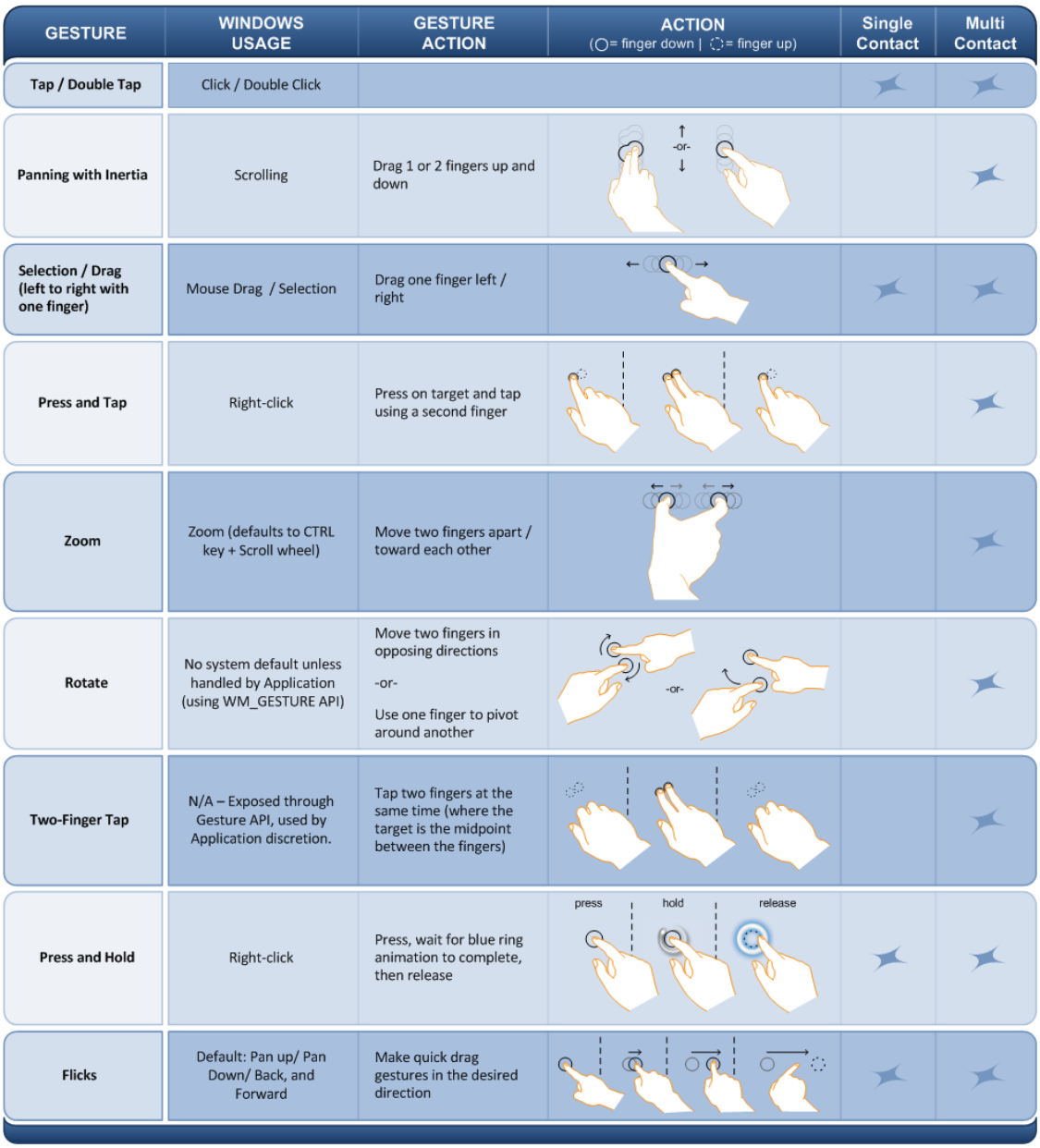


Figure 2.1. *2D Gestures supported by Windows 7 and actions mapped to the gestures by default. [18]*

Gesture-based control in two dimensions suits well for exploring information. It is especially practical in touch screen interfaces, as multi touch enables pushing, pulling, sorting, and visually arranging objects on the screen. This can be done very intuitively and naturally as the actions remind a lot of real world data exploration. [17]

Kin, Agrawala, and DeRose compared the performance of touch-screen direct-touch selection, bimanual selection and multifinger selection to selection with keyboard and mouse. In their study the test subjects had to select multiple targets with each of the mentioned techniques. Study showed that the direct-touch selection with one finger provided large performance benefits compared to mouse and keyboard selection. Bimanual selection added a small benefit to direct-touch. Multifinger selection provided no additional benefits and in some cases even reduced accuracy. [19]

Knoedel and Hachet compared the efficiency and precision of direct and indirect manipulation for rotation, scaling, and translation docking task. The task was performed both in three dimensional and two dimensional environments with the direct-touch screen and touchpad. The study showed that the time needed for task with direct interaction through touchscreen was shorter than the time needed with indirect interaction. In turn, the indirect interaction provided better efficiency and precision than the direct interaction. [20]

2.2.4 3D gestures

To be able to recognize gestures in three dimensions, the system requires some form of vision-based device or a controller, which can track the position of the user. Three dimensional gestures can be performed with the full body. For the recognition of human movement, the computer needs a model or an abstraction of the motion of the human body parts. Two main categories of gesture representation are: 3D model-based methods and appearance based methods. [7]

In 3D model-based methods, a three dimensional model defines the spatial description of the human body. 3D model-based gesture representation uses an automaton to handle the temporal aspect of a gesture. It divides the gesture in three parts: 1. the preparation or prestroke phase, 2. the nucleus or stroke phase, 3. the retraction or post stroke phase. These phases can be represented as transactions between one or several spatial stages of the three dimensional human model. The main benefit of these models is the recognition of gestures by synthesis: The advancing of the gesture is processed while one or several cameras follow the target. These methods offer precise detection of the gesture, but generally at the cost of computational performance. [7] Three main types of 3D models are:

- Textured kinematic/volumetric model, which contains a highly detailed model of a human body with skeleton and skin surface information.
- 3D geometric model, which is less precise than kinematic or volumetric models in terms of skin information but provide necessary skeleton information.
- 3D skeleton model, which contain only skeleton information. Skeleton provides information about the articulations and their 3D degree of freedom. [7]

Appearance-based methods include two dimensional static model-based methods and motion-based methods. Two dimensional static model-based methods include the following:

- Color-based models, which use colored body markers to track the movement of the full body or body part.
- Silhouette geometry -based models, which may include several geometric properties of the silhouette, for instance convexity, perimeter, compacity, surface and bounding box.
- Deformable gabarit-based models, which are based on deformable active contours. [7]

Two main categories of motion-based methods are:

- Global motion descriptor, which is based on stacking a sequence of tracked 2D silhouettes.
- Local motion descriptor, which overcomes the limitations of global motion by considering sparse and local spatio-temporal descriptors more robust to brief occlusions and to noise. [7]

One of the main benefits of using three dimensional gestures is the increased naturalness of control, when compared to both touch screen gestures and keyboard and mouse. Three dimensional gestures allow the data and the interface to share the same three dimensional space. [14] This allows the user to focus more on the task because the user always knows the position of his/her needed body parts. This helps in learning new tasks. When the control is done through the natural gestures, the user does not have to learn how to perform the action. For instance, when performing a completely new task with a keyboard and a mouse, the user has to first find out how the task is performed with these controls. With natural three dimensional gestures user can perform the action immediately. In conclusion the keyboard control requires a longer cognitive process than natural three dimensional gestures to be performed correctly. [21]

Two main difficulties of using three dimensional gesture-based interfaces are temporal segmentation ambiguity and spatial-temporal variability. Temporal segmentation ambiguity means the difficulty of defining of the starting and ending points of the continuous gesture, and spatial-temporal variability means that the gestures differ a lot between individuals. [22] In some cases these difficulties can lead to the complete lack of gesture recognition or to false interpretations of gestures. Because of this, the tasks which require high precision or accuracy are not suitable for gesture-only interfaces.

Wickeroth, Benölken and Lang have built a gestural recognition system for manipulating three dimensional objects and studied its usability of it in a user study. The study showed that the computer vision systems are at the level where it might effectively replace some traditional interfaces and augment others, enabling new functionalities and novel applications. [14]

Three dimensional gestures are applied widely in the game industry, most notably with Microsoft Xbox Kinect. Three dimensional gestures can be used to control the selector pointer on the screen. This can be done for instance with hands or by tracking the head position and estimating the point where the user looks. [23] Three dimensional gestures can also be used in vision-based sketching. [24] Studies show that the three dimensional gesture-only interface is very suitable for medical image viewing applications. [22] [25]

Kristensson, Nicholson and Quigley presented a bimanual markerless gesture-based interface for 3D full-body motion tracking sensors, such as Kinect. Their interface predicts the users' intended one-handed or two-handed gestures while they are being articulated. Their interface could be used to give straight commands with one or two hands, or by modulating the one-handed gestures with the non-dominating hand while the dominating hand is performing the gesture. As a part of this research they built a gesture set

which included the entire alphabet that could be used in writing in the air. Their research included a user study where the users tested the interface. The results showed that their interface was capable of 92.7-96.2% accuracy in gesture recognition. [26]

Three dimensional gesture-based interaction with Nintendo Wii tangible user interface (TUI) was compared to keyboard interaction in a study by Guo and Sharlin. Researchers compared the speed and accuracy in performing two different tasks: a posture-task; in which the user's postures were directly mapped to robot dog, and a navigation-task; in which the user had to navigate a robot dog through a route by using abstract mapping of gestures. The Wii TUI outperformed the keyboard control in both tasks in speed, and the number of errors significantly decreased when the Wii control was used. The study shows that the gestural TUI can be a more suitable option for user interface in certain tasks. [21]

Lourenço and Thinyane compared the usefulness of gesture-based interaction and keyboard/mouse control in user study. Users had to perform simple tasks, for example pointer movement, clicking and zooming, with both their Wii3D gesture framework and keyboard and mouse. The study showed that the users found the mouse more intuitive for single pointer applications, but for the multi-touch interaction the users preferred their three dimensional gesture-based control. The results also show that the users tend to prefer the more familiar interaction methods to completely new ones. [27]

Sreedharan, Zurita and Plimmer studied suitability of three dimensional gesture interaction in virtual reality environment in a user study. Researchers built a simple gesture-based interaction framework around Nintendo Wii controller and Second Life virtual reality. Gestures were abstract, simple pointing gesture and waving gestures, which were mapped to commands yes, no and hey. Users had to navigate through a course and answer questions using the gestures in certain places on the course. The results showed that the users thought that gestural interface was slightly easier to use than keyboard and mouse. The number of right answers with the gestural interface was around the same level as the number of right answers with keyboard and mouse. Generally users preferred gestural interface to keyboard and mouse. [28]

2.2.5 Applicability of gesture-based interaction to digital manufacturing

Digital manufacturing is done in 3D world, which is similar to computer aided design (CAD) environment. 3D world is described in more detail in subchapter 2.4.2. The modeling of systems and process simulation in three dimensions are the main functions of digital manufacturing systems. There are currently no solutions or research using three dimensional gestures to fully support the modeling functions of digital manufacturing. Therefore gesture-based control can be only applied to specific tasks in digital manufacturing applications. Basically every kind of gesture interaction that is used in other three dimensional environments, such as games or virtual reality, can be applied to digital manufacturing environment.

Gestures are used widely in 3D applications with touch screen. Two dimensional gesture-based control is suitable for controlling the user view and for performing simple

manipulations. Also other types of simple commands can be activated through the gestures.

Three dimensional gesture-based interaction is used in games to control the user view and to simulate the real life human actions, such as walking, pressing buttons, swinging tennis racket or dancing. The gesture-based interaction is also used in mimicking other types of actions, for example a bird flying by waving hands up and down. In general the three dimensional gesture-based control in games is used to perform simple tasks, which are mapped to gestures that are easily performed. Rarely used advanced gestures are used in activating commands.

In 3D world, a camera is used to control the user view. Camera control can be applied to gesture-only interface. Other simple enough tasks that could be implemented to interface include simple manipulation tasks; such as resizing a component size or building a simple layout, simulating human model movements and activating simple commands such as start and stop simulation.

Toma, Postelnicu and Antoya studied the applicability of multimodal interaction for 3D modeling. Their study showed that the interaction devices and gesture interaction is currently at the level where the interaction focuses only on activating certain functionalities of the design software. Nevertheless, the interaction interface and corresponding methods should be enhanced so that the user can focus fully on modeling of 3D objects. [29]

Kuo and Wang introduced a motion generation from semantics (MGS) system to articulate the body movement of digital human models. Their study shows that human natural language instructions can be applied to human models in digital manufacturing environment. Their system can facilitate system usability and increase the human motion simulation feasibility. [30] This study can be applied directly to human models in digital manufacturing software. Using real human gestures in human 3D models increases the naturalness of the models and improves their visual representation.

2.2.6 Conclusion

Generally gesture-based interaction is widely accepted as one of the main interaction methods in two dimensional control environments. It is used both in touch pad and touch screen environments. Due to the success of smart phones and tablets the two dimensional gesture-based interactions have been applied to a wide area of applications and a lot of example uses of touch screen in different scenarios can be found from manufacturers' application stores.

Three dimensional gestures and interaction with them is a relatively new application field to the customer market. So far three dimensional gesture-based interfaces have been widely applied only in the video gaming industry. Most of the research concerning three-dimensional gesture-based interaction is done with gaming equipment such as Nintendo Wii or Microsoft Kinect. A lot of research papers end up with the conclusion that currently the technology allowing three dimensional gesture-based interaction is at the level that it can be used effectively also on other application areas than gaming. But

basically the best practices and the standards for using three dimensional gesture-based interaction are still developing and finding their final form.

2.3 Design patterns

Design patterns are an important and integral part of modern software design. In this subchapter the term ‘design pattern’ is introduced and explained and the main benefits of using design patterns are observed. Later in this section the focus is on the presentation patterns that would be suitable for the development of the new interfaces.

In the field of software engineering, design patterns are used as a problem-solving discipline. Software patterns have roots in literate programming, appearing as early as in the 1970s, but they became popular with the success of the object-oriented programming in the 1990s. Gabriel defines a software design pattern in his book A Timeless Way of Hacking as follows: Each pattern is a three-part rule, which expresses a relation between a certain context, a certain system of forces which occurs repeatedly in that context, and a certain software configuration which allows these forces to resolve themselves. [31] Design patterns have been created to offer solutions to problems that programmers face daily in their work. [32] Using design patterns helps designers to isolate the different parts of the software project, which makes the overall system easier to understand and maintain. [33] Also, they create a common vocabulary for communicating designs and promote reuse at design phase. [34]

The Existing design patterns contain years of design experience as experts in the field of software engineering have their work captured in these patterns. By using the knowledge of these patterns developers can save time and efforts as the problem they are facing might be already solved by someone else. [34] Different design patterns are suitable for different situations; therefore selecting the proper design pattern depends on the problem they are trying to solve. Finding the right design pattern for the problem in hand is sometimes a difficult task. A good way to start a design process is to browse through the books written from the area, such as; Design Patterns: Elements of Reusable Object-Oriented Software by Gamma, Helm, Johnson and Vlissides or Design Patterns Explained: A New Perspective on Object-Oriented Design by Shalloway and Trott. An alternative solution is to check online sources such as The Hillside Group’s Patterns Catalog [35] or Portland Pattern Repository [36] for design patterns.

There are patterns that provide general solutions to large-scale design problems as well as patterns that are suitable for solving specific design problems. Architectural patterns are ones that refer to problems in the architectural level of abstraction. By the definition these problems are ones that cover the overall system structure instead of trying to target individual problems. Architectural patterns can be divided into subcategories depending on the problem they are trying to solve. [37] For example, there are architectural patterns for areas of interaction, data modeling, data integration, business modeling, and data presentation.

Patterns exist for providing solutions to problems of HCI. HCI pattern is a general solution to commonly-occurring usability problems in interface design or interaction design. HCI patterns are collected in pattern languages which are complete collections of patterns for certain design problems within a given domain. [38] For example "The Design of Sites" by Van Duyne et al. is a pattern language that helps in designing web sites. [39] Other examples of HCI pattern languages are Tidwell's UI Patterns, Welie's Interaction Design Patterns, Laakso's User Interface Design Patterns and the UPADE language by Engelberg and Seffah. [38]

Interaction devices work on UI level, so in this section the main focus is on architectural patterns for presentation. Model-View-Controller (MVC), Presentation-Abstraction-Control (PAC), multitier architecture, Presenter first and Seeheim model are examples of presentation patterns where the UI is decoupled from other parts of the system. Choosing which pattern to use depends on the properties of the developed system. MVC pattern is largely adapted by Microsoft and therefore suits well for software development in its .NET framework (.NET).

MVC and its derivations Model-View-Presenter (MVP), and Model-View-ViewModel (MVVM) are architectural presentation patterns. They answer to the design problem of how to present the information to the user in different kind of development scenarios. Interfaces use Windows Presentation Foundation (WPF) as presentation technology. With WPF it is recommended to use the MVVM design pattern as it was created specifically to be used together with WPF. [40] As both MVP and MVVM patterns have evolved from the MVC pattern, it is natural to have interest in the evolution process from MVC to MVP to MVVM. Next three subchapters introduce these patterns and presents each of their advantages and most suitable development areas.

2.3.1 Model-View-Controller

Model-View-Controller is a fundamental presentation design pattern that separates the user interface logic from business logic. In the MVC pattern the modeling of domain, the presentation of the data and the actions coming from the user are separated into three classes. These classes are model, view, and controller. The relationship between three the classes of MVC is shown in figure 2.2.

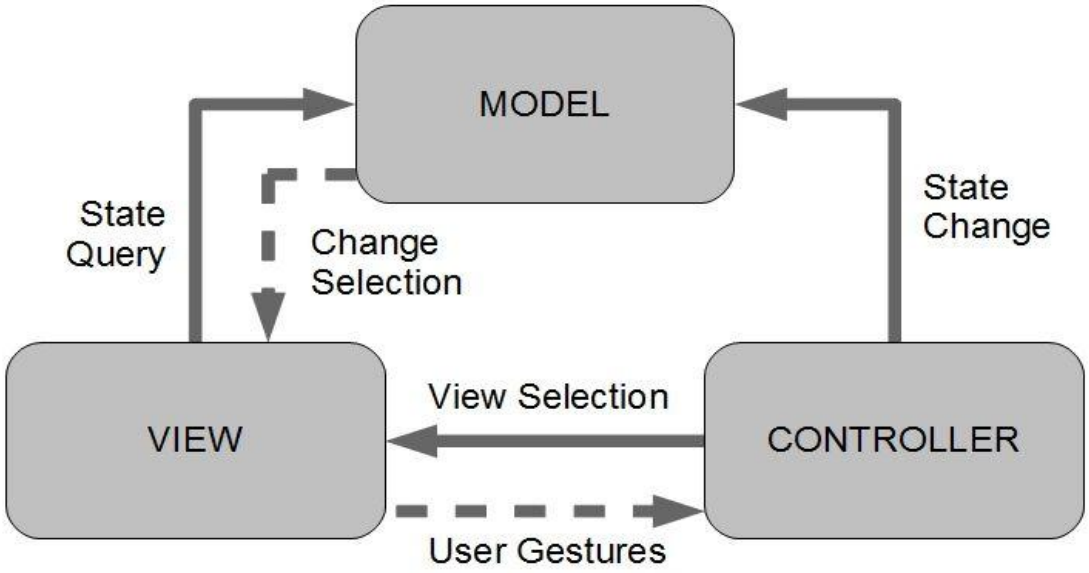


Figure 2.2.*The structural relationship between three classes in MVC.*

Model:

- Encapsulates the application state.
- Exposes application functionality.
- Is responsible for the behavior and data of the application domain.
- Responds to the view's requests about its state, and changes its state according to the controller's instructions.
- Notifies the view about the changes inside the model. [41] [42]

View:

- Is responsible for displaying the information of the model to the user and accepting the user input.
- Requests updates from the model.
- Allows controller to select the view. [41] [42]

Controller:

- Defines application behavior.
- Is responsible for actions concerning updating the view and changing the information of the model.
- Selects the view for response.
- One controller is responsible for one view. [41] [42]

An example of how the MVC pattern works: The view shows the user a form which model defines. The user fills the form and submits it. The data is sent to the controller, and depending on the information, the controller changes the model state according to predefined rules and user submitted information. After changing the model the controller updates the view according to the model's information. Then the user is shown the requested view.

The controller and the view depend on the model, but the model is not dependent on either the view or the controller. This design allows programmers to focus on designing

and building the model, while UI experts can focus on creating the visual presentation of the application. The separation of the view and the controller suits well for web applications as the browser in client-side handles the view, while the server-side controller handles the HyperText Transfer Protocol (HTTP)-requests and actions concerning that. [41]

There can be multiple user interfaces in the MVC system. Each user interface represents part of the application data. According to the MVC pattern, changes in data should automatically afflict all of the user interfaces. Also, any view of application should be able to be modified without changes to related application logic. [31]

2.3.2 Model-View-Presenter

The Model-view-presenter presentation design pattern is derived from the MVC design pattern, and it solves the same decoupling program as the MVC pattern does. The difference between the UI and the data model is done in the MVP pattern by completely isolating the user interface from the business logic. [43] Isolation between the model and the view is done by the presenter. The structure of the MVP design pattern is illustrated in figure 2.3.

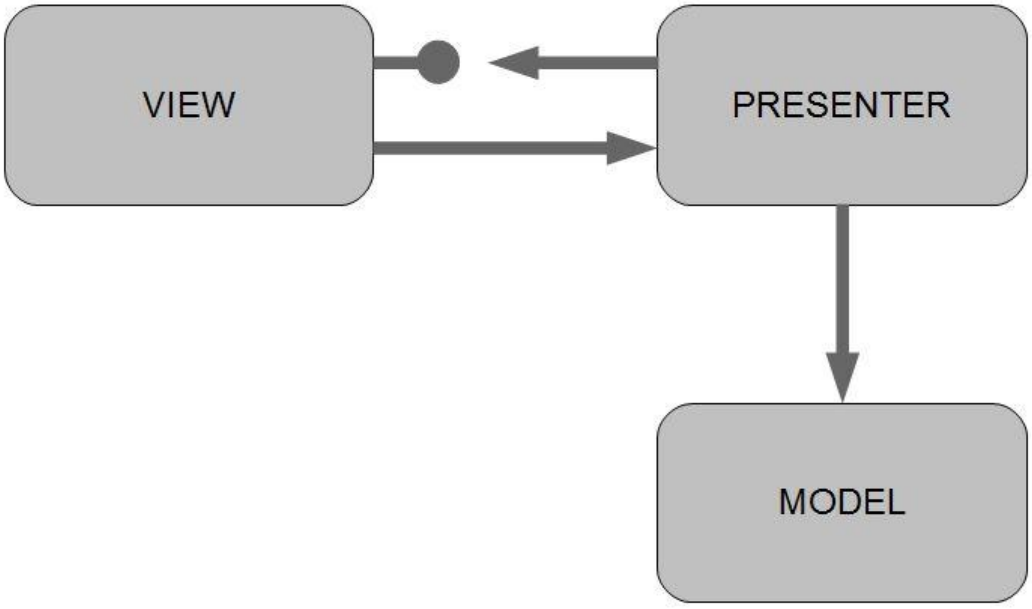


Figure 2.3. *Relationships between the main components in the MVP design pattern.*

The core idea in implementing application according to the MVP model is that the application is split into three main components: Model, View, and Presenter.

- The model component is responsible for encapsulating all the business logic and the data in the application.
- The view component is responsible for displaying the user interface and accepting the user input.
- The presenter component is responsible for orchestrating the use cases of the application. [43]

An example of how the MVP pattern works: When the user clicks the save button on the form, the event handler delegates to the presenter's OnSave method. Then the presenter lets the model do the saving actions. After it is done, the presenter will call back the view through its interface so that the view can display the information that the save has been completed.

As the model component is decoupled from the view component, it is recommended to create interface for all the model's business logic operations and to use the 'Factory method' pattern to return a concrete implementation of the model for the view component to use. This allows the internal changes to the model without changing the view component. [43]

The view represents the presentation layer of the MVP pattern. The view does not perform any business logic, or directly interact with the model. The interaction with model is done by invoking methods on Presenter. The view is fully interchangeable, which means that every concrete view for the presenter must implement an interface which defines all the methods and properties that are required for a view. [43] This enables high customizability of views for UI designers.

The presenter does not have any information about the actual UI layer of the application. It knows that the UI interface exists and it can talk to it, but it does not care about the implementation of that interface. This makes the presenters reusable between different UI technologies. [44]

There are two main variations of the MVP design pattern: Passive view and supervising controller. In passive view, the view and the model are completely isolated from one another. The view contains only the presentation information and no logic at all. The model might raise events, the presenter subscribes to them and updates the view. Passive view has no direct data binding, so instead the presenter uses the view's setter properties to update the view data. The main benefit of passive view is good testability, which comes from the clear separation of the presenter and the view. Using passive view requires more coding than the supervising controller as the data binding is left to the coder's responsibility. [45]

In supervising controller the presenter handles the user input. The view binds directly to the model and the presenter's responsibility is to pass the correct model to view so that the binding can be done. The view's control logic is located in the presenter. The main benefit of supervising controller compared to passive view is the reduced amount of coding, which comes from the use of data binding and at the expense of testability and less encapsulation. [46]

The MVP design pattern in general suits well for the event-driven applications such as Windows client applications built by using Windows Forms (Forms). [40]

2.3.3 Model-View-ViewModel

The Model-View-ViewModel design pattern is evolved from the MVP design pattern. MVVM separates a view from its behavior and state as MVP and MVC patterns do, but in the MVVM pattern, the separating part is a view model, an abstraction of a view,

which contains a view's state and behavior. [47] Relationships between the main parts are illustrated in figure 2.4.

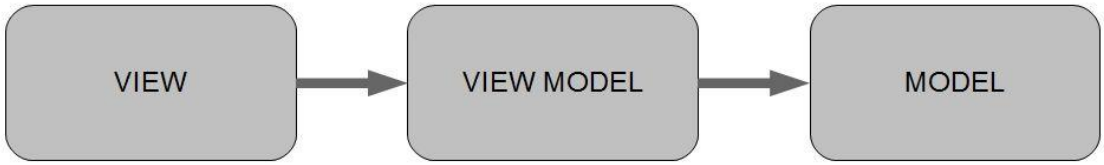


Figure 2.4. *Relationships between the main parts of the MVVM design pattern.*

The core parts and their functions in MVVM:

- Model is responsible for representing the data coming from the database or other services.
- View is responsible for the visual representation of the data and works as an interface to user actions.
- View model ties the view and the model together. It wraps the data from the model and prepares it for the view's use. View model also controls the interactions between the view and other parts of the applications. [48]

An example of how MVVM works: The user submits a form. The view model gets the required information from the bound model and performs actions according to the user submitted data. Next the view model prepares requested information to be shown. After the view model is prepared, the view gets the required data from it and shows the user the new view.

In the MVVM pattern the view is aware of the view model, but the view model is not aware of the view. The view model is aware of the model, but the model is not aware of the existence of the view model. Therefore, the model and the view do not know anything about each other's existence. [48]

The MVVM pattern is tightly integrated in the application development with the Windows Presentation Foundation (WPF) platform as it is designed to standardize a way to leverage core features of the WPF in user-interface creation. [47] The WPF and its new concepts enable the use of the MVVM pattern. The new concepts are:

- WPF Bindings, which connect two properties together.
- WPF Data Templates, which convert non-visual data into visual presentation.
- WPF Commands or Microsoft Expression Blend SDK interactivity behaviors, which pass events from the views to the view models. [48]

These new concepts provide necessary communication methods for passing information between the view and the view model. Communication can be also done by C# events from the view model. WPF bindings are the recommendable way for passing information from the view model to the view, because the use of C# events to trigger changes in the view implies the code behind within the view in registering the event handler. The communications and their directions are illustrated in figure 2.5 [48]

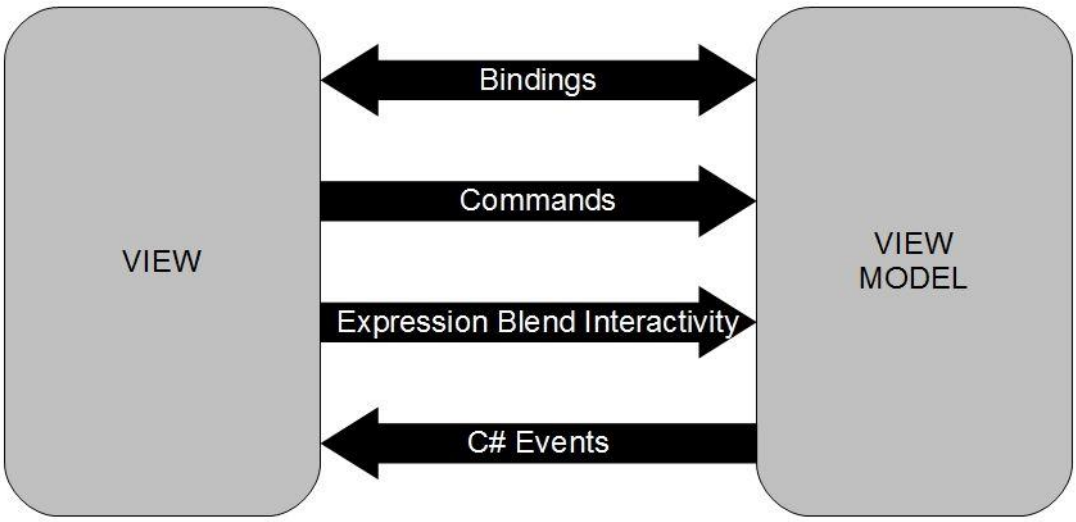


Figure 2.5. *Communications between the view and the view model in the MVVM design pattern.*

Advantages of using the MVVM pattern come from the good separation of concerns, as the view is only responsible for presenting the information, while the non-visual view model is in charge for all of the interactions in the rest of the software, including model and other view models. Other advantages of using MVVM pattern include: flexibility in changing the view while using the same view model, re-use of views and view models in different software, improved separation of UI and development, which helps in testing the software. [48]

2.4 Technologies and tools

This subchapter introduces the main technologies and tools that were used in development of the interfaces between 3D world and interaction devices touch screen, Microsoft Kinect, and 3DConnexion SpacePilot PRO. These technologies are the PACT analysis, 3D world, Component Object Model, .NET framework, and interaction devices.

2.4.1 PACT analysis

The PACT acronym comes from the words people, activities, contexts and technologies. PACT is a framework for designing interactive systems. The PACT analysis can be used both for designing new systems and analyzing existing ones. [49] In this thesis the PACT analysis is used to scope design problems for each of the interaction devices. This subchapter explains what PACT is, how the PACT analysis is used in scoping a design problem and what the contents of the main categories of the PACT are.

In many cases, software systems have not supported the users' needs or requirements optimally. Software design has focused on technology and its possibilities instead of people using them. An essential idea of PACT framework is that the design is human-centered. [49] The core idea of any concept related to human-centered design is

that the users are involved in different phases of the development process. Users are the central part in these concepts. Their involvement lead to more efficient, effective, and safer products and contributed to the acceptance and success of the products. [50]

The PACT analysis splits the designing problem into four main categories: people, activities, contexts and technologies. The development process and the relationship between activities and technologies is as follows: The activities take place in a certain context. They establish requirements for technologies. Technologies in their part offer possibilities that change the nature of activities. Changed activity again results in a new requirement for technologies and so on. The people are in the middle using these activities that are enabled by technologies. [50] The relationship between the elements of PACT is illustrated in figure 2.6.

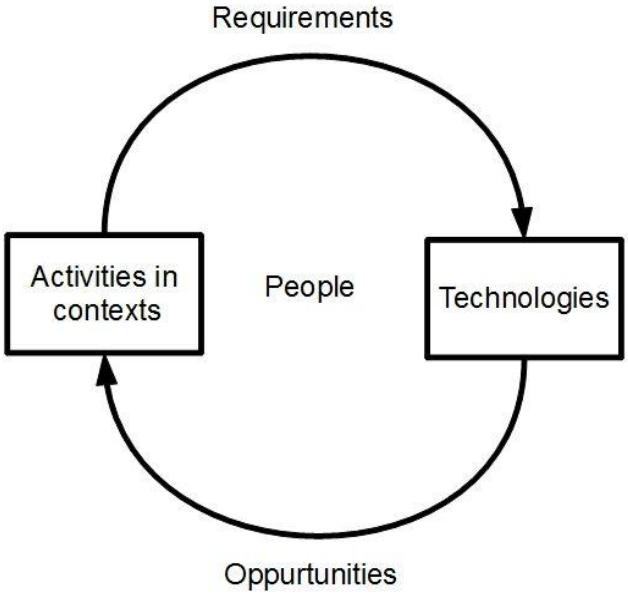


Figure 2.6. *Relationship between the elements of PACT.*

When scoping a design problem with PACT, the designer scopes out as many Ps, As, Cs, and Ts that are possible. After scoping out all of the categories, the developer defines the possible use scenarios by mixing the elements from these categories together. Scenarios are stories about people undertaking activities using technologies in certain contexts. [49] An example of a use scenario could be a young male artist using an electrical drawing board at home to draw a painting. The Young male artist is ‘people’, drawing a painting is an ‘activity’, ‘context’ is his home, and ‘technology’ is the electrical drawing board.

The PACT framework is used to develop conceptual scenarios that technology can support, which are then transformed into specific use cases in the development process. [49] By knowing what the user tries to achieve in certain circumstances, the developer can focus on developing functions that actually serve users’ needs instead of building random functions that somebody might use sometimes. This framework can be easily scaled according to the problem at hand by defining the activities as small as required. This can be done all the way to the single task level activities. [49]

Due to the time limitations and the nature of this thesis, the user tests were limited to people inside the development company. Therefore the actual human-centered development where the users are involved in every phase of the development process was not used, although its methods were used in the definition phase of the interfaces. Also, due to time limitations the scoping of unsuitable use scenarios was left completely outside of this thesis work.

People

People are the users of a new system. People are different in physical characteristics, such as in weight and height. People differ in cognitive skills and personalities, as well as how their five main senses – touch, taste, hearing, sight and smell – function. The physical aspect must be considered when designing a new system. [49]

People differ also in psychological ways. The design should focus on people with poor abilities by providing enough help and making the new system easy to use for everyone. Cultural differences should also be considered when the new system is being designed mainly to be aware of cross-cultural references. [49]

Needs and abilities concerning attention and memory vary a lot between individuals, and they change depending on factors like tiredness and stress. People are better at recognizing things than remembering complicated instructions. This fact should be used in the design process and consequently, the focus should be on the natural way of displaying information and using the natural and familiar input methods. [49]

The mental model of the user is a key concept in this model. Generally every user has a different mental model, and how they interpret surroundings depends on that model. The designer has his or her own conceptual model of the system and it has to be delivered to the user through the system. The design needs to be done so that the people will form a correct and useful mental model of the system. [49]

Activities

Activities are the tasks performed in the system. Activities can vary from simple tasks to very complex, lengthy activities. Various characteristics should be considered when selecting the tasks to the system. The most important characteristic is the purpose of the activity. Other characteristics include the following: [49]

- Temporal aspects: Frequency of task performing, time pressures, whether the task is single or continuous, and response time of the system.
- Co-operation: Whether the task is independent or working with others.
- Complexity: Whether the task is well-defined or vague.
- Safety-criticality: Can the failure result in an injury or a serious accident, designing what happens in error circumstances.
- The nature of activity: Requirements of activity, medium required for activity. [49]

Contexts

Context is the environment where the activity happens. Because the context is closely tied together with activities, they are usually analyzed together at the same time. Three useful types of contexts can be identified. These are the organizational context, the social context and the physical circumstances where the activity takes place. It is important to consider a wide range of contexts in which the activity can be performed. [49]

- Physical environment: The physical place where the activity is performed.
- Social context: Availability of help, privacy issues, and social norms for example.
- Organizational context: Changes in communication and power structures, circumstances under which the activity takes place. [49]

Technologies

Technologies that are used in the system are explained in this part. Interactive systems typically have both hardware and software components. They can perform a variety of functions and usually contain plenty of data or information content. Technologies consist of input devices which are used to enter the data to the system safely and securely, and output devices that display the information in both streaming video output and chunky media such as text and still photographs. The communication between people and devices, and the content delivered through technologies should be considered in this part. [49]

2.4.2 3D world

Digital manufacturing is done in a virtual environment called 3D world. In 3D world, by the definition, everything is illustrated in three dimensions. In this subchapter, the main tasks and the users of 3D world are first introduced shortly. It is followed by introduction of Component Object Model (COM) technology which is used in interaction between interfaces and 3D world. At the last part of this subchapter the camera controls of 3D world are introduced as they are used in each of the interfaces.

All of the digital manufacturing tasks mentioned in subchapter 2.1 can be performed in 3D world. For instance, spot welding can be simulated in 3D world. This is illustrated in figure 2.7. For this thesis, a specific implementation of 3D world is used.

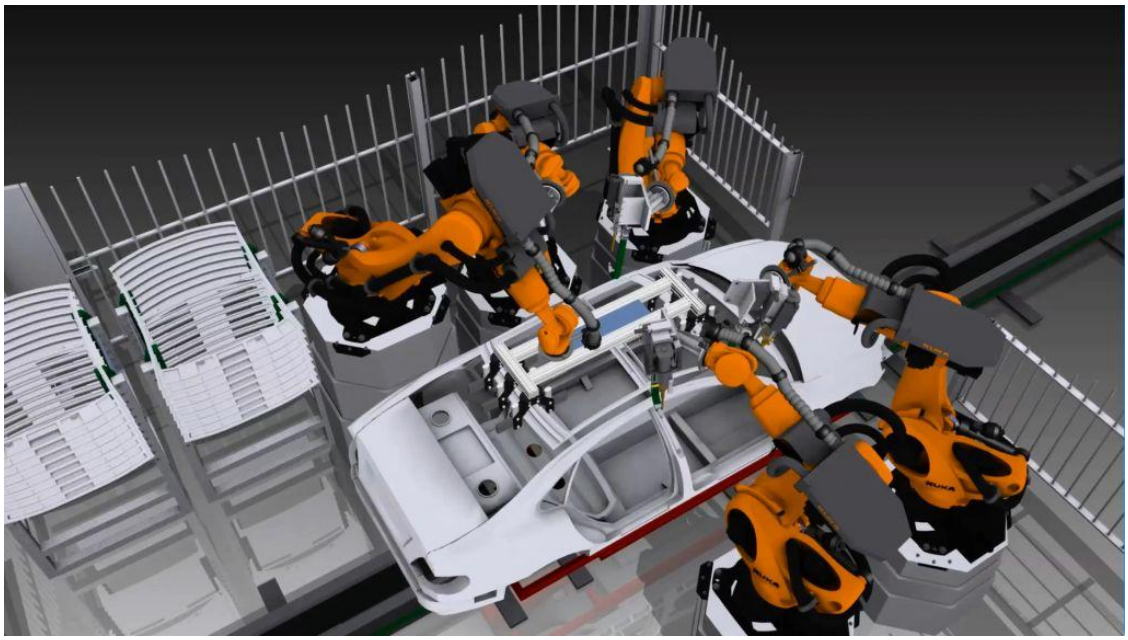


Figure 2.7. *3D world: Robots doing spot welding to a car.*

In 3D world, the user can create the layouts for the facilities by dragging and dropping layout components from the electronic library. Components include robots, conveyors, product feeders, human models and visual objects such as walls and floors. The user can create components by importing existing CAD models or by building completely new components with 3D world tools. Components can be connected to each other and their movement can be adjusted to simulate real life applications. After building a complete layout the manufacturing process can be visually simulated with actual inputs and outputs. Digital manufacturing tasks are performed on created layouts.

People who use 3D world are application engineers, marketing and sales people, and people in deciding positions. Application engineers are responsible for modeling layouts and making new designs. Sales and marketing people use these layouts to visualize the process to possible clients and superiors. The people in deciding positions use 3D world to get the information out of the process and use it to their decision making.

Basic camera controls

The user view of 3D world is called a camera. Position of the camera can be manipulated by the basic camera controls pan, orbit, zoom and fill. In 3D world pan, orbit and zoom are camera modes that are activated by either clicking the corresponding icon in the toolbar with the mouse or by using hotkey modifiers on the keyboard. Fill command is applied once by clicking its icon. Icons are illustrated in figure 2.8. All of the interfaces for the interaction devices mentioned in chapter 2.4.2 use these controls either by directly imitating their functionality or by using them as a base for their actions.

Figure 2.8. *Icons of basic camera controls.*

Pan

The pan function is used to scroll through the 3D window in a two dimensional plane. Panning is done by manipulating width and height coordinates of the camera, while the depth coordinate and the rotation of the camera stay the same. In 3D world panning imitates grabbing the world from the cursor's position and moving the camera in the plane. In panning, when the left mouse button is pushed down and mouse is moved, the window scrolls to the opposite direction to mouse movement. For example, in pan mode when the mouse is moved to the left, the window is scrolled to the right.

Orbit

The orbit function is used to move the camera around its center of interest. Orbiting is done by changing the tilt and the pan rotation around the camera's center of interest. The distance to the center of interest stays the same while orbiting is done. Orbiting imitates grabbing and rotating camera on the plane of an imaginary ball that is around the camera's center of interest. With orbit mode on, when the left mouse button is pushed down and the mouse is moved, the camera rotates the center of interest horizontally to the direction of the mouse movement and vertically opposite to the direction of the mouse movement. For instance, in orbit mode when the mouse is moved up and left, the camera rotates downwards and left around the center of interest.

Zoom

The zoom function is used to zoom in and out on camera's center of interest in 3D world. Zooming in is done by moving the camera closer to its center of interest in order to obtain a closer view in 3D window. Zooming out is done by moving the camera backwards from its center of interest in order to obtain a more distant view in 3D window. Zooming is done by changing the depth coordinate of the camera, while width and height coordinates and the rotation of the camera stay the same. In zoom mode, when the mouse is moved up the camera zooms in. Respectively when the mouse is moved down, camera zooms out.

Fill

The fill function is used to center the camera so that the whole layout fits in the screen. Filling is done by checking positions of the layout components and moving the center of interest in the middle of the components. After that the gaze length of the camera is set so that every component fits in the screen. Rotation of the camera is not affected by fill command.

2.4.3 Component Object Model

Interaction between 3D world and the new interfaces was done through component object model interface of 3D world. COM is an object-oriented, distributed, platform-independent technology for creating binary system components that are able to interact. COM was created by Microsoft in 1993 and it is a foundation technology for many other technologies such as ActiveX and Microsoft OLE. [51]

COM is a binary standard, which specifies an object model and programming requirements that enable COM objects to interact with other objects. By definition, a binary standard is a standard that applies after a program has been translated to binary machine code. The source language of COM components may vary and the structure can be very dissimilar to each other. Also the location of COM components can be within a single process, in other processes, or in remote computers. The only language requirement for COM is that the language supports structures of pointers, and that the functions can be called through pointers. In COM object the access to the object's data is achieved exclusively through one or more sets of related functions, which are called interfaces. The functions of interfaces are called methods. COM requires that the only way to access the methods of an interface is through a pointer to the interface. [52]

In addition to providing a basic binary standard, COM provides the set of required functions for all components. It defines certain basic interfaces, which provide functions common to all COM-based technologies. COM also defines the cooperation of objects in distributed environment and it helps providing the system and component integrity with its added security features. [52]

COM interaction happens between a server and a client. A COM client is any code or object that gets a pointer to a COM server, and uses its services by calling the methods of its interface. A COM server is any object that provides these services to clients. The services of a COM server are in a form that any COM client that is able to get a pointer to any interface of the COM server can call any service of a COM server. [53]

The main types of COM servers are in-process and out-of-process servers. In-process servers are implemented in a dynamic linked library (DLL), while the out-of-process servers are implemented in an executable file (EXE). Out-of-process servers can be located in either a local or a remote computer. In-process servers are able to run in surrogate EXE process. With the help of this mechanism they can run the process on a remote computer. [53]

2.4.4 .NET framework

The target software framework for the new interfaces is .NET framework (.NET). .NET is a software network developed by Microsoft. It is included in the Microsoft Windows operating system as an integral component of it. The development of .NET began in the late 1990s and it was originally known as next generation Windows services (NGWS). The term NGWS was used for Microsoft's plan for producing an Internet-based platform. [54] In this subchapter the .NET basics and the .NET technologies that are used in the development of the interfaces are introduced. The technologies in use are Windows Presentation Foundation (WPF) and C# programming language.

.NET framework's component model is based on the COM component model. Components that are written and compiled to the common language runtime (CLR) environment are called modules. These modules are similar to corresponding COM modules and as in COM, the modules in .NET are pre-compiled in DLL or EXE files. The type library, which contains the definitions of classes that are exposed by the COM server, is roughly equivalent to manifest of .NET application. This similarity allows interoperability between the COM component and the .NET application. The type library of the COM component can be added into .NET application's references. This enables creating and instantiating objects from the classes exposed by the COM component. .NET application then manipulates the COM object's properties, methods and events. COM components run in their own process thread, which is located outside of the .NET's process space. This means that Windows is independently responsible for managing the COM component's resource requirements and passing of the messages between the COM component and its consumer. [55]

First beta version of .NET was launched in 2000 and the latest official release of framework in May 2012 is numbered 4.0. [56] The .NET framework is designed to:

- Provide a consistent object-oriented programming environment in varying code storage and code execution locations.
- Minimize the software deployment and versioning conflicts.
- Provide a safe execution of code, regardless of the origin.
- Eliminate the performance problems of scripted or interpreted environments.
- Bring consistency to the application development across different types of applications, for example Windows phone applications and Windows desktop applications.
- Ensure that the code based on the .NET Framework can integrate with any other code. [57]

The two main components of .NET are the CLR and the .NET framework class library. CLR is responsible for managing the code while executing it. It provides core services such as thread management, memory management and remoting, while being responsible for strict type security and other forms of code accuracy that promote robustness and security. The class library is an object-oriented, comprehensive collection

of reusable types for application development. The main components and features of the .NET framework are illustrated in figure 2.9. [57]

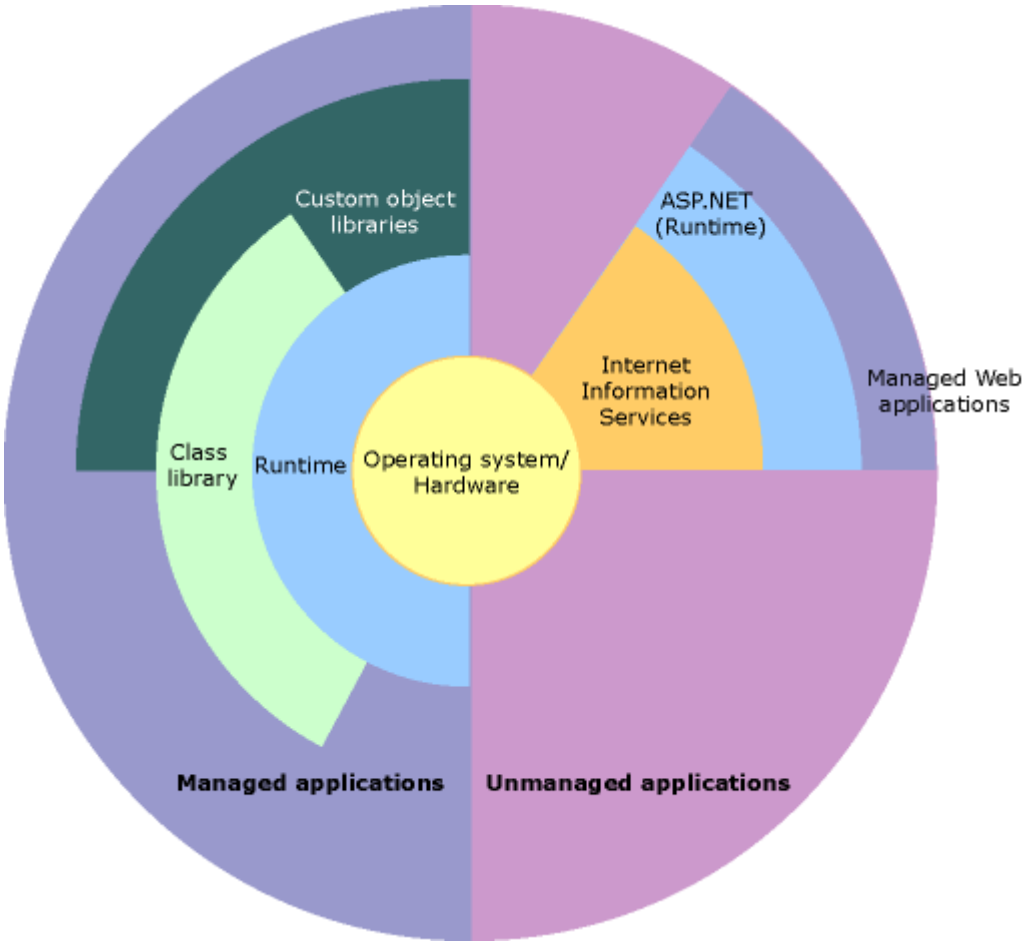


Figure 2.9. *Main components and features of the .NET framework. [57]*

The Main features of the CLR are:

- Management of code execution, memory, thread execution, compilation, code safety verification and other system services.
- Enforcing of code access security, which means that code can only access the resources that are available for it and nothing else.
- Implementing of a strict type-and-code-verification infrastructure called common type system (CTS).
- Providing varying levels of trust for managed components depending on different factors. Components in the same active application may have different trust levels.
- Providing enhanced performance with features such as just-in-time (JIT) compiling and memory manager.
- Making features of the .NET framework available to all supported languages.
- Being responsible of handling interoperability of managed and unmanaged code.
- Handling the automatic memory management.

- Possibility of hosting runtime by server-side applications. [57]

.NET Framework Class library:

- Is a collection of reusable types which tightly integrate with the CLR.
- Is Object-oriented.
- Enables seamless integration of third party code with classes in .NET Framework.
- Enables accomplishing of common programming tasks such as data collection, string management, database connectivity and data access.
- Includes types that support variety of specialized development scenarios such as ASP.NET web applications or WPF applications. [57]

Windows Presentation Foundation

Windows Presentation Foundation is the new generation for developing Windows client applications. WPF was created to help the programming of rich applications that were difficult or impossible to build with previous .NET presentation technology Windows Forms. Often those applications required several other technologies and the integration of various technologies was a difficult task. WPF combines most commonly used presentation technologies, such as Portable Document Format (PDF) and 3D graphics, under one technology and they can be now used flawlessly together in the same application. [58] Because of this WPF can be used to build a variety of Windows applications, ranging from traditional console applications to media players. [40] WPF was introduced in the .NET framework version 3.0. [56] WPF is used as the presentation system for new interfaces.

WPF separates appearance from behavior. The appearance specification is done generally in extensible application markup language (XAML). The behavior implementation is done in managed programming language like Visual Basic (VB) or C#. [59] XAML is a markup language based on Extensible Markup Language (XML). It follows or expands XML structural rules. The CLR enables run time execution of XAML. XAML is not one of the common languages, but instead XAML types are mapped to CLR types to instantiate a run time representation when the XAML for WPF is parsed. [60]

The core of WPF is a resolution-independent and vector-based rendering engine which uses the features of the latest graphics hardware. [40] The primary programming model of WPF is exposed through managed code. Major code portions of WPF are PresentationFramework, PresentationCore and milcore. Milcore is the only unmanaged component of these. It is written in unmanaged code because of its tight integration with a DirectX engine. The DirectX engine is responsible for all display parts of the WPF technology, allowing for efficient hardware and software rendering. The composition engine in milcore gives the extra performance when needed by giving up the advantages

of the CLR to gain performance. The major components of WPF are illustrated in figure 2.10. [61]

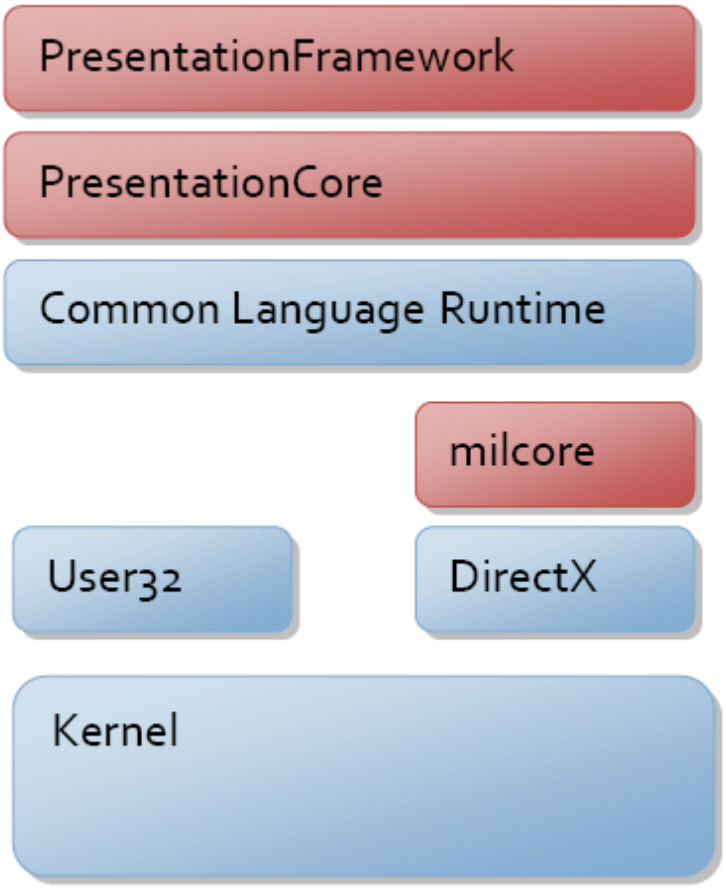


Figure 2.10. *The Major components of WPF. [61]*

C#

C# is a programming language designed by Microsoft. It is designed for programming applications in .NET framework. [62] C# is one of managed languages. Managed language means that the code targets .NET framework and is executed in CLR. [63] C# is based on languages such as C, C++ and Java. C# is object-oriented language, as C++ and Java are, but includes support for component-oriented programming. C# is designed for building durable and robust applications. [62] Following features support this function:

- Exception handling for structured and extensible approach to error detection and recovery.
- Garbage collection which automatically reclaims memory occupied by unused objects.
- Type-safe design which removes a lot of common programming errors such as indexing arrays beyond their bounds. [62]

C# has a unified type-system, where every object inherits from single root object type. All types share a set of common operations, and values of any type can be stored, transported, and operated upon in a consistent manner. C# supports also user-defined reference types and value types, which allow dynamic resource allocation and in-line storage of lightweight structures. In the development of C#, a lot of effort has been put to versioning, for example by including separate virtual and override modifiers and by having explicit interface member declarations. [62]

The C# compile-time process is as follows: C# compiler compiles the source files into an intermediate language (IL) that conforms to the common language infrastructure (CLI) specification. Then, the code and resources of IL are stored in assembly, which is an executable file on the disk, usually with an .exe or .dll extension. Metadata of the application is stored in a manifest which is located in assembly. The manifest holds the information about the assembly's types, culture, version and security requirements. [63]

In the execution process of C# the created assembly is loaded into the CLR. First the CLR checks the information in the manifest and might take actions based on it. Then if the security requirements are fulfilled, the CLR does JIT compilation, which converts IL code to the native machine instructions. [63] Figure 2.11 illustrates the compile-time and the run-time processes of the C# program.

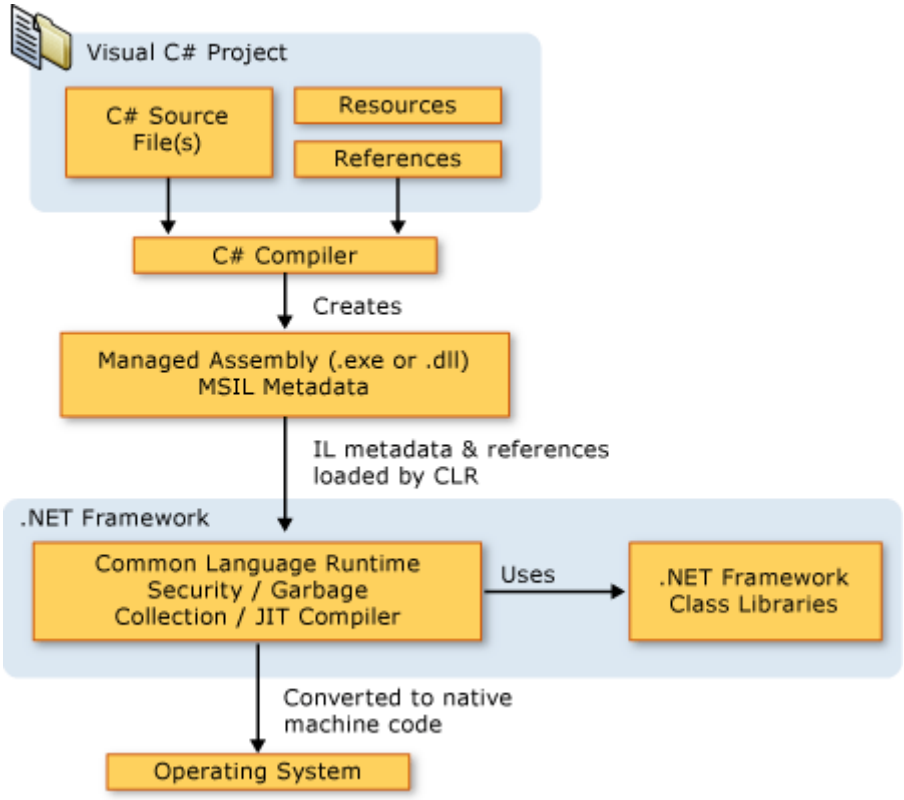


Figure 2.11. *The compile-time and run-time processes of a C# program. [63]*

2.4.5 Interaction devices

The interaction devices that are used for this thesis work are Microsoft Kinect, touch screen and 3DConnexion SpacePilot PRO. In this subchapter the main functions of the mentioned technologies are shortly introduced.

Microsoft Kinect

Microsoft Kinect is a motion sensor, originally developed for Microsoft Xbox 360 gaming console as a controller. [64] Kinect was launched in the United States November 4, 2010. Only three days after the launch was hacked and the open source drivers for Kinect development were released. [65] On June 16, 2011, in response to open source drivers Microsoft released a beta version of its own SDK for non-commercial applications only. [66] Kinect for Windows was released February, 1, 2012, together with the official release of the Kinect SDK for commercial applications. [67]

Kinect sensor provides the full-body 3D motion capture, voice recognition, and facial recognition capabilities. Kinect sensor incorporates multiple sensors. The sensors are a color red-green-blue (RGB) camera, three-axis accelometer, depth-sensor, and four-microphone array. The sensors are located in horizontal bar which stands on a small base connected with motorized pivot. [64] Kinect sensor and its main components are illustrated in figure 2.12.

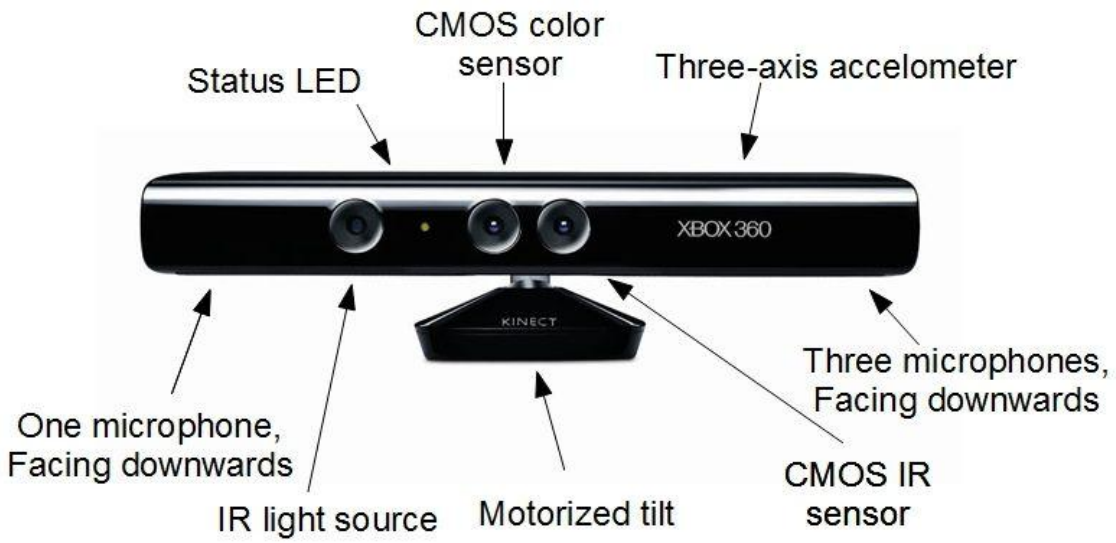


Figure 2.12: *Kinect Sensor and its main components.*

Depth sensor of Kinect consists of the infrared (IR) projector and IR camera, a monochrome complementary metaloxide semiconductor (CMOS) sensor. The depth sensing technology is based on structural light principle. The IR projector is an IR laser that passes through a diffraction granting and turns into a set of IR dots. The IR camera sees the IR dots. The relative geometry between the IR projector and IR camera and the

IR dot pattern are known. By matching these together the 3D image is created. Depth image can be created in any lighting conditions. [64]

Kinect tracking area is in front of the sensor. The field of view is determined by depth camera settings. In default mode the Kinect is capable of seeing people standing between 0.8 m and 4 m, but as the users need to be able to use their arms, the practical distance of default mode is 1.2 m to 3.5 m. The horizontal field of view in default mode is illustrated in figure 2.13. The practical distance can be seen from figure 2.14, in which the vertical field of view is illustrated. In near mode the Kinect is able to detect standing people from 0.4 m to 3 m, with a corresponding practical range of 0.8 m to 2.5m. [68]

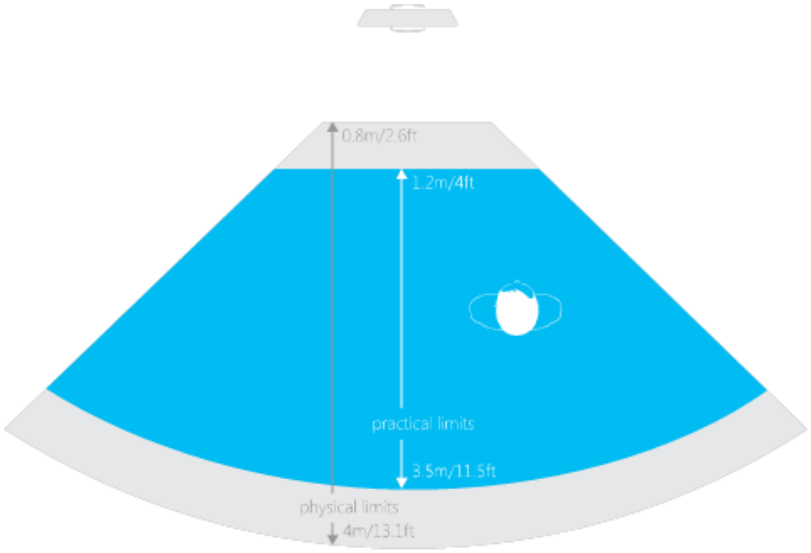


Figure 2.13. *Horizontal field of view of Kinect [68]*

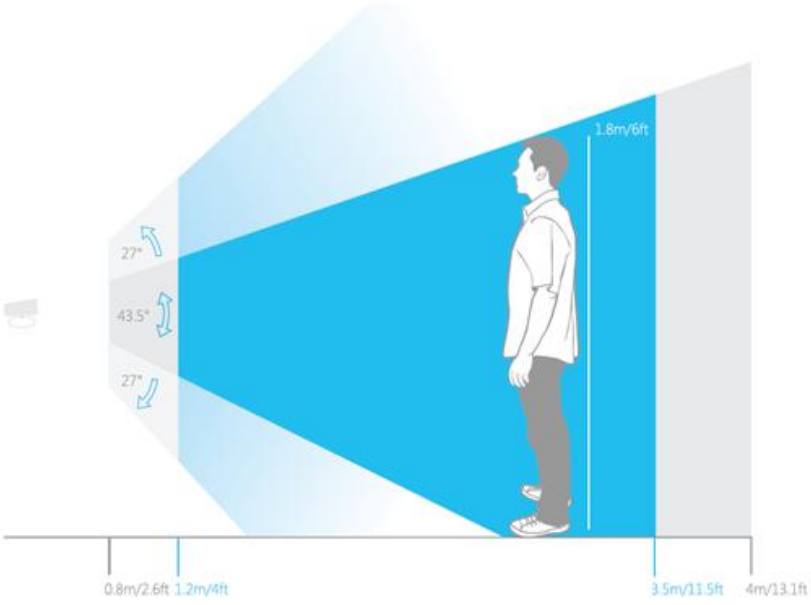


Figure 2.14.*Vertical field of view of Kinect [68]*

The voice recognition technology first makes an audio profile of the room which maps out the reflectivity of the room. The voice recognition is done by beam forming, which allows Kinect to focus on specific points of the room to listen. While listening, the audio processor is doing multichannel echo cancellation based on the echo profile of the room. With this technology, Kinect can identify the source of the voice in the room and use the acoustical model to recognize the voice commands given. [69]

The facial recognition technology uses the RGB camera image to extract nine main objects from face, meaning eyes, nose, etc. Images are then filtered to remove illumination variations and assigned some code. Then Kinect determines the pose, which means whether the subject is looking straight to the camera, left or right. Next, the system matches results with its database in order to find the best match. The facial recognition technology requires proper lighting conditions because technology is based on the color image which changes in different lights. The matching technology interprets the expression falsely if the lighting is poor. [70]

Kinect for Windows SDK provides the access to audio data streamed by the audio stream, and to color image data and depth image data streamed by the color and depth streams. All data is accessed through the natural user interface (NUI) of the Kinect for Windows application programming interface (API). Figure 2.15 illustrates the position of the NUI in Kinect Architecture. [71]

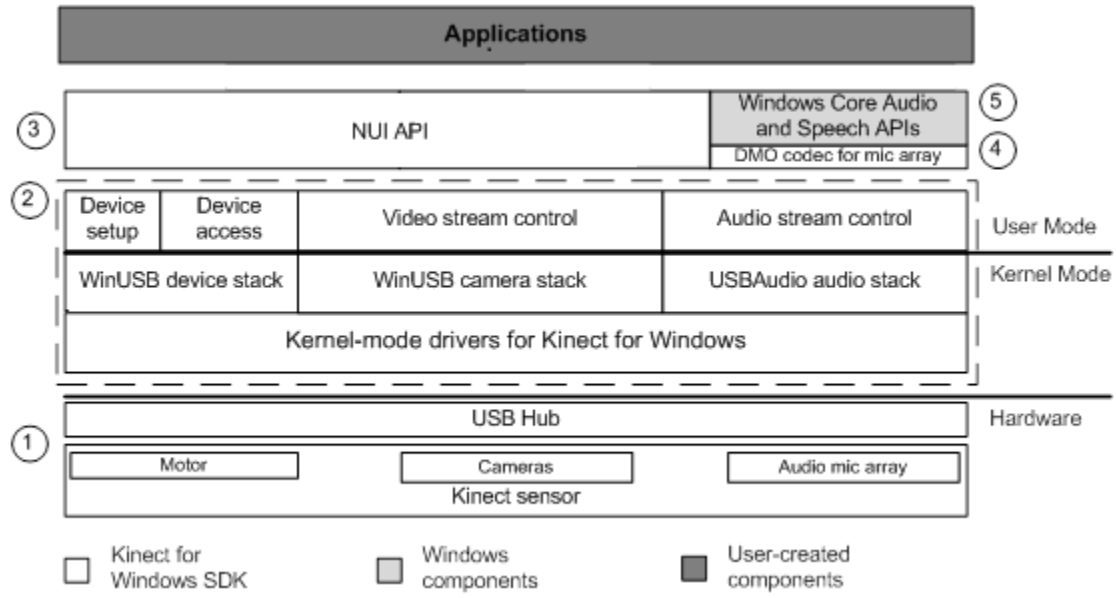


***Figure 2.15. A**rchitecture of Kinect for Windows SDK [71]*

In addition to sensor data, the SDK provides tracking information of six users in the field. More accurate tracking information of maximum of two people is provided in form of a skeleton stream. Skeletal information is created from the depth stream, which is converted to the skeleton joints of the human body. Skeletal tracking is designed for recognizing users facing the sensor. The tracking is not accurate if the user is standing sideways. Skeleton stream provides the tracking information of twenty skeletal joints in

the default mode and ten upper body joints in the seated mode. [68] The tracked joints in the default mode are illustrated in figure 2.16. The SDK is integrated tightly with Microsoft Speech APIs which enable developer to use the speech recognition technology. The SDK is also integrated with the Face Tracking SDK. [71]

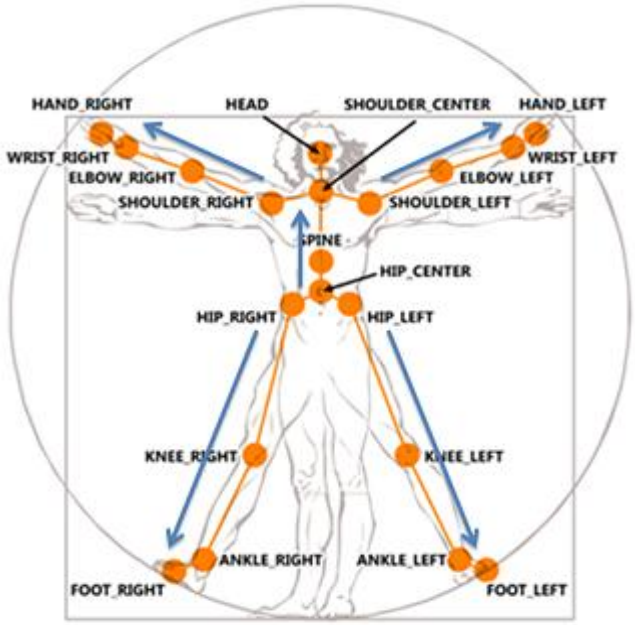


Figure 2.16. *Joints and their positions in default mode in Kinect skeleton stream. [68]*

Touch screen technologies

A touch screen is a display that can be used as an input device at the same time. Interaction with the computer is usually done by touching the screen with a finger or stylus. The touch screen was introduced in the 1960s, [72] but just lately the technological advances and the success of smart phones and tablets in the early 2000s have created a big demand for touch screen devices. [73] The most commonly used technologies in touch screen are resistive, capacitive, infrared and surface acoustic wave (SAW) technologies. [74]

An essential part of a modern touch screen device is that it supports touching to multiple places simultaneously. Multi-touch is a requirement for gesture-based control as the basic gestures such as pinching and rotating require at least two fingers to be performed correctly. Multi-touch enables using the channels that allow more actions to be mapped to gestures.

In resistive touch screen there are two layers of electronically conductive material, which are separated by transparent insulating dots. When the screen is pressed, the two layers touch and the electric circuit is complete. This is illustrated in figure 2.17. The position of the touch changes the voltage of the circuit and the screen's controller measures the coordinates based on that.

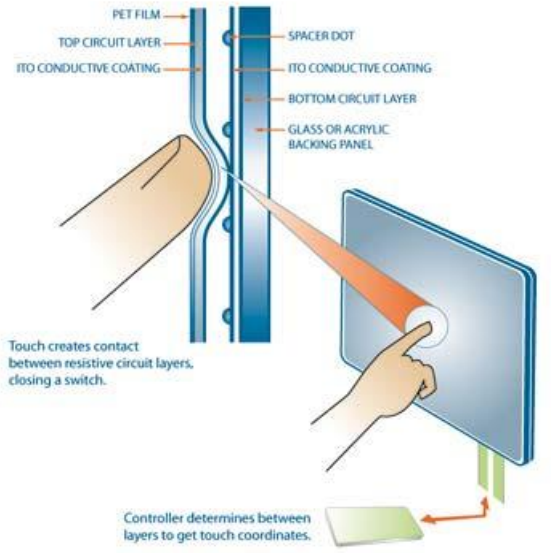


Figure 2.17. *Resistive touch screen technology: When touched, two electronically conductive layers connect and the electric circuit is formed. The position is detected by the changes in current of the electric circuit. [75]*

Resistive screens use physical process and that's why they can be pressed with any kind of pointer, for example with a stylus or a hand with gloves. This makes them very suitable for industrial environment or outdoors. These screens are also very durable, and because of that they are widely used in different kinds of sale terminals. The main negative property of the resistive touch screen is that it lets only 75% of light through, which makes them unsuitable for systems that use high-definition screens as their main feature. [74]

The capacitive touch screen is usually formed by two parallel indium tin oxide (TIO) panels which are separated from the sensor glass. TIO is a conductive material and the two panels have a small current between them. [74] When the user touches a cover panel of a capacitive touch screen it causes some of the current to draw away from the circuit by the body's natural capacitance. Touch screen controller notices this change in the current and determines the position of the touch. [76] Functionality of a capacitive touch screen is illustrated in figure 2.18.

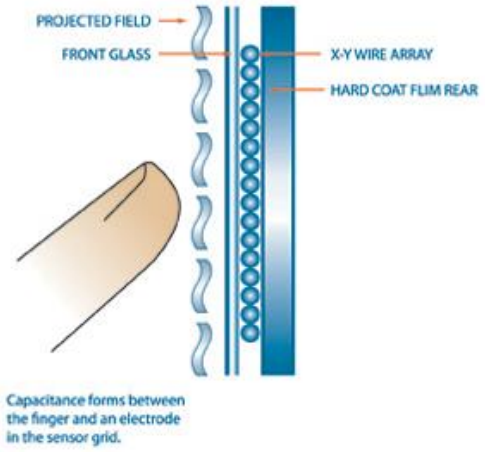


Figure 2.18. *Capacitive touch screen: Closing finger draws some current away from the electric circuit. Touch screen controller notices the change in current and determines the position of the touch based on that. [75]*

Capacitive screens are more sensitive than the resistive ones, since the design does not require any physical contact with the panel. But the design also makes it impossible to use a stylus or any other pointer without electrostatic charge. Capacitive technology lets 90% of the light through. [74] It is used widely in modern handheld devices.

In infrared touch screen technology the screen is surrounded by infrared light-emitting-diodes (LED). Every LED has a matching photodetector installed in a grid across the screen area. When the user touches the screen the infrared beam is broken. The coordinates of the touch are derived by photodetectors whose beams were broken. Functionality of the infrared touch screen is illustrated in figure 2.19.

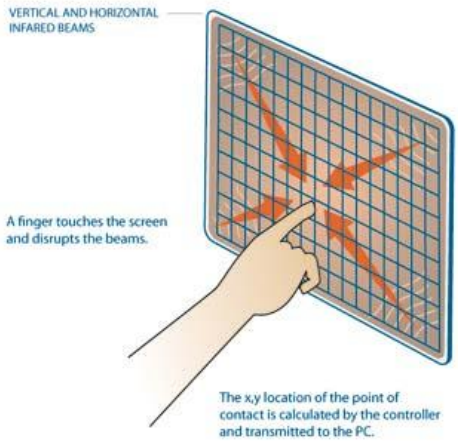


Figure 2.19. *Infrared touch screen: A finger disrupts infrared beams and the position of the touch is decided by the photodetectors whose beams were broken. [75]*

Infrared layer is positioned just above the screen and it does not require a physical contact to get the input. Because infrared touch screen does not require actual touch, the surface of a touch screen can be made from any transparent material. This allows infrared touch screen to be very durable. Infrared systems can be operated with any kind of

pointers, which makes them suitable for systems located outdoors. [20] Deficit of optical technology is that it is sensitive to contaminants, which may cause miss interpretations. This technology offers 100% light transmission with no additional layers covering the screen. Infrared technology is more expensive than others and is usually used in large screens and military applications. [76]

Surface acoustic wave touch screens use beams of ultrasonic waves to form a grid above the display screen. [76] When the user touches the screen the waves in the grid are interrupted. The refraction of waves is measured electronically by X and Y sensors and is then converted to coordinates. Functionality of the SAW technology is illustrated in figure 2.20.

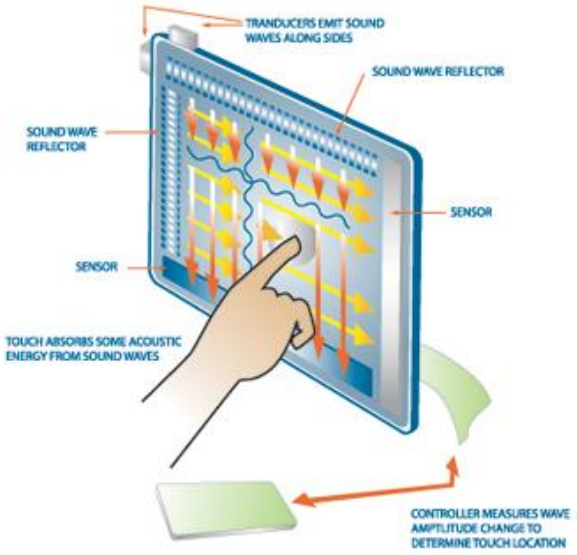


Figure 2.20. *SAW touch screen: A finger disrupts acoustic waves and the position is measured by sensors. [75]*

As the technology concept for SAW with sound is similar to infrared systems with light, the advantages are also similar. It is durable, can be used with any kind of pointer and it has 100% light transmission. SAW screens are more easily affected by water and dirt than infrared screen, which limits their outside usage. The requirement of ultrasonic transducers makes the technology unsuitable for mobile devices. This technology is also rather expensive and it is used often in public information kiosks and automatic teller machines (ATM). [74]

3DConnexion SpacePilot PRO

The main products of 3DConnexion are known as movement controllers or 3D mice. These controllers are used for accelerating the productivity for 3D manipulation and navigation tasks. The main features of 3DConnexion SpacePilot PRO are introduced in this subchapter. SpacePilot Pro can be seen in figure 2.21.

Figure 2.21. *3DConnexion SpacePilot PRO [77]*

3DConnexion 3D mice are meant to be used together with the keyboard and traditional mouse. In the optimal setup the 3D mouse is placed on one side of the keyboard and the regular mouse on the other, and it is used with the hand that is not controlling the regular mouse. The 3D mouse desktop setup can be seen in figure 2.22. 3D mice are intended to be used in navigation and basic 3D manipulation tasks while tasks such as selections, advanced manipulation tasks and typing are performed with keyboard and traditional mouse. [78]

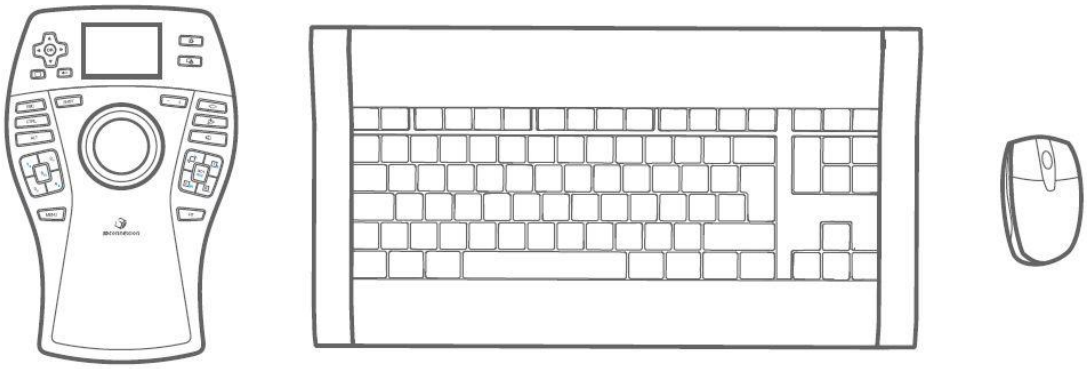


Figure 2.22. *SpacePilot PRO desktop setup. [78]*

3D mice are designed to be user friendly and easily integrated in the design process. According to the company, the use of 3D mouse provides an intuitive and a more natural way of interacting with a computer in 3D environments. The main feature of 3DConnexion space mice is the central controller cap, which every 3D mouse has. Supporting functions depend on the model of a 3D mouse. Supporting functions include

liquid crystal display (LCD) display, programmable macro buttons, keyboard buttons and other buttons. [79] SpacePilot PRO is the product with most supporting functions in 3D mice product family.

The controller cap in SpacePilot PRO is a six degrees-of-freedom (6DoF) sensor. The cap reacts to pressure. It can be both translated and rotated in horizontal, vertical and depth directions. The cap functions are illustrated in figure 2.23. [77] Translations and rotations can be performed at the same time in three dimensions. The cap can return data ranging from -32767 to 32768 on each axis, but the usable range is just a fraction of it. It is not practical in most applications to use enough force or torque to reach the large numbers. By using the full range, the low end precision is lost. The usable range of data tends to be around -350 to 350. [80]

Supporting functions of SpacePilot PRO include buttons for keyboard modifiers, programmable function keys, navigation setting keys, 3D Mouse keys, LCD workflow display and keys for controlling the display. [78]

Figure 2.23. *Cap functions of SpacePilot PRO: Cap can be both translated and rotated in 3 dimensions. [80]*

Using 3D mouse in applications requires an interface for it. 3D mouse input can be accessed in two main ways: either by using the 3DxWare SDK or by using the raw input of 3D mouse.

SDK:

- C++ API.
- Requires 3DxWare driver.
- Well documented.

Raw input:

- Application gets the data directly from the device.
- Does not require driver.
- Can be used in programming with various languages.
- Poorly documented with 3D mouse.

3D mice are typically used in mechanical engineering CAD software, but also applications in the area of architecture and construction, media and entertainment, gaming and geo information have support for 3D mice. However, it is possible to integrate 3D mouse to any kind of program through the programming interfaces. [78]

3 INTERFACES AND THEIR IMPLEMENTATIONS

The main research question of the thesis was: How can the interaction devices – touch screen, Kinect, and SpacePilot PRO – be used in interacting with 3D world? Literature and researches form the basis of answering this question, but information about the actual possibilities and usefulness of these devices could have only been found out by using the actual devices in interacting with 3D world. To be able to interact with 3D world, the interfaces for each of the devices had to be built. The amount of information in previous researches and implementations varied a lot between these devices. Due to this, different approaches for building interface for each of the devices was taken. Generally any of these devices cannot completely replace the functionality of keyboard and mouse in digital manufacturing environments. Therefore the focus in this section is on the activities that are the most suitable for each of these devices. The development process involved building the software interface between the output of the devices and 3D world as well as building the user interface to access the interaction devices.

Touch screens have been on the market for two decades and a lot of information about interacting with touch screen in three dimensional applications exists. Therefore, there are various three dimensional applications that use touch screens in interaction. The approach for building this interface was to first find out how these applications are typically controlled with touch screen devices. Various applications were tested on two different touch screen devices. The PACT analysis was used to define the possible use scenarios. After scoping out the design problem the most suitable functions were implemented to the interface.

Microsoft Kinect is a relatively new interaction device. Therefore the amount of previous researches and applications using the Kinect in three dimensional applications was limited. The approach for building this interface was explorative. Information about using the Kinect was gathered during the development of the interface. The interface was built by trying out different solutions for gesture-based control. After a version of the interface was built, it was analyzed and the analysis was used to improve the next version.

3D mice are commonly used in CAD environments, which are similar to 3D world. Therefore the approach for this interface was to enable similar functionality to 3D world that the most common CAD environments have. SpacePilot PRO is the product with the highest amount of supportive functions in the 3D mouse family, but the interface was wanted to be usable with any 3D mouse model. Therefore the focus in the development

was on the basic functions performed with the controller cap. The documentation and demos of 3DConnexion SDK were used as a basis for this interface.

In this chapter each of these interfaces are firstly briefly introduced. After that the development process of each interface is described. Next, the implemented functions are introduced in more detail and the decision making behind the implementation is discussed. Future work is discussed in the last section of each of the subchapters.

3.1 Touch screen interface

Nowadays, as the touch screens are getting increasingly more common everywhere, having suitable interface for it is more a requirement than additional benefit for any software, be it desktop software or mobile software. Both mouse cursor and keyboard can be controlled through touch screen, which makes MTS either a partial replacement to keyboard and mouse control or an additional input method to be used aside keyboard and mouse.

This section is about building the touch screen interface for 3D world. The first subchapter is about testing different 3D applications with Apple iPad tablet device and Google Android smart phone. This was done in order to find out how 3D applications are typically controlled with mobile touch screen devices. In the next subchapter the PACT analysis is used to define the most suitable use cases and scenarios for the interface. After that the development process of interface and technology selections are briefly discussed. Later subchapters focus on implementation of the touch screen actions. In the last subchapter the future work is discussed.

3.1.1 Controlling 3D applications with mobile touch screen devices

This subchapter is about testing different types of 3D applications in mobile touch screen devices. Testing devices were Apple iPad and Google Android smart phone Samsung Galaxy S2. Testing was done in order to find out how 3D applications are usually controlled with touch screen devices.

For both devices several applications were tested and the type of controls was observed. The suitability of controls for different types of applications is discussed in the last conclusion part of this subchapter. The knowledge gathered from these tests was used in creating the interface for MTS devices.

iPad

The tested applications were 123D Sculpt, Google Earth, MetalStorm: Wingman, Call of Mini: Zombies, Structural View, Bentley Navigator for iPad, TapTapBlocks, BIMx and Star Legends. The first item in the bulleted list is a description of the application, and the rest of the bullet points introduce the application’s control methods.

123D Sculpt

- Manipulation of 3D models with various tools.
- One-finger drag is used for manipulations on object.
- One-finger drag is used for orbiting outside the object.
- Two-finger drag is used for panning.
- Pinching gesture is used for zooming.

Google Earth

- Globe software.
- Panning on Earth is done by dragging with one finger.
- Zooming is done with the pinching gesture.
- Quick zooming on point is done by double tapping with one finger.
- Rotating the Earth is done with the rotate gesture.
- Rotating the user view is done by dragging with two fingers.

MetalStorm: Wingman

- Fighter flying simulator.
- Turning the plane is done by tilting the device.
- Accelerating and shooting is done by using software buttons.
- Special movements are done by using swipe gestures.

Call of Mini: Zombies

- 3rd person shooter game.
- Movement is controlled with software joystick.
- Rotating the view is done by dragging finger anywhere else on the screen than on the joystick icon.
- Shooting is done by pushing the software button.

Structural View

- CAD model viewer.
- Orbiting is done by dragging with one finger.
- Panning is done by dragging with two fingers.
- Zooming is done with the pinching gesture.

Bentley Navigator for the iPad

- Navigate in 3D models.
- Panning on a map is done by dragging with one finger.
- Zooming both on a map and in the 3D mode is done with the pinching gesture.
- Focus on a hotspot in the map is done by tapping on it.
- Rotating the view in the 3D mode is done by dragging with one finger.
- Long press is used to view the properties of components in the 3D mode.

- Rotating the device changes to the next page of the open document.

TapTapBlocks

- Simple building application.
- Tapping on an existing block either adds a new block or removes the existing one.
- Orbiting is done by dragging with one finger.
- Zooming is done with the pinching gesture.

BIMx

- 3D model viewer.
- Zooming is done with the pinching gesture.
- When close to the model, dragging with one finger rotates the camera.
- When far from the model, dragging with one finger orbits the center of interest.
- Panning is done by dragging with two fingers.
- Tapping the model once moves the camera to first person mode.
- When the camera is in first person mode, software joystick can be used to simulate walking in 3D model.
- In first person mode, movement can be done forwards or backwards and left or right with the joystick.

Star Legends

- 3rd person adventure game.
- Movement by tapping the ground once with one finger, which changes the center of interest to the tapping point.
- Movement in two dimensions is also possible with a software joystick.
- Orbiting is done by dragging with one finger.
- Zooming towards the center of interest is done with the pinching gesture.

Android phone

Tested applications were Minesweeper 3D, Tiki Cart 3D, Exorcist Fantasy 3D shooter, Rocka Bowling 3D, Caster Master 3D, SpaceCat, Pocket Legends, and Turbo Viewer. The first item in the bulleted list is a description of the application, and the rest of the bullet points introduce the application's control methods.

Minesweeper 3D

- Classic minesweeper game on a 3D cube.
- Opening windows is done by tapping the screen with one finger.
- Switching between modes is done by using software buttons.
- Rotating the cube is done by dragging with one finger.

Tiki Cart 3D

- Cart driving game.
- Accelerating the cart is done by pressing the screen anywhere.
- Turning the cart is done by rotating the device.
- Activating commands is done by pressing software buttons.

Exorcist Fantasy 3D shooter

- 3rd person shooting game.
- Software joystick is used for moving.
- Software buttons are used for activating commands.

Rocka Bowling 3D

- Bowling game.
- Throwing the ball is done with one finger swiping gestures.
- Commands are activated with software buttons.

Castle Master 3D

- 3rd person adventure game.
- Movement is done by tapping on the ground, which changes the center of interest of the camera.
- Orbiting is done by dragging with one finger. It can be only done horizontally.
- Zooming is done with the pinching gesture.

SpaceCat

- Arcade flying game.
- Accelerating the plane is done by pressing the screen.
- Turning the plane is done by rotating the device.
- The plane can be also controlled alternatively with software buttons.

Pocket legends

- 3rd person adventure game.
- Movement is done by tapping on the field, which changes the center of interest.
- Movement can be done also by using software joystick.
- Orbiting is done by dragging one finger anywhere else than on the joystick. It can be only done horizontally.
- Zooming is done with the pinching gesture.

TurboViewer

- Cad model viewer.
- Orbiting is done by dragging with one finger.

- Zooming is done with the pinching gesture.
- Panning is done by dragging with two fingers.
- Filling is done by double tapping the screen with one finger.

Conclusion: Controlling 3D applications with mobile touch screen devices

Generally each type of control was used only for single action. For instance, two-finger dragging was always applied to panning function in every mode of the application, and rotating the device was only used to turning of the car. Mixing the same control to perform different actions in different modes might be confusing.

The main difference between tablet and smart phone applications was that the applications in tablet have more functions and better graphics. 3D software on smart phones was mainly games. Both devices used the same type of controls. Basic controls were two dimensional gestures, software joystick, software buttons, device rotation, and built in digital compass.

Each of these controls work properly in certain applications. Basic two dimensional gestures are suitable for controlling the user view and activating specific commands, the software joystick is suitable for controlling movement, software buttons work for activating commands, the acceleration sensor works as an alternative method for controlling certain types of movement, and the digital compass works for applications where the direction that the user is facing matters. The most important thing in selecting the control type is that the use of the controls is natural. For example, the pinch gesture in zooming is natural, or turning the car by tilting the device is natural.

The worst use of controls was on applications where the controls worked in an unnatural way. If the use of a gesture requires a lot of memorizing and learning it is neither natural nor intuitive. For example, rotating the device in changing the page on document is unnatural. Mixing too many control types in the same application can make the use of the application difficult and confusing. For example, using software joystick and device rotation in the same application for different actions is confusing, since both are usually used for the same purpose, controlling the movement.

3.1.2 PACT analysis

PACT analysis is used to define the most likely use scenarios for this interface. As most of the technology selections were made before defining the scenarios, this analysis starts from defining rest of the technology selections. After that the most suitable user group is decided. The next step is to define the activities for the interface. In the last phase, the contexts as well as the complete use scenarios are determined.

3D world cannot currently be used in smart phones. Therefore the target platforms for this interface are the medium sized multi touch surfaces; Windows laptops with touch screen, desktop touch screens and Windows tablets. In MTS the software keyboard cannot be usually used at the same time with the touch mouse controls. Therefore

the keyboard modifiers such as ctrl or alt cannot be used with touch taps. Gesture-based control can be used as a replacement for these keyboard modifiers. Two dimensional gestures are the main usable control type for 3D world as they do not require any additional software buttons or software joystick to interface. Gestures are available on any kind of touch device, as for example the acceleration sensor cannot be used with touch screens on laptops.

Basically any 3D world user might use the touch screen interface, but the sales and marketing people could benefit the most from using it. Marketing people can have 3D world installed on a tablet. The tablet allows them to be more mobile and to be able to show easily the basics of 3D world in varying locations.

People in deciding positions usually use 3D world to run the simulations to give the base to their decision making process. MTS interface enables using the 3D world with tablets in remote locations, for example in trains or planes. But the additional benefits of the MTS interface for people in deciding positions are small, since usually the decision making process requires typing in some form. Typing is significantly slower with the MTS when compared to traditional keyboard.

The MTS surface does not bring any advantage to application engineers, since MTS surface is slower for basic mouse and keyboard tasks. Also, the accuracy of touching with finger is not high enough for the tasks that require high precision, such as off-line programming or building complex layouts.

As people and technologies have been selected, the next selection is the activities. The two dimensional gesture-based control cannot completely replace the functionality of mouse and keyboard, but it will bring advantages in certain environments performing certain tasks. Tests in the previous subchapter 3.1.1 revealed that the two dimensional gestures suit very well for controlling the user view. User view is traditionally controlled through mouse and keyboard modifiers, and as the keyboard modifiers are not available through touch screen, the user view controls are more than suitable for the first set of activities. Also, the marketing people use the software mainly for visualization, which makes these activities suitable for them to use. Second set of activities are the basic manipulation tasks that are performed in 3D world by dragging with mouse with the left button down. They are a part of basic functionality that marketing people will show to new customers.

People who use the activities define the context as well as the possible use scenarios. Physical context can be wherever the touch screen can be used; therefore it is not a limiting factor to usage. Social context can be wherever the 3D world itself is used which does not limit the use of the interface. Neither does the organizational context limit the use of this interface. The physical context for target users marketing and sales people could be any remote location where they need to show something for multiple people. For example the physical context can be at a fair, in a meeting, or in any big sales event.

An example of a use scenario for this interface is in small meeting at the customer premises where only some people are present. Marketing person brings his/her own

tablet to the meeting, builds small layout by dragging components on the screen, runs simulation there, and moves the camera around the layout with gestures. The customers can try the tablet and see and feel 3D world from the position where they are currently sitting.

Other possible use scenario for marketing people is at a fair. When 3D world is installed on tablet, marketing person can use it to catch the attention of people around the presentation booth by showing 3D world in it. MTS can be used also to show the basic functions privately to possible customers at the booth.

3.1.3 Development process

The development of this interface was done according to waterfall model. In this case the first set of tasks, the camera controls, was created and after evaluation it needed some basic actions to support fluent camera control. Basic supporting touch screen actions were implemented to the interface the next. In the last phase, the basic manipulation tasks were added to the interface in order to be able to use all of the main basic functions.

As the gesture-based control was selected as the main interaction method, the challenge was to find out how the gesture-based control should be implemented to the interface. Basically there were three main methods to utilize gestures in applications on .NET framework: to use only touch points and define own gestures, to use the raw input gestures or to use WPF manipulations. [81]

Defining own gestures would have been a good solution if there were some special gestures involved. For this interface, only the basic gestures were needed. Therefore defining them would have been replicating already made work. This option was abandoned due to aforementioned reason.

Raw input Windows messages are sent when the gesture is being performed on the touch screen. They can be used by handling the messages inside the application. All of the supported gestures in Windows 7 [18] can be received through the raw input. The problem with handling the raw input messages was that they are not supported in WPF. Other deficit of using raw input was that the raw input messages can only deliver information about one gesture that is performed currently. [81] For example, scaling and rotation can be performed with two fingers at once but only one can be detected at the time with the raw input messages.

WPF has built-in support for gestures in form of manipulations. Manipulations use events that report rotation, scaling and translation transformations simultaneously. That allows the developer to handle all of them in the same time. [81] WPF manipulations were chosen as the gesture implementation due to selection of WPF technology and limitations of raw input messages.

MVVM model was used in the touch interface development process. Gestures are performed on the invisible WPF canvas that is placed on the top of 3D world. View's code behind file handles the manipulation events manipulation starting and manipulation delta that occur when the gestures are being performed on touch screen. In event

handler, the COM interface of 3D world is accessed and respective actions are performed in 3D world. The manipulation delta event handling process is illustrated in figure in appendix 1. The class diagram of the touch screen interface is shown in figure in appendix 2.

Regular touch input is captured by the same WPF canvas. Touch events are used in capturing the tap and hold gestures. Touch events are handled in the view's code behind file. Handled events are touch down and touch up events. Actions are performed in event handlers similarly to manipulations. Actions triggered depend on the amount of taps, the number of fingers touching and the duration of the touches.

All of the implemented use cases on touch screen are illustrated in figure in appendix 3. Figure shows the actions that are available in different modes through the gestures. Modes can be changed by tapping the regular software buttons in 3D world.

3.1.4 Camera controls

The camera controls form the basic mode of the touch screen interface of 3D world.
The goal was to apply similar controls that were used in previously tested MTS applications — Turbo Viewer, BIMx, Structural View and 123D Sculpt. These applications were controlled through basic camera controls — pan, orbit, zoom, and fill. These functions were used with two dimensional gestures. In 3D world, fill can be tapped on the top panel; therefore it did not need to be applied to gesture-based interface.

Orbit

Orbit is mapped to one-finger drag when no component is selected. Orbiting is done in the same way that it is done with the mouse. When one finger is pushed to the screen and dragged on it, the camera starts orbiting around the center of interest horizontally in the same direction as the finger movement and vertically in the opposite direction to the finger movement. For example, when the user pushes the screen with one finger and drags it towards the top right corner, the camera orbits the center of interest right and down.

Pan

Pan is mapped to two finger drag and can be applied in any state of the application. Panning is done in the same way that it is done with the mouse. When two fingers are pushed to the screen and dragged on it, the camera starts panning in the opposite direction to the movement of fingers. For example, when the user pushes the screen with two fingers and drags the fingers towards bottom left corner, the camera pans towards the top and right. Pan and zoom can be performed simultaneously, since they both use two fingers, but different movement on the screen.

Zoom

Zoom is mapped to two finger pinch gesture and can be applied in any state of the application. The pinching gesture zooms towards the middle point of opening fingers. Respectively zooming out is done outwards from the middle point of pinching fingers. Zoom and pan can be performed simultaneously, since they both use two fingers but different movement on the screen.

3.1.5 Basic touch screen actions

Basic touch screen commands include actions that can be applied in any state of the application. These commands are the basic tapping and holding gestures.

- One-finger tap is used in selecting or unselecting components in 3D world. Selecting is done by tapping on components. Multiple components can be added to selection. Unselecting is done by tapping somewhere else in 3D world.
- One-finger long press is used as a right mouse click. It opens a context sensitive menu depending on the position of the finger.
- Double tap with one finger is used to change the center of interest in 3D world.
- Otherwise the interface uses the default touch input of Windows 7. [18]

3.1.6 Manipulating actions

When the component is selected the basic manipulating actions — translate, rotate, plug and play, and interact — can be applied to the component.

- Translate changes the position of the component in 3D world.
- Rotate rotates the component in 3D world.
- Plug and play translates the component in 3D world, and connects two components together if they have matching interfaces.
- Interact is used to interact with components that allow interaction.

These actions are applied to one-finger drag gesture. Dragging the finger does the same function as the mouse with left button clicked down. Manipulations are done in the direction of finger movement. When the component is selected, one-finger drag camera function orbit is disabled. Switching between the manipulating modes is done by tapping the top panel icons with one finger. Icons can be seen in figure 3.1.

Figure 3.1. *Icons of manipulating actions in 3D world.*

3.1.7 Future work

Generally gesture-based interaction with touch screen suits really well for interacting with 3D world. In the future, the touch screen activities should be integrated to the core of 3D world instead of using them through the COM interface. This can be done easily,

because currently the interface does not require any additional input buttons or other layout elements to it.

One improvement would be creating bigger handles on rotate action. Currently they are only one or two pixels wide and it is difficult to hit the correct handle with a finger.

Other possibility in the future is to create first person mode, similar to one that was used in tablet application BIMx. In BIMx the camera was positioned around the human eye level and the camera movement was controlled with the software joystick, imitating human movement.

3.2 Microsoft Kinect interface

Interaction through Kinect is available by using gestures or voice commands. The development focused on gesture-based control. However, the scale of tasks in 3D world was too wide for gesture-only interface. Therefore gesture-based could not replace the functionality of keyboard and mouse.

This section is about building interface to 3D world for Microsoft Kinect. The first subchapter is about selecting tasks and possible use scenario with the PACT analysis. In the next subchapter the development process and selections are discussed. Next three subchapters are about the implemented activities. In the last subchapter the future work is discussed.

3.2.1 PACT analysis

Technologies for this interface were pre-selected as Kinect was the input device and the output device was any screen that could provide video output. Three dimensional gestures were selected as the interaction method for this interface. The selection of other parts of PACT is discussed in this subchapter.

The people who use 3D world and the Kinect interface are defined by limitations of Kinect and gestural interface. Tasks that require a lot of typing with keyboard are not suitable for the gestural interface. Although the defining of gestures for the alphabet is possible [26], use of this kind interaction does not bring any advantage compared to keyboard as typing using gestures is much slower. Tasks that require high accuracy and precision are not suitable for Kinect's gesture-only interface [22] [26]. These kinds of tasks include offline-programming, controlling robot movement or building complex layout. The required accuracy is hard or impossible to achieve with gestural interface.

These limitations count out the application engineers and deciding people, who need keyboard and high accuracy daily in their work. The only remaining user category is the sales and marketing people. Sales and marketing people do not necessarily need either high accuracy or the keyboard and mouse in their presentations. They could use the Kinect interface in presenting the layouts. Kinect would provide a new and interesting way to communicate with the 3D world, which can create some extra depth to their presentation. The sales and marketing people are selected as the possible users.

The context is selected next. The most dominant of the context requirements, the physical context is defined by the limitations of Kinect. Kinect setup requires at least two meters of space in between the user and the camera to work properly [68], which limits the small spaces out, for example office rooms. Vision-based interaction devices are easily disturbed by any noise. That limits their use to spaces where the background is static and there are no obstacles or movement between the Kinect and the user. Kinect interface is not suitable for busy offices or for a fair, where plenty of people are moving in the background. An example of a physical context is a conference room with only the presenter standing in the front.

As the people, context, and technology are selected, the only remaining selection is the activities. The visualization is the main function that the marketing people use 3D world for. It is selected as the basis for the activities. The visualization is done through the user view. The user view in 3D world is called the camera. The first set of activities is the continuously manipulating camera controls pan, orbit and zoom. These activities were implemented to the interface.

The camera controls proved to be not suitable for the gesture-only controls. This experience was used in creating specially designed camera controls for Kinect. These controls form the second set of activities of the Kinect interface. They are implemented in the first person mode. Activating commands and modes is done through the menu, which forms the third set of activities of the interface.

In the defined use scenario a marketing person keeps a presentation in a conference room for a relatively large audience. He/she illustrates a big layout by using the first-person mode. In presentation he/she simulates walking through the facilities and starts and stops the simulation by using the menu in pre-defined places.

3.2.2 Development process

Development process of this interface was exploratory. Exploratory research is a methodological approach that is used when building a new theory or discovering something completely new. The research is exploratory, when no earlier model is used as a basis of the study. In exploratory research process the knowledge is being built-up while the research is done. The goal of exploratory research is to create theory from the data in a process of continuous discovery. [82] In this case, exploratory research process was used because there was no existing model or theory to be utilized as a basis for this kind of research. This development process aimed to building a usable interface to 3D world for Microsoft Kinect.

The research process started with browsing through the demos and documentation of Kinect for Windows SDK. Documentation and demos suggested the use of MVVM design pattern when building applications using Kinect. This is because the Kinect for Windows uses various multimedia channels and WPF presentation technology was developed to support different multimedia functions fluently under one platform and MVVM design pattern is designed specifically to be used with the WPF technology. WPF and MVVM were selected as base technologies for the functionalities. After the

basic technologies were selected, the next task was to apply functions on the interface. The first actions applied to the interface were the camera controls with right hand gestures.

In the first approach online gestures were used for manipulating the camera position. The position of the right hand was compared between two consecutive frames and the hand movement was used in performing the actions in 3D world. Using only the right hand gestures for manipulating the view created a new type of problem. How to switch between manipulation modes?

As the right hand was busy in manipulating the view, some other part of the body had to be used in switching between modes. The answer was to use offline left hand gestures. Offline left hand gestures were selected because the left hand can be idle or move randomly while performing manipulations on right hand. But when it performs a certain gesture the application activates other mode.

Open source project Kinect toolbox [83] provided the framework for offline gesture recognition. Kinect toolbox recognizes template gestures, swiping gestures and postures. The tracking of offline gestures happens as follows: Kinect sends events with every captured frame. The view's code behind file has event handlers for each type of Kinect events. View's code behind file sends information about user movements to model that is used for gesture recognition. The class structure that was used in gesture recognition and manipulating 3D world camera position is shown in class diagram in appendix 4.

Each type of gesture has a definition and certain rules that must be applied before completion. Gestures were recognized after a number of consecutive frames following the rules were achieved. The frame processing sequence is illustrated in figure in appendix 5. Figure shows an example of how the swipe gestures are recognized and used in activating modes. The same process was used in recognizing all of different types of gestures and postures.

The mapping of four different offline left hand gestures turned out to be a difficult task. The number of false interpretations increased quite a lot when there were only minor differences between the gestures. This problem was solved by creating a neutral mode called menu where nothing can be manipulated. The offline gestures were still used for activating manipulation modes from neutral mode and online gestures were used for controlling the camera in manipulation mode. The difference was that the cancel command, which activated the neutral mode, was the only available command while any manipulation mode was on. This selection allowed the whole body to be used for activating the different modes while the neutral mode was on. It decreased the number of false interpretations.

This setup was used in creating the camera controls. Different types of gestures were tried in activating camera manipulations. The next subchapter 3.2.3 provides more information of different approaches for camera controls. After trying two different setups the idea of combining different types of online gestures to the same manipulation mode came out. With this idea in mind first person mode was created, where all of the camera

manipulations are in the same mode. The menu is used in first person mode to activate commands while the camera manipulation is being done. The first person mode is introduced in subchapter 3.2.4 and development of menu is discussed in subchapter 3.2.5. All of the implemented use cases are illustrated in figure in appendix 6. Availability of different use cases in different modes can be seen from the figure.

Building a simple layout can be done by using the Kinect interface. As a test, a simple layout building solution was applied to interface. It was left out of the final version because of problems with the accuracy and difficulties in selecting the components.

3.2.3 Camera controls

Basic camera controls include actions pan, orbit, zoom and fill. These controls are introduced in chapter 2.4.1. Pan, orbit and zoom are continuously manipulating the camera position, while fill is applied only once when activated. This subchapter focuses on building the interface for continuously manipulating actions.

The COM interface of 3D world does not allow the direct use of the built in interactive camera controls; instead it allows manipulating camera position through camera's parameters. There was no example of using Kinect with this kind of camera controls in 3D applications. Therefore building of the interface for these controls was exploratory. It was done by trying out different approaches for the controls. The approaches that were tested out include hand gestures, hand postures and full body gestures. After discovering a solution, it was evaluated, and then the bad qualities were corrected as much as possible in the next approach.

Hand gestures

The basis for the first approach was to imitate the use of camera controls with a mouse. In this approach the right hand acts as a mouse with the left button clicked down. Application tracks the position of the right hand in Kinect video. This is illustrated in figure 3.2. Online gestures are recognized by comparing the position between two sequential frames. Because each of these tasks use right hand for gestures, they are implemented in their own modes.

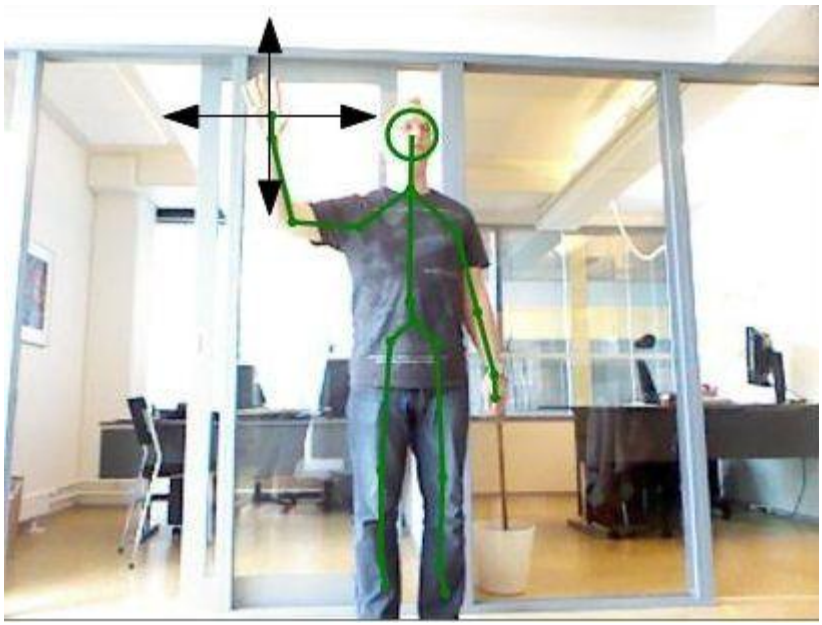


Figure 3.2. *Hand gestures: In pan and orbit the application tracks the right hand movement in height and width directions in Kinect video.*

Pan

- 3D window scrolls opposite direction to hand movement.
- For example, when user moves hand upwards and left the screen scrolls downwards and right.

Orbit

- Camera rotates its center of interest horizontally to the direction of hand movement.
- Camera rotates its center of interest vertically to the opposite direction of hand movement.
- For example, when user moves hand upwards and left, the camera orbits around the center of interest to down and left.

Zoom

- Camera zooms in when the user moves the right hand closer to Kinect.
- Camera zooms out when the user moves the right hand away from Kinect.

Evaluation of the first approach

Positive observations:

- Natural use of gestures, imitating the mouse and real-life applications.
- The gestures are not affected by user position.

Negative observations:

- Zooming in depth direction is hard to perform steadily.
- Temporal segmentation ambiguity: defining when the gestures start and stop. For instance, when the user tries to pan to the left and the hand cannot reach any further to the right, the user has to pull it back to the left, which makes the screen pan to the right.
- Changing of center is difficult, only available through pan mode.

Hand postures

In second approach the temporal segmentation ambiguity problem is solved by using hand postures. By moving hands to a certain direction from the neutral area the user makes a posture, which activates a command which continues as long as the posture is being made. Each of the tasks is implemented in their own modes.

Pan

Panning starts when the user raises his/her right hand above the waist line. Neutral area for panning is located at width and height of right shoulder in Kinect video. This is illustrated in figure 3.3. When the right hand is positioned inside the neutral area, the camera stands still. In pan mode, the application determines the direction of right hand from right shoulder by comparing their positions in width and height in Kinect video. Depending on the direction the 3D window starts scrolling.

- 3D window scrolls to the opposite direction of hand direction from shoulder.
- For instance, when the right hand is moved above and left of the shoulder level, the camera starts to pan down and right until the hand is lowered again to the shoulder height or the action is canceled.

Orbit

Orbiting starts when the user raises his/her right hand above the waist line. Neutral area for orbiting is located at width and height of the right shoulder in Kinect video. When right hand is positioned inside the neutral area, the camera stands still. In orbit mode, the application determines the direction of right hand from right shoulder by comparing their positions in width and height in Kinect video. Depending on the direction, the camera starts rotating around its center of interest.

- Camera rotates the center of interest horizontally to the direction of the right hand from the right shoulder.
- Camera rotates the center of interest vertically to the opposite direction of the right hand from the right shoulder.
- For example, when the user moves the right hand to the right and above the right shoulder level, the camera starts rotating to right and downwards around the cen-

ter of interest until the hand is moved back to shoulder level or the action is canceled. This is illustrated in figure 3.3.

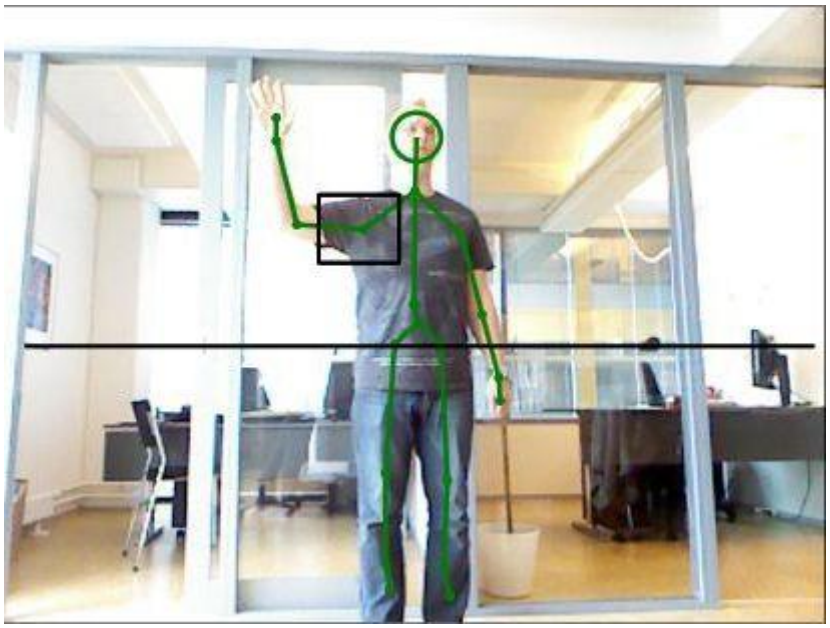


Figure 3.3. *Hand postures: Neutral area for panning and orbit is at the shoulder level. When the user raises right hand above the waist line to the shoulder level, the action starts. The direction of action is determined by the right hand direction from the right shoulder. In the figure, hand is above and right from the shoulder level. When orbiting, the camera constantly rotates right and below the center of interest. When panning, the camera scrolls towards down and right.*

Zoom

Zooming is done with two hands, and it is based on the pinching gesture performed in touch screen. Neutral zone for zooming is located around width of both shoulders in Kinect video. When the user's both hands are located at the shoulder level, the camera stands still. The application determines the direction of hands by comparing position of the left hand to the left shoulder and position of the right hand to the right shoulder in width and height in Kinect video. Depending on the position of the hands the camera starts zooming. Zooming with two hands is illustrated in figure 3.4.

- Camera zooms out when the user's both hands are inside of the shoulder level.
- Camera zooms in when the user's both hands are outside of the shoulder level.
- Zooming is done as long as both hands are located on the same side of shoulder level or until the action is canceled.

Figure 3.4. *Hand postures: Zooming out with both hands inside the shoulder level.*

Evaluation of the second approach

Positive observations:

- Zooming function is easier to perform with this version than the first version.
- Continuous functions work more fluently compared to ones based on comparison of two frames.

Negative observations:

- Decreased naturalness, when compared to the first approach due to the use of neutral zones.
- When the hand is moved in front of the body, Kinect tracking gets sometimes disturbed. This is because when one of the body joints gets behind some other joint, the Kinect skeleton tracking system cannot make decision where the joint is exactly located. This is illustrated in figure 3.5.
- Center of interest can still be changed only through pan function.

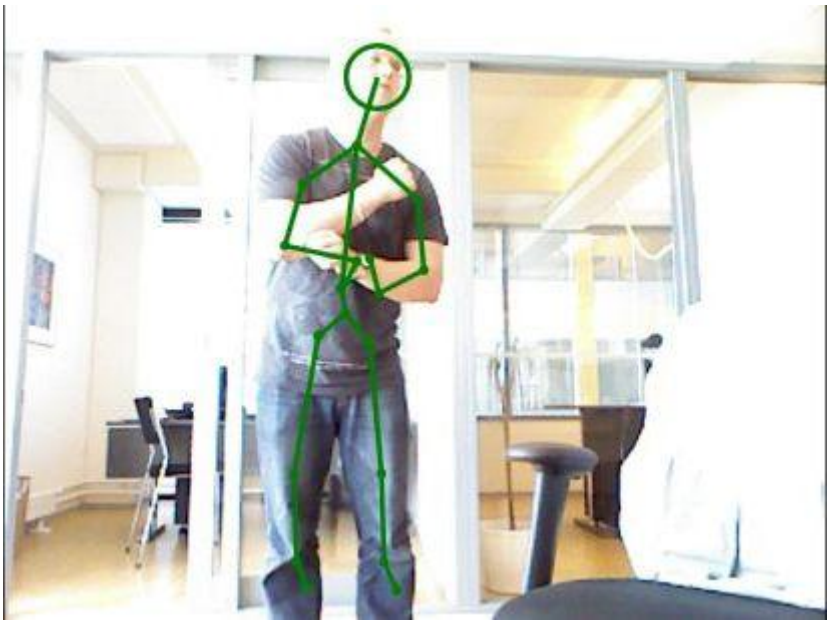


Figure 3.5. *Kinect skeleton tracking system sometimes gets disturbed when joints go on top of each other. Notice the position of skeleton hands compared to actual hands on the image. This happens often with arms, but also with the other joints in the body.*

Full body gestures

In this approach, using only hand gestures is abandoned due to the Kinect tracking problems, when joints go on top of each other. Instead, the full body gestures are used to decide the direction of where the camera should be going. The continuous functions are used in this approach. Significant difference to previous approaches is that the zoom and orbit modes are combined in the same mode in this version. Pan is still done with the hand postures similarly to the second approach.

Orbit

Horizontal orbiting is done with the shoulders. Positions of both shoulders are tracked in depth direction. Their positions are being compared and used in determining the direction of orbiting. Vertical orbiting is done similarly to previous approach with the hand posture.

- Camera rotates horizontally to the direction that the user turns.
- For instance, when the user turns 30 degrees to left, the camera starts rotating its center of interest towards left. This is illustrated in figure 3.6.
- Camera rotates vertically to the opposite direction, that the right hand is located to the right shoulder.
- For example, when the user's right hand is above the shoulder level, the camera rotates its center of interest to down.

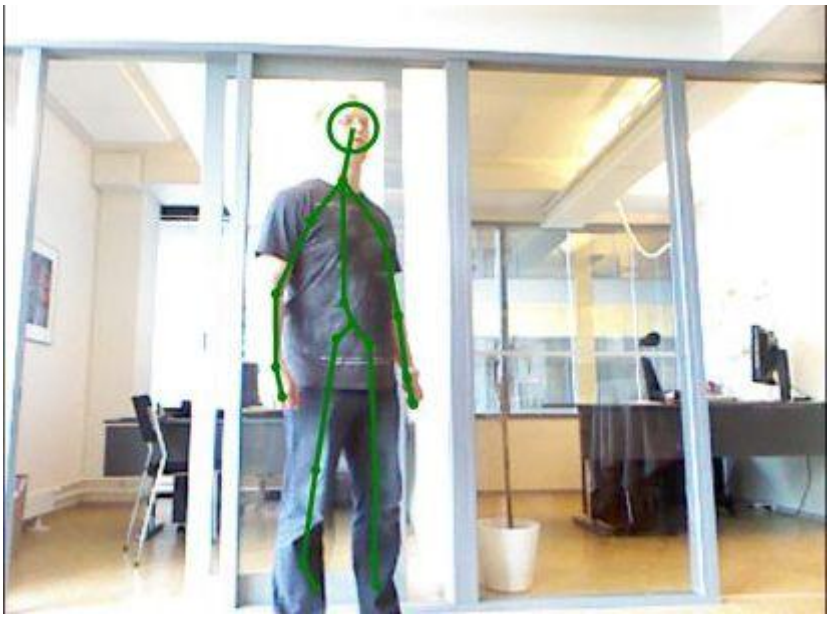


Figure 3.6. *Body gestures: The right shoulder is moved closer to the camera than the left shoulder. The camera rotates its center of interest towards right.*

Zoom

For zooming, the distance of the user's head from the camera is being tracked. Neutral area for zooming is between 2.0 m and 2.5 m from Kinect in depth direction. When the user stands in the neutral area the camera does not zoom. Zooming areas are illustrated in figure 3.7.

- Camera zooms in when the user moves closer to the camera than 2.0 m.
- Camera zooms out when the user moves further from the camera than 2.5m.

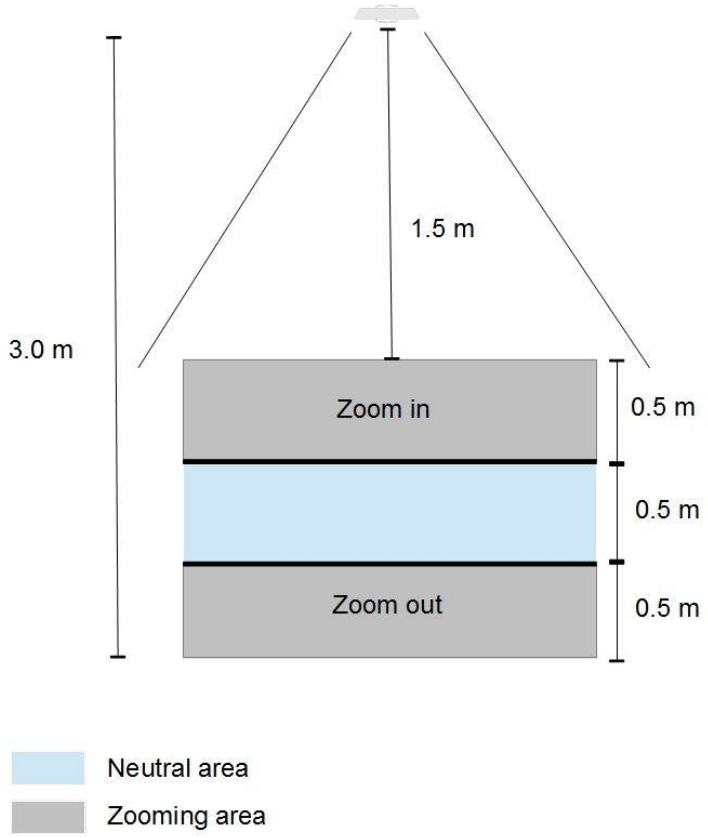


Figure 3.7. *Zooming areas: When the user steps from the neutral are to the moving area, and stands 1.8 m from Kinect, the camera zooms in.*

Evaluation of the third approach

Positive observations:

- Full body gestures enable combining zoom and orbit in the same mode, when the different body parts are being used for different actions.
- Requires less activating of different modes.
- Increased naturalness compared to both previous approaches.

Negative observations:

- Pan function unchanged.
- User is limited to gesture area.
- Center of interest can still be changed only through pan function.

3.2.4 First person mode

The lack of ability to change the center of interest fluently through traditional camera controls created a demand for a new type of camera control, more suitable for Kinect interface. In the new approach the traditional controls are abandoned. Instead, the camera is directly controlled through its parameters. A requirement for this approach was to combine all the available continuous actions under the same mode. Another requirement

was to increase the naturalness of the interface. From these premises the first person mode was created.

In first person mode, the camera is being manipulated so that the view looks like a person is walking in 3D world. Camera manipulation is done by setting the camera to 1.80 m height from 3D world's xy-plane, and moving the view so that the camera stays always 1.80 m above the ground. This is illustrated in figure 3.8.

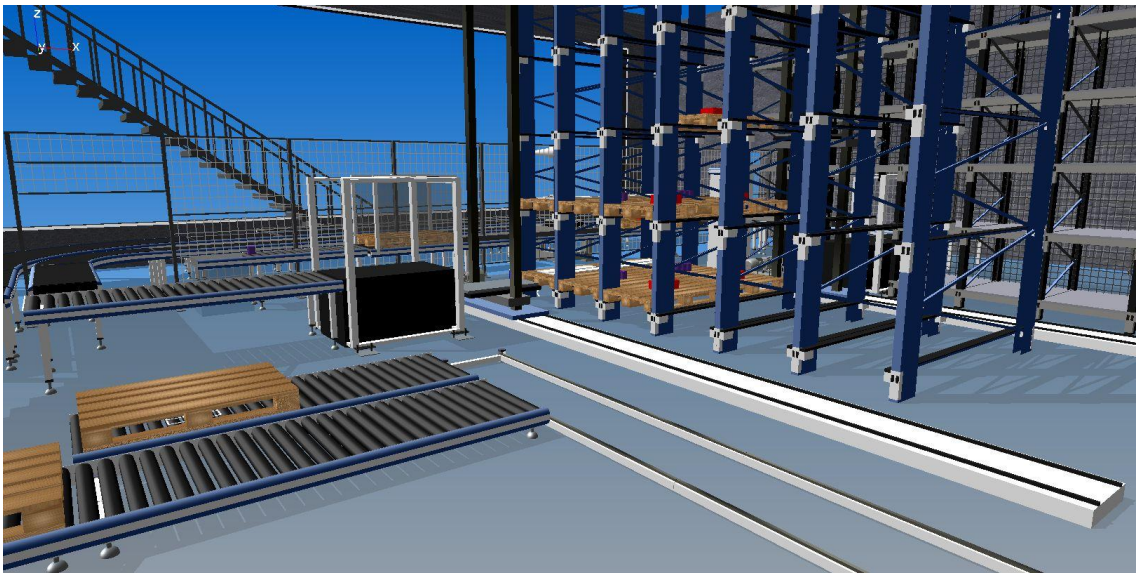


Figure 3.8. *First person mode: The camera is set to look 3D world from 1.80 m height from the floor.*

The first person mode can be divided in three main tasks that can be performed simultaneously. These tasks are: moving, turning and tilting. This subchapter explains these functions in more detail and tells about implementing them to the interface. Also, the most important option of first person mode, limiting the movement is explained in this subchapter.

Moving

Moving the camera is done in two dimensions. The user can move the camera forwards or backwards and to left or right. Moving can be done towards either single dimension or combine elements from two dimensions and simultaneously move towards both. The elements must be from different dimensions. This means that the user cannot simultaneously go forwards and backwards or left and right. When moving, the position of the camera changes, but the rotation of the camera stays still.

Head tracking is used for moving the camera. Camera moves continuously with the speed of 0.75 m/s, independent of direction. Neutral area for movement is located between 2.0 m and 2.5 m from the camera and it is 1.0 m wide. When the user is standing in this area, the camera does not change its place.

If the user wants to move the camera forwards, he/she has to move closer than 2.0 m from the camera, but because of Kinect field of view, not closer than 1.5 m from the camera. [68] Respectively, when moving backwards, the user has to move further than

2.5 m from camera. In order to move the camera left, the user has to move to the left of the camera so that the head is 0.5 m left from the camera center line, but not further than 1 m away from the center line. [68] The same rules apply for moving the camera to right; user's head must be 0.5 m to 1.0 m right from the center line. For instance, when the user's head is 1.8 meters away from the camera and 0.8 to left from the camera center, the camera moves simultaneously forwards and left. Map of camera movement directions is illustrated in figure 3.9.

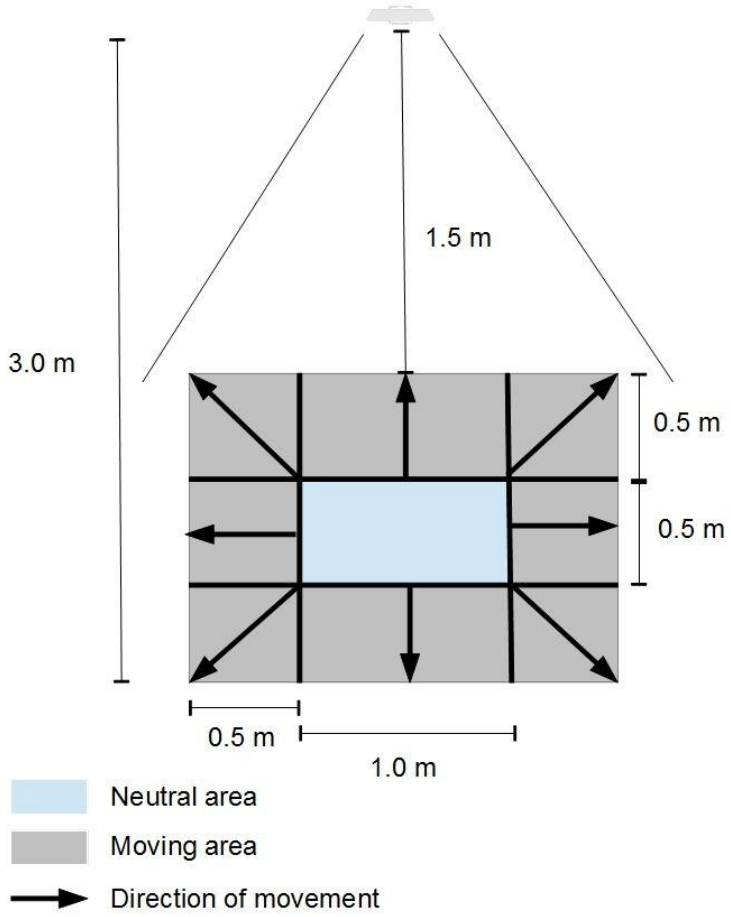


Figure 3.9. *Kinect camera movement map: User steps from the neutral area to moving area and camera starts moving at the speed of 0.75m/s towards the direction that the arrow indicates.*

Turning

Changing the direction of the camera is done by manipulating the rotation parameters of the camera. Camera can be turned towards left or right by manipulating the camera's yaw rotation parameter. The horizontal rotation of the camera affects the direction of movement.

Turning the camera left or right is done by rotating the shoulders. The camera is turned in the way that the user is turning. The application uses the distance of the shoulders from the camera to determine the direction of rotation. When the difference of

shoulder positions is 0.1 m – 0.4 m the camera starts turning with the speed of 15 degrees in a second. When the difference of shoulder positions is greater than 0.4 m, the camera starts turning faster, with the speed of 75 degrees in a second. For example, when the user turns right, by moving his/her right shoulder 0.2 m closer to camera than left shoulder, the camera starts turning to right with the speed of 15 degrees in a second. This is illustrated in figure 3.10. Respectively, when the user turns left, by moving his/her right left shoulder forwards 0.4 m more than the right shoulder, the camera starts turning left with the speed of 75 degrees in a second. The camera can be turned when standing in any position of Kinect movement area, which is illustrated in figure 3.9.

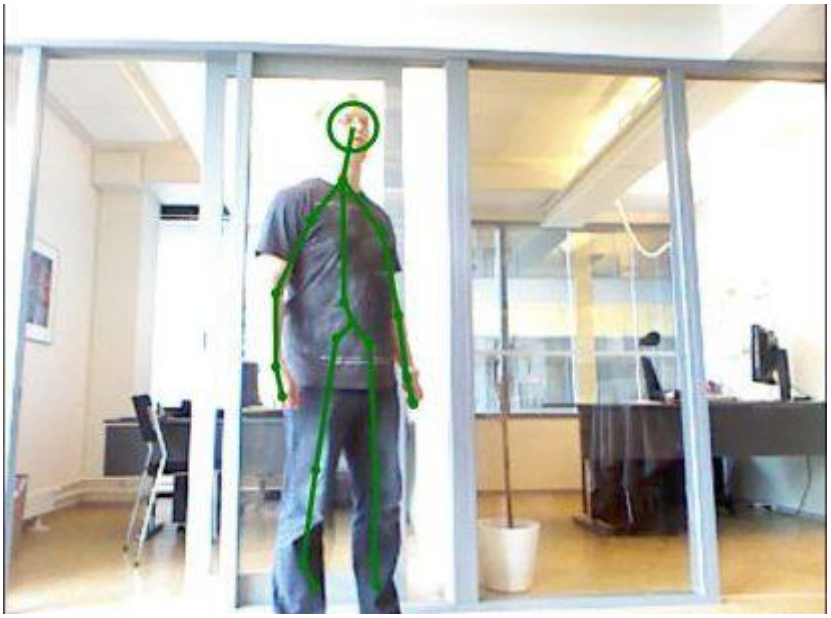


Figure 3.10. *Turning the camera left, the right shoulder is 0.2 m closer to the camera than the left shoulder. Camera starts turning left with the speed of 15 degrees in a second.*

Tilting

Tilting the camera upwards or downwards is done by manipulating the camera's pitch parameter. Tilting the camera is used only for looking, therefore tilting the camera does not affect the direction of movement.

Tilting camera up and down is done with the right hand. There are two neutral zones for tilting the camera. First zone is below the waistline, and second zone is located at the shoulder height. The tilting mode is activated when the user raises his/her right hand above the waistline to shoulder height. To tilt the camera upwards, the user raises his/her right hand above the shoulder line. Camera rotates upwards with the speed of 15 degrees in a second until the hand is lowered to shoulder level. This is illustrated in figure 3.11. To tilt the camera downwards, the user has to lower his/her right hand below the shoulder line, but above the waistline. Camera rotates downwards with the speed of 15 degrees in a second, until the hand is raised to shoulder height. When the user lowers

his/her right hand below the waist line, the tilt angle is reset to the value that it was before tilting.

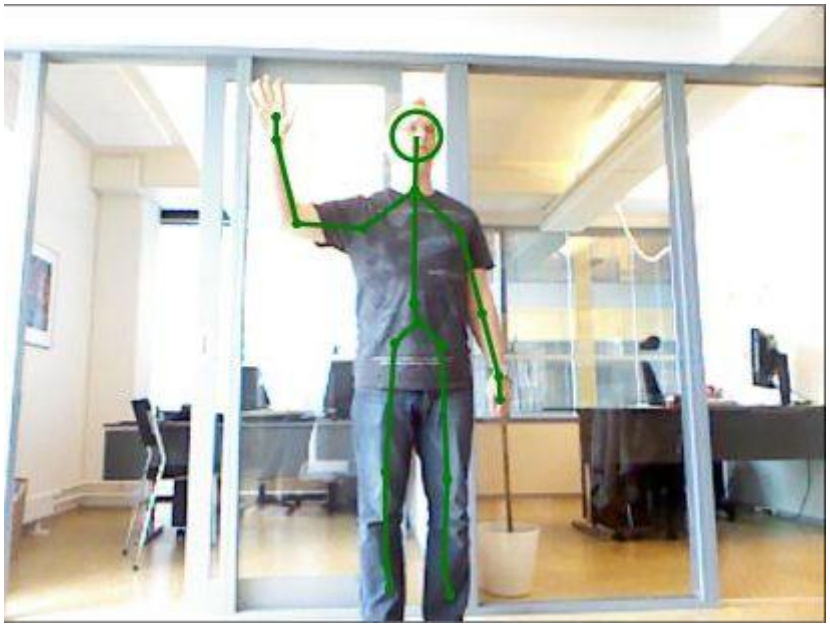


Figure 3.11. *Tilting the camera: Right hand is raised above the shoulder level. The Camera is tilting upwards with the speed of 15 degrees in a second.*

For tilting the camera the head tracking was considered first to be more natural, but the idea was abandoned as the user also has to see the screen while he/she is performing the gesture. For instance, if user looks down in order to tilt the camera downwards, it becomes really hard to see the screen at the same time.

Limiting the movement

The movement can be limited to a certain floor type by setting a parameter from the application and covering the area with this floor type where the movement is wanted. By restricting the movement, the walk in 3D world can be made more realistic. For example, the camera can be restricted to areas where people walk normally. The floor can be tilted sideways, for example to follow stairs. When the camera arrives to tilted floor it follows its plane 1.80 m above it. By not restricting the movement to a certain floor type, the application allows free movement. With this option the camera goes through all the layout elements of 3D world.

Evaluation of first person mode:

Positive observations:

- All of the required functions are under a same mode.
- Moving and turning the camera is easy and natural.
- Previous problems are fixed.

Negative observations:

- Tilting the camera is not natural and requires more learning than other functions.

3.2.5 Menu

Menu contains a set of activated functions available in different modes through offline gestures. The activated commands depend on the mode in which the application currently is. To test what would be the best practice for this kind of system different approaches for menu were tried. There were four different types of offline gestures available: swipe gestures, template gestures, postures and pointing gestures. This subchapter focuses on trying different approaches for menu functions.

Activating commands is typically done in Xbox 360 games by pointing gestures or by combined pointing and swipe gestures. For example, commands in games are activated by moving cursor on certain spot in the screen using the right hand. Another example for menu is swiping the right hand towards left at a certain height.

The screenshots of this section use the Kinect video output which mirrors the image of the actual process. Therefore the screenshots look like the left hand is being used when talking about the right hand. The captions describe the actual actions happening in the image.

Swiping gestures

For the first approach only swiping gestures were used. Swiping hand gestures are done by rapidly moving a hand from a point to a certain direction. In this approach the commands were mapped to swipes to different directions. For example, canceling current action and returning to idle state was mapped to swipe to down with right hand. This is illustrated in figure 3.12.

Figure 3.12. *Swiping gesture: Canceling the current action and returning to idle state by swiping down with the right hand.*

This approach was problematic because the number of accidental swipes increased as the amount of different swiping directions increased. For instance, the user swipes with the left hand to left correctly, activating a command. After a successful swipe the user normally lowers the hand to the leg, accidentally activating another command that was mapped to swipe down with the left hand, which negates the effect of the first swipe gesture. Stricter rules for swipes would have solved this problem, but at the cost of usability, as the swipes would have been more difficult to perform correctly.

The use of swiping gestures also requires the information of what is behind each command because the swipes are not self-describable. This approach was abandoned due to abovementioned problems.

The second option for swiping gestures is that the user swipes to a certain direction in different heights to activate commands. This approach is used in Harmonix's Dance Central game for Xbox 360. The menu of the game is illustrated in figure 3.13. The game was tested and in the game the menu was hard to use especially when the number of options in the menu increased. It was difficult to swipe exactly from a certain height. This kind of menu resulted in plenty of false selections. Consequently, this option was never applied for this interface.

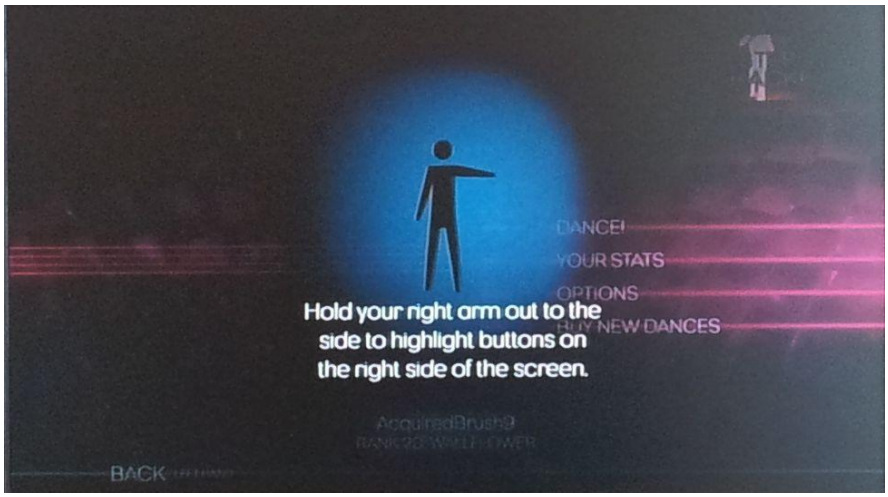


Figure 3.13. *Dance central menu: Selections are activated by swiping from the correct height.*

Template gestures

In the second approach template gestures were used. When using template gestures, path of hand in gesture is saved in a template. The application tracks user movement and compares it to the templates that it has. When the movement is similar to the template, application triggers an action. In this case, templates were used for letters. Commands were mapped to match their initial letter. For example O was used for activating the orbit mode and Z was used for activating the zoom mode. This is illustrated in figure 3.14.

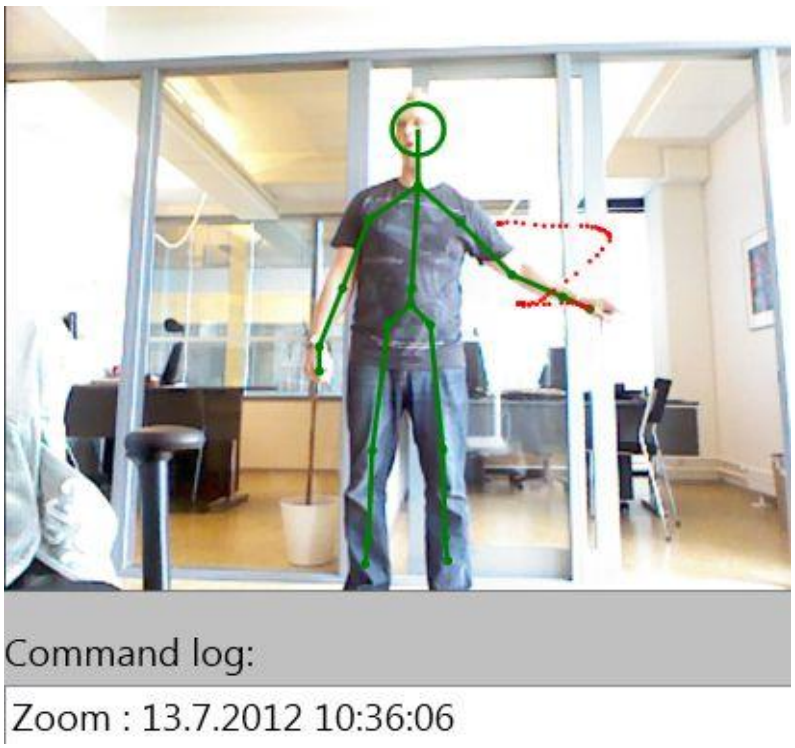


Figure 3.14. *Template gesture: Activating zoom mode by drawing letter Z in the air.*

The problem with this approach was that these kinds of gestures are unique for each user. Each user draws letters in different shapes and sizes. This could be solved by adding more templates to every letter used. But as the amount of templates raised so did the number of misrecognized gestures and still the near 100% gesture recognition was not achieved.

Another problem that came up was the definition of starting as well as the ending point of template gesture. It was hard to know exactly when the user starts to perform a gesture and when it is complete. This too varies a lot between individuals. The amount of false recognitions was too high, which caused the abandoning of this approach.

Postures

In the third approach, postures were used. Postures are a form of static gesture. In posture recognition user makes a pose, and the application tries to determine whether the joints are in correct positions for a certain posture. An example of a posture would be raising the left arm to a 90 degree angle from the body and keeping it there for two seconds. This is illustrated in figure 3.15.

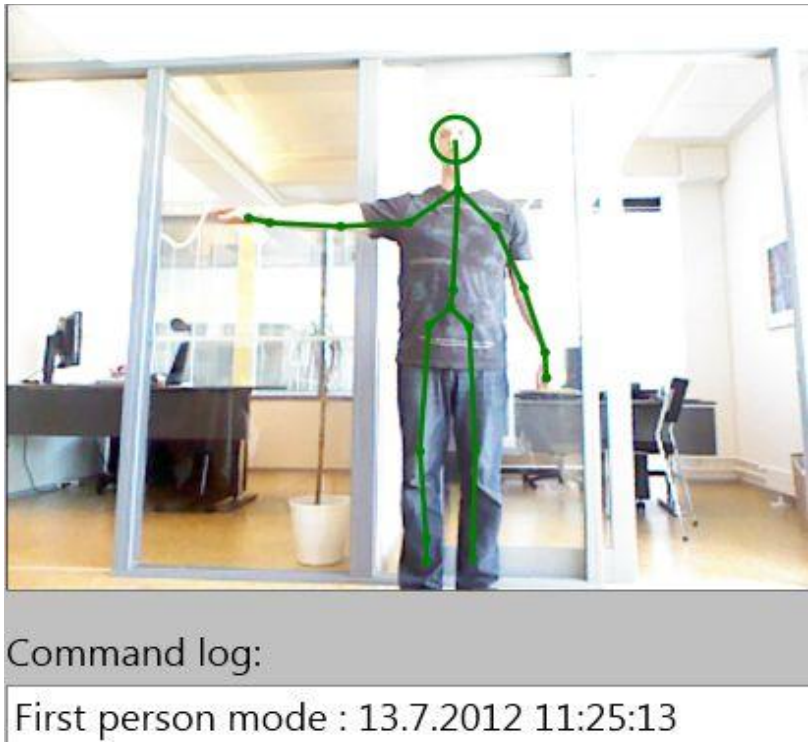


Figure 3.15. *Posture: Activating first person mode by raising the left arm to a 90 degree angle from the body and keeping it there for two seconds.*

The problem with posture recognition when using Kinect is that if the joints go behind each other the tracking might get disturbed. This is a serious case especially when the movement stops. Its build-in tracking software cannot find the lost joints properly from the relatively static image. It requires movement for finding out the connections between joints. Basically this limits the posture use to ones that are performed outside

the image of the body in Kinect video. And because the right hand is used in other gestures, only the left hand is available to posture recognition.

Despite the difficulties, the posture recognition works outside the body frame more reliably than the first two approaches. Activating different commands worked with almost 100% accuracy when there were four different left hand postures available at the time. If more commands were available, the accuracy got lower due to similarity between some postures.

The main problem with the postures was that they are not usually self-describable and require knowledge about commands mapped to them. The posture-only approach was abandoned because of it.

Swipes, templates and postures

The fourth approach was a mixture of the three previous approaches. In this approach, one command was mapped to one type of gesture at the time, limiting the amount of currently available gestures to three: One swipe gesture, one template gesture, and one posture. This lowered the number of gesture misinterpretations compared to both templated gestures only and swipe gestures only approaches. Nevertheless, the number of misinterpretations was higher than with the posture-only approach.

Problem with this approach is that it requires a lot of learning and memorizing of how to perform a gesture and about actions that lie behind each gesture. Abovementioned problems caused the abandonment of this approach.

Pointing gestures

The fifth and final approach was to use pointing gestures. Pointing gestures are postures that are used to point at something. In this approach the user uses his/her left arm as a pointer. The available selections are added to the maximum of three boxes in the left side of the screen. Every box has a description of an activity it triggers. The user selects the action by making a posture in which the left arm, from elbow to hand, is pointing to the wanted activity for two seconds. When an action gets triggered, the selections in the boxes change to respond the current state of the application. Pointing gesture is illustrated in figure 3.16.

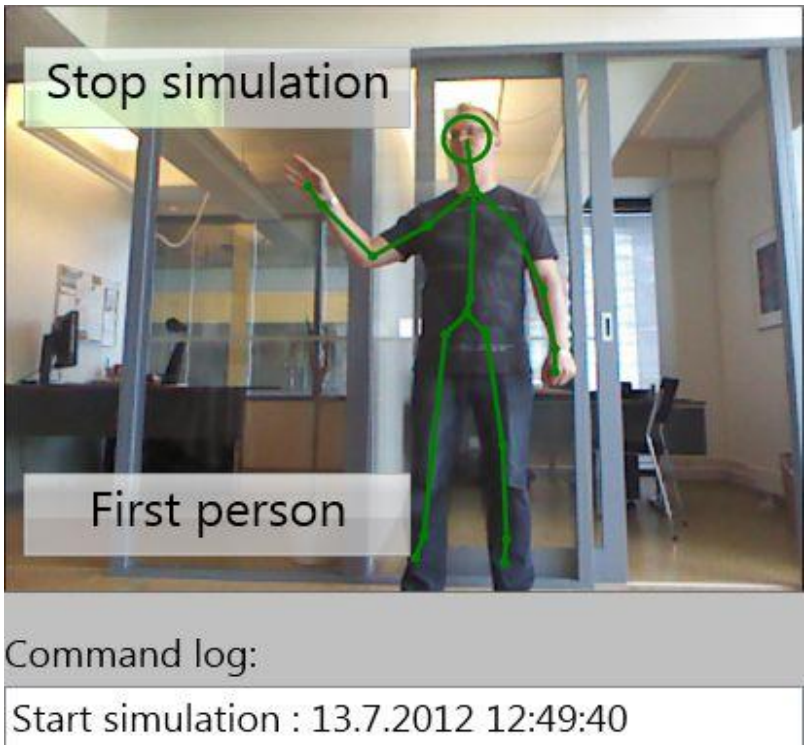


Figure 3.16. *Pointing gesture: In the middle of activating the stop simulation command by pointing to it.*

To help the user to know what activity he/she is currently pointing, a progress bar is added to every box to visualize the progress of the pointing gesture. The progress bar lowers the amount of misinterpretations to almost zero, as the user gets constant visual feedback about the selection that he/she is currently making. Pointing gestures are a very natural way of selecting objects and if the selections are descriptive enough, the user is able to use the application correctly on the first time he/she is using the application.

An example of menu usage with pointing gestures: The application is in first person mode. There are two options available: start the simulation in the top left box and go back to idle mode in bottom left box. The user wants to start the simulation. The user points to the top left corner with the left hand. After pointing there for two seconds the simulation starts. The top left selection changes to stop the simulation, while the bottom left selection stays the same. The user has started the simulation.

3.2.6 Future work

In future the Kinect interface should be integrated more tightly to the core of 3D world. Using the COM interface in interacting adds an extra step in the communication, which slows the usage a bit. Interface suffered from performance issues when the simulation was on. The deeper integration with better prioritizing of Kinect input might help with this issue.

More fluent use of camera controls requires more work in the future. The work should focus on better utilization of full body gestures. How to change the center of

interest flawlessly with full body gestures and how to pan without using the hand gestures are some of the most important issues.

In the first person mode more interactions with the environment could be added. For instance opening doors and handling switches could be performed intuitively with gesture-based control. The general intuitiveness and naturalness of first person mode should be considered in the future. How to tilt the camera without using hand gestures could be the first target of improvement.

In future more functions could be added to this interface. For example mouse cursor and clicking with the mouse could be controlled through this interface. This would allow more interesting functions to be available. For example building complete layouts could be done when the mouse is controlled with gestures.

3.3 3DConnexion SpacePilot PRO interface

This subchapter is about building interface for 3DConnexion SpacePilot PRO. In the first subchapter the most likely use scenario for this interface is defined using the PACT analysis. After that the development and decision making process that was used in building this interface is discussed. The following three subchapters consist of introducing the implemented functions. The final subchapter is about the future work.

3D mice are commonly used in various CAD environments. 3D world is similar to CAD environment because in both environments the camera as well as objects can be moved freely in the world. In CAD environments the controller cap is used to control the camera movement and to manipulate the position and the rotation of objects. For this interface the goal was to apply similar functionality to 3D world.

The main requirement for this interface was that it could be used also with other types of 3D mice than SpacePilot PRO. The key feature of every 3D mouse is the 6DoF sensor. Due to this the development process focused on the functions that could be performed with the controller cap.

3.3.1 PACT analysis

PACT analysis was used for this interface to define the most suitable use scenarios. Technologies were pre-selected as the SpacePilot PRO was the input method and the any screen that is capable of showing 3D world was an output method. The rest of the PACT elements had to be chosen.

3D mice provide an additional input method to be used aside keyboard and mouse. They do not try to replace the functionality of keyboard and mouse. Therefore any user of 3D world can use them without losses in usability. Traditionally application engineers have used 3D mice for controlling the user view as well as translating and rotating the objects during the designing process. The application engineers are selected as the most potential users, as they usually use this type of interaction and might already own the equipment. Nevertheless 3D mouse can bring benefits to any user that feels comfortable using it.

The context and the activities are selected to support the most typical use of 3D mice by application engineers. As said in the previous paragraph, 3D mice are typically used in controlling the user view, which means translating and rotating the camera. The camera (user view) mode is selected as a first set of activities. The Second main functionality of 3D mice is rotating and translating objects in three dimensional space. The object mode is selected as a second set of activities. Third set of activities is the supporting functions to the two main functions. This set includes commands reset and fill. The context is any environment where the designing process is typically done. A typical physical context is an office room.

In the defined use scenario, an application engineer uses a 3D mouse in an office while building a layout. He uses 3D mouse to control the camera in order to see the layout better from a different angle. Then he performs manipulating actions with keyboard and mouse. After performing an action he changes the view point again with the 3D mouse in order to obtain a better look of what he just did.

3.3.2 Development process

The development of this interface started with browsing through the demos and documentation of SDK for a 3D mouse [80] and looking through the material on 3DConnexion web page [77]. In SDK demos the 3D mouse is used either to control the camera or to control the object translations and rotations. For camera control the controller cap is used to move the camera in the direction of the cap movement. Camera control does not require any other input device than the 3D mouse. In object control, same translations and rotations that are physically applied to the cap are applied to selected object(s). For object control the keyboard and mouse are used in selecting the objects that are being manipulated at the time. The implemented use cases for this interface can be seen from the figure in appendix 7.

The next target was to find out how the 3D mouse input can be used. The web page documentation revealed that there were two main options to utilize 3D mouse input: either by using SDK or by using the raw input of 3D mouse. SDK offers API for C++ while raw input provides direct access to 3D mouse input in C# for example. Raw input was chosen as the access method, because the COM interface of 3D world is accessed more easily with C# than C++.

3DConnexion provides some sample code which uses the raw input to access the 3D mouse. The sample consists of a set of classes which provide an easy access to the device input data, most importantly to the motion events and the button events of 3D mouse. Motion events from the 3D mouse carry the input data of the controller cap in two vectors: Translation vector and rotation vector. Translation as well as rotation vector consist of three integer numbers that deliver the information about the amount and direction of the translation and rotation. The numbers are scaled from -350 to 350, which is the usable range of the sensor. [80] The sign determines the direction and the value is determined by the pressure that the user gives corresponding direction. Button events carry the button mask information that can be used to identify the pressed button.

These inputs can be then utilized however the user wants to. The class diagram of the SpacePilot PRO interface is illustrated in the figure in appendix 8.

The following example of a camera translation demonstrates how the 3D mouse input is processed: The controller cap is translated to left. The raw input is processed in a responding class and a motion event is generated from that input. Event handler captures the event, checks what mode is currently on and lets the responding class create a new translation vector to 3D world. The translation vector is then returned to the event handler and used to translate the camera to left through the COM interface of 3D world. This process sequence is illustrated in the figure in appendix 9.

3.3.3 Camera mode

In the camera mode the motion events of the raw input are used to control the camera. Camera mode is activated when there are no components selected in 3D world. Camera is controlled by manipulating the parameters of camera through 3D world's COM interface. Camera can be translated in three dimensions and rotated in two dimensions.

The motion event's translation vector is used to translate the camera. When the sensor is pushed forwards, the camera moves closer to its center of interest, as in zooming in. The camera moves to the direction of the screen coordinate system's depth z-axis. The difference between world coordinate system and screen coordinate system is illustrated in figure 3.17. When the sensor is pulled backwards, the camera moves away from its center of interest. The center of interest is moved the same amount to the same direction as the camera. By doing this, the camera does not ever reach its center of interest. The speed of camera movement is determined by the value of vector's z-translation. For instance, when the cap is pushed forwards to the cap limit, the camera moves forwards with full speed. Translation vector's z-translation is 350, while both x and y-translations are 0.

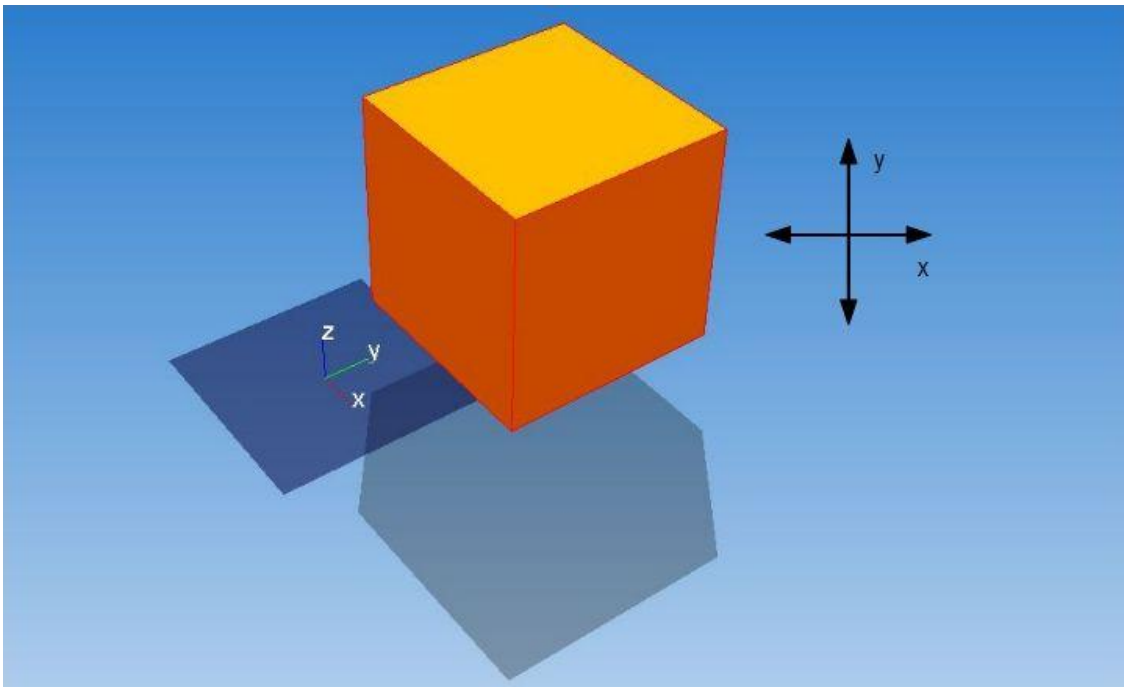


Figure 3.17. *Difference of two coordinate systems: On the left, the world coordinate system. On the right, the screen coordinate system. Z axis in screen coordinate system is perpendicular to both x and y-axis and is going towards the screen. Camera mode uses both systems in different functions.*

When the sensor is translated in horizontal direction to left or right, the camera is translated to respective direction along the screen coordinate system's horizontal x-axis. The speed is determined by the value of the translation vector's x-translation. When the cap is translated vertically, the camera translates to according direction along the world coordinate system's vertical y-axis. Cap translations are illustrated in figure 3.18. The speed is determined by the value of translation vector's y-translation. For example, when the sensor is pushed down to the cap limit and to right to the middle of cap's range, the camera is moving in screen coordinates downwards with a fast speed and to right with a moderate speed. Translation vector's x-translation is somewhere around 175, y-translation is -350 and z-translation is 0.

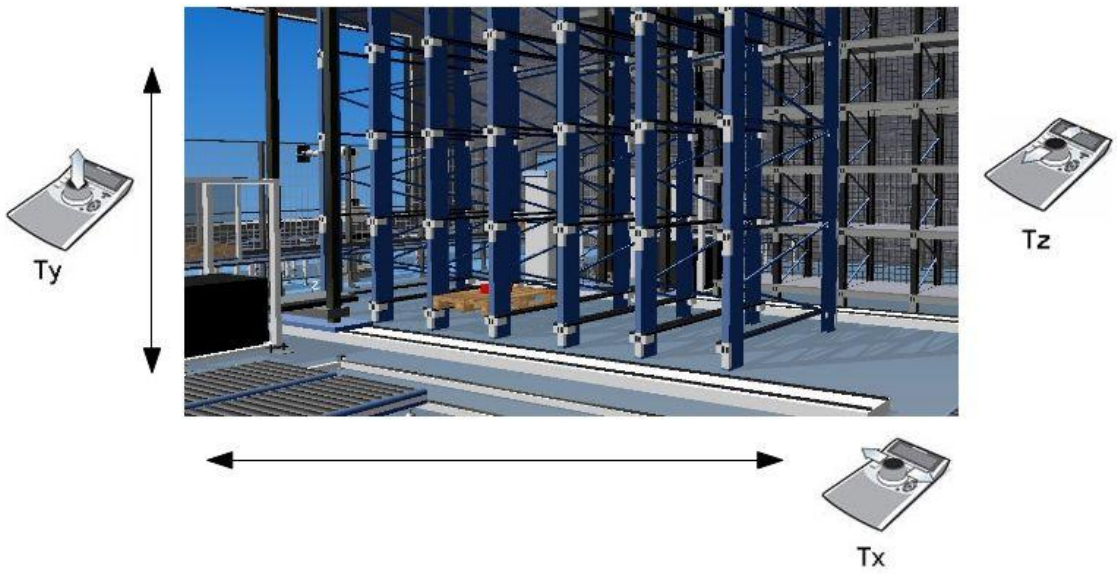


Figure 3.18. *Camera mode: Translating the cap in x direction moves the camera in the direction of horizontal axis in the screen coordinate system. Translating the cap in y direction moves the camera in the direction of vertical axis in the world coordinate system. Translating the cap in z direction moves the camera in the direction of depth axis in the screen coordinate system.*

The motion event's rotation vector is used to rotate the camera. When the cap is rotated around its vertical y-axis the similar rotation is applied to the camera. In 3D world the camera is rotated around the world coordinate system's vertical z-axis. The speed of rotation is determined by the torque that the user uses to rotate the cap. For example, when the sensor is rotated around its vertical axis to right with the minimum recognized torque, the camera starts turning slowly to right. Rotation vector's y-rotation is 1and both x and z rotations are 0.

When the cap is tilted around its horizontal x-axis, the camera rotates accordingly around the screen coordinate system's horizontal x-axis. The speed of rotation is determined by the torque that the user uses to tilt the cap. For instance, when the user tilts the cap backwards, with the maximum torque of the cap, the camera starts rotating upwards. The rotation vector's x-rotation is -350 and both y and z-rotations are 0. Vertical rotations are limited so that the user can only turn the camera 90 degrees upwards and 90 degrees downwards.

The cap's z-rotation is not used in the camera mode. Because of this decision the camera never moves to the position where the view is tilted sideways. Basically this means that by tilting the cap forwards the camera always rotates downwards and the user eventually sees the floor below. Rotations in the camera mode are illustrated in figure 3.19.

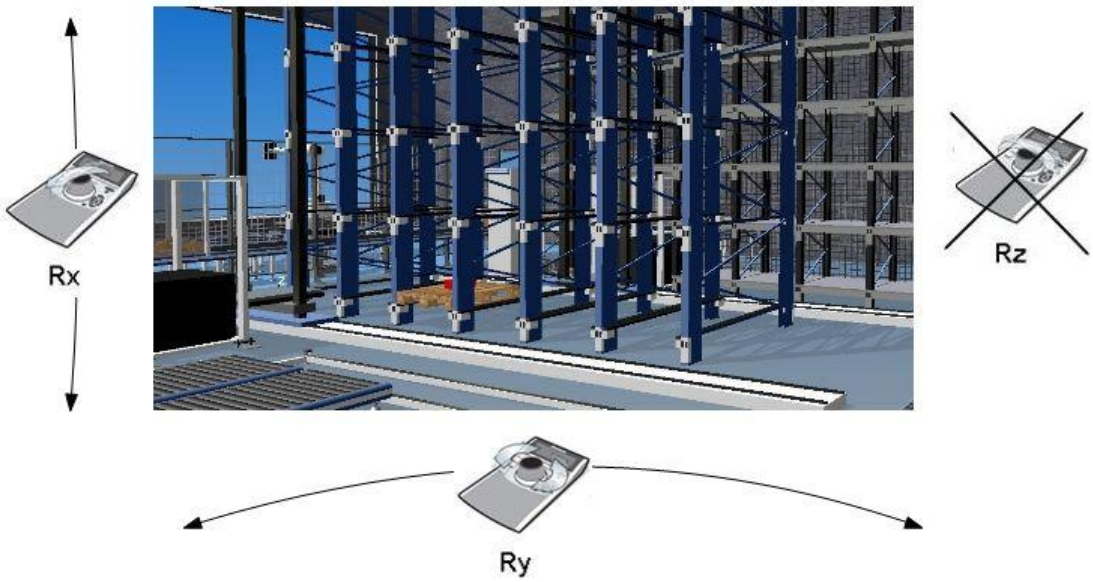


Figure 3.19. *Camera mode: Rotating the cap around its x axis rotates the camera around the screen coordinate system x-axis. Rotating the cap around its vertical y axis rotates the camera around the world coordinate system's vertical z-axis. Z rotation of the sensor is not used in the camera mode.*

Evaluation of the camera mode:

Positive observations:

- Camera mode works similar to the documentation of the SDK and demos.
- After learning how to use 3D mouse, the camera is easy to control with this mode.

Negative observations:

- When a traditional mouse is used at the same time with 3D mouse, 3D world draws selection frames around every component that the mouse focuses on. The rendering gets disturbed until the mouse control is stopped.
- It is not possible to change the directions of axes.

3.3.4 Object mode

Object mode is used for translating and rotating components in 3D world. It uses the raw input motion events of 3D mouse. Object mode is activated when user selects a component in 3D world. The manipulations are applied to the selected component. Object mode is deactivated by unselecting the component. Components are manipulated through the COM interface of 3D world. They can be both translated and rotated in three dimensions.

The motion event's translation vector is used in translating the component in 3D world. Components are translated in screen coordinate system. When the cap is pushed forwards, the selected object moves further away from the camera, while other parts of

3D world stand still. Respectively, when the cap is pushed backwards the object moves towards the camera. Object translation to left or right in screen coordinate system is done by pushing cap horizontally to respective direction. When the cap is pushed to left the component moves left in the screen coordinate system. The vertical movement of the component in the screen coordinate system is done by pushing and pulling the cap vertically. When the cap is pulled up, the component moves up in the screen coordinate system. Respectively when the cap is pushed down, the component moves down in the screen coordinate system. Translations in object mode are illustrated in figure 3.20.

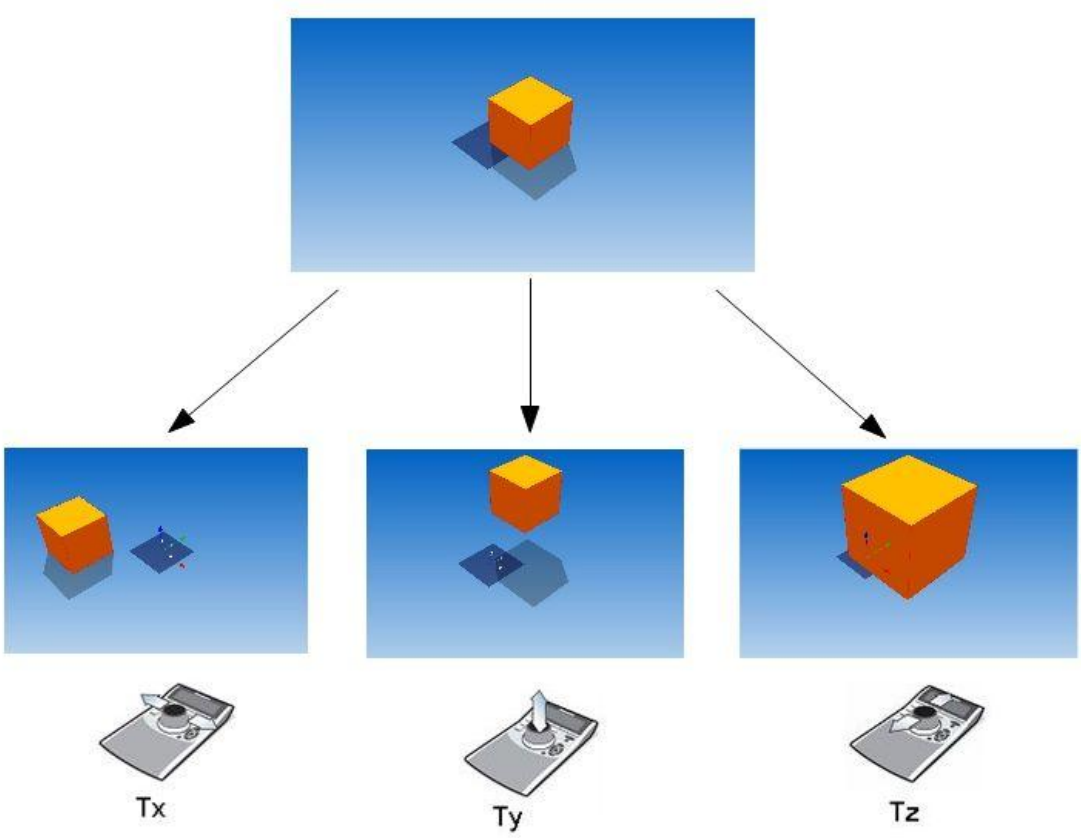


Figure 3.20. *Translations in the object mode: On the left, when the cap is translated to left, the block moves left in the screen coordinate system. In the middle, when the cap is pulled up, the block moves up in the screen coordinate system. On the right, when the cap is pulled backwards, the block moves forward in the screen coordinate system, which means closer to camera.*

The rotation vector of a motion event is used in rotating the component in around component's own coordinate system. Rotating the cap around its vertical axis causes the component to rotate around its own coordinate system's vertical axis. The speed of rotation is determined by the torque that the user uses to rotate the cap. For instance, as the cap is rotated left, the component rotates left as well. Tilting the cap around its horizontal axis rotates the component around its coordinate system's horizontal axis. Similarly,

tilting the cap around its depth axis rotates the component around its coordinate system's depth axis. Rotations in object mode are illustrated in figure 3.21.

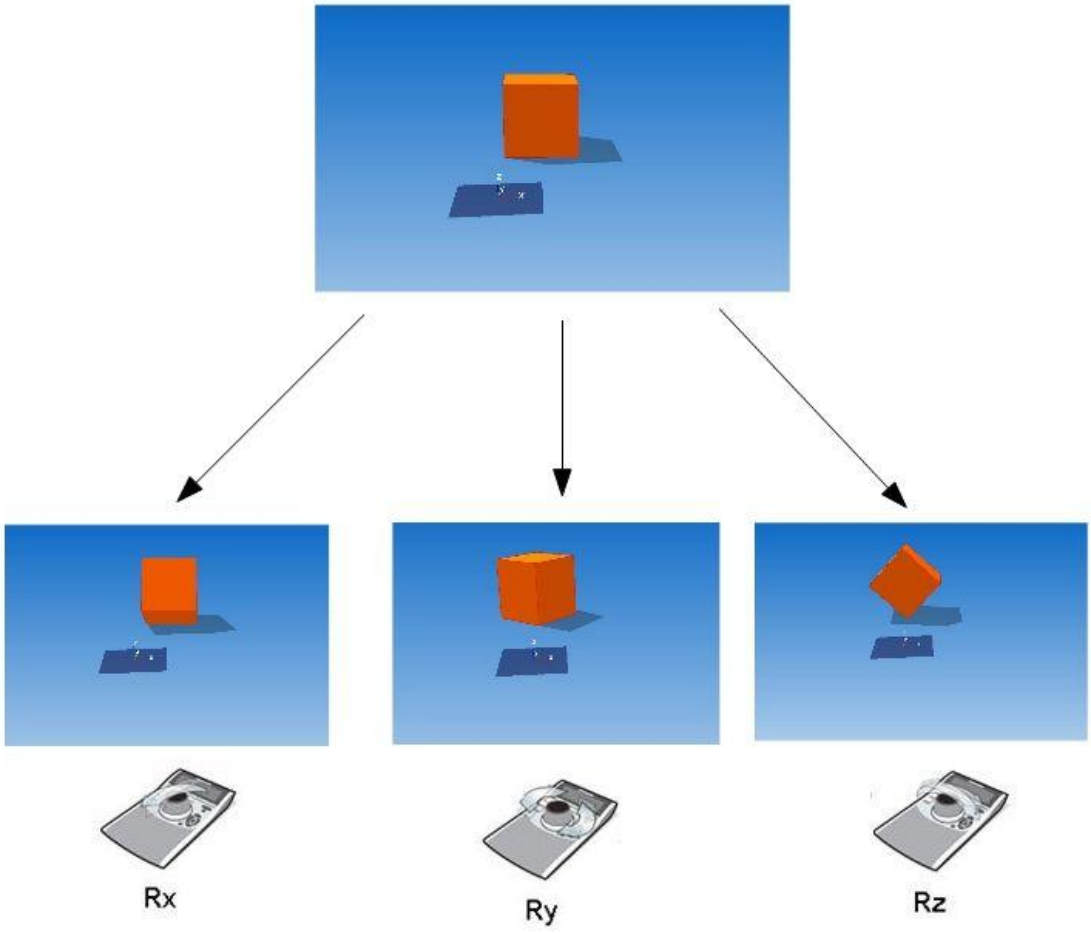


Figure 3.21. *Rotations in object mode: On the left, rotating the cap around its horizontal axis forwards rotates the block around its horizontal axis forwards. In the middle, rotating the cap around its vertical axis to left rotates the block around its own coordinate system's vertical axis to left. On the right, rotating the cap around its depth axis rotates the block around its own coordinate system's depth axis. Block's coordinate system's origin is located in its lower left corner.*

Evaluation of the object mode:

Positive observations:

- Translation works correctly in screen coordinate system.

Negative observations:

- Rotation is possible only in component's own coordinate system.
- Rotation origin is the component origin, which is located in a random location on a component. To be able to rotate component correctly, the rotation origin should be located always in the middle of the component.

3.3.5 Supporting functions

Every 3D mouse model has two standard buttons, fit and menu. [80] The fit button is generally used to reset the view. In this interface the fit button is used to activate the fill command, which centers the view so that every component fits in the view. Fill command is described in more detail in subchapter 2.4.3. The menu button generally opens a pop-up menu, and its contents and functions can be changed according to the needs of the software. Switching between functions could have been done through the menu, but currently it is not used, because the switching between functions is done by selecting and unselecting the components.

In the object mode the component is easily lost when it is quickly moved and rotated in 3D world. Another option to reset the view in object mode is to use the reset function. Reset function moves the selected component to the world origin and the camera to look to the world origin from a 40 degree angle from the ground. Reset function can be activated by quickly double tapping the top of the cap.

3.3.6 Future work

Some rendering issues occurred when 3D world was through COM interface. Sometimes the render command was called too early or too late in order to catch the correct phase of the cap movement. In the future, deeper integration of 3D mouse and 3D world should be considered. Deeper integration allows prioritizing 3D mouse events to according level. Rendering issues could be fixed when the 3D mouse events are handled inside the core of the system. Timing of render command can be changed so that it is called after all necessary actions are performed. Also the component frame drawing problem, which occurred when camera mode was used, could be fixed inside the core by canceling the frame drawing when the 3D mouse is used.

The issues with the object mode date back to the limitations of the COM interface of 3D world. Rotating components in the screen coordinate system requires converting the component's coordinate system to screen coordinate system. The representation of the object's orientation in 3D world through COM is yaw-pitch-roll. Yaw-pitch-roll representation in one coordinate system is hard to convert to another coordinate system. By using the object rotation matrix, which is used in the core, the conversion can be done easily. Problem with using the COM interface is that the object rotation matrices cannot be directly manipulated through it. By handling the events and manipulations in the core, the conversions could be done and the rotations could be performed correctly in the screen coordinate system.

Changing the coordinate system origin to the center of the component is difficult or impossible through the COM interface. It might be possible to change it in the core.

The possibility to use invert directions for axes in both camera and object mode should be added in the future. For example, pushing the cap forwards zooms outwards in invert camera mode. Also, the possibility to swap the mappings of vertical and depth movement could be added. For instance, in camera mode, pulling the cap up is mapped

to zoom in instead of upward pan. These options give the user the freedom to select the most familiar way to use his/her 3D mouse.

Work in the future includes adding the possibility to use plug and play functionality when the object mode is on. This is done by limiting the vertical movement of components and adding snap function when matching interfaces are close to each other. With plug and play function, building complete layouts could be done using the 3D mouse. The menu button could be used to activate this function.

One thing that should be considered in the future is the standardization of the 3D mouse. This is done through the certification program of 3DConnexion. [84] 3DConnexion helps in building the ideal implementation of the 3D mouse interface. When software is accepted to certification program of 3DConnexion, the experts will help with problems that developers are facing and share the knowledge and best practices in using the 3D mouse. By going through the certification program, software gets added to the listing of certificated software on the 3DConnexion website, the company gets permission to use the 3D mouse certification logo and a possibility to utilize cooperative marketing. [84]

4 DISCUSSION AND CONCLUSIONS

Digital manufacturing provides tools for modern design making process of manufacturing industries. The use of digital manufacturing software allows manufacturing companies to learn more about the process before actual manufacturing. It helps companies to avoid unforeseen problems, which eventually leads to increased productivity. Digital manufacturing reduces both time to market and product cost, lowers engineering changes to product design and reduces production tooling during launch. Digital manufacturing is done in an environment called 3D world.

Actions in 3D world are traditionally controlled with keyboard and mouse. However, the modern interaction devices enable more natural and intuitive types of human-computer interaction than the traditional input method with keyboard and mouse provide. Gesture-based interaction is one of the key concepts in utilizing the benefits of modern interaction devices. This thesis studied the possibilities of using modern interaction devices, namely touch screen, Microsoft Kinect and 3DConnexion, in interacting with 3D world.

The thesis studied interaction possibilities and experimented different types of interaction methods. Gesture-based control suits well for particular tasks in digital manufacturing software, but cannot replace the traditional interaction with keyboard and mouse. The usefulness, and general benefits and disadvantages of interacting with each of the devices are discussed later in this chapter.

This thesis was done in order to find out what the possibilities of interacting with these devices are, not to build a final implementation of these interfaces. The full human-centered design process involving actual end users could not have been used in this thesis due to time limitations and the nature of this thesis work. This thesis can be used as a basis for the future development process.

The interfaces for each of the devices were built and most suitable use cases were selected to be implemented to interface. As all the functions and options of 3D world were not available through COM interface, the full potential of these devices could not be achieved in some cases. Placing the interfaces inside the core of 3D world would fix most of the problems that occurred with devices as well as problems with rendering.

This MSc. thesis work succeeded generally well in achieving the set goals. However, the development process of the Kinect interface caused some troubles along the way. The schedule for building a functional interface was three months and after that the interface was presented in a large robotics fair. The learning process to use both Kinect and 3D world started from square one, and it took almost two months to use them in a useful way. This meant that the third and final month was dedicated to the actual devel-

opment of the interface. It turned out that the selection of Kinect as an interaction device was not considered properly.The suitability for the use in the fair or the actual use scenarios were not studied before the actual development process, and the results on the fair were not as good as were hoped. Leap Motion [85], for example, would have been more suitable device utilizing three dimensional gestures. It requires smaller space between the user and the device and it would have been more suitable for use at fair. The final implementation would have probably been better with more time on the actual planning and development part.

The learning work for 3D world was done in the Kinect part of this thesis which enabled more focused development on the other interaction devices. The development process went mostly as planned for them. COM interface caused some trouble along the way as some of the functions were not available through it. On the other hand, COM enabled the use of C# and WPF in the development, which made the programming much easier when compared to the original source code language C++.

Touch screen

Touch screen allows using finger as a pointer in applications. Multi-touch enables using multiple fingers simultaneously in interacting with computer environments. It is a requirement for the gesture-based control on touch screens, as most of the basic gestures require at least two fingers touching the screen to be able to be recognized. Gestures performed on the touch screen are called two dimensional gestures. The basic two dimensional gestures simulate real-life human actions and are performed with one hand. Two dimensional gestures consist of a certain amount of fingers touching the screen and performing defined movement on the screen. Basic movements are translation, rotation, scaling and tapping.

In order to find out the most suitable use cases for touch screen interface, three dimensional applications were tested with two different touch screen devices. The gestures were commonly used for controlling the user view and activating simple commands. Camera (user view) controls and basic object manipulations were implemented to the interface. As a result the two dimensional gesture-based interaction suits well for both controlling the camera and manipulating the position and the rotation of objects in 3D world. This interface could be directly implemented inside the core of 3D world, as it does not require any additional controls.

The implementation of touch screen interface worked well and everything that was planned was achieved with this interface. Touch interaction works well as a replacement for keyboard and mouse in defined environments and it will bring some benefits to users in certain use scenarios. The future improvements on the touch screen interaction would require changes in 3D world itself. In 3D world, making buttons and handles bigger and lowering the amounts of typing could be things to be considered in the future. Building first-person mode with the software joystick could be an interesting possibility for add-on in the future.

Microsoft Kinect

Microsoft Kinect enables using real-life three dimensional human gestures in interacting with computer environment. It allows humans to use the same kind of interaction with the computer as they do with each other. Other benefit of using three dimensional gesture-based interaction is that it allows the data and the interface to share the same three dimensional space. Defining starting and ending points of gestures as well as the differences in gestures between humans are the two main difficulties when using three dimensional gestures. The three dimensional gesture-based control is still a new area of research and the best practices and standards for using and recognizing gestures are constantly developing.

As there was no research directly involving digital manufacturing and the three dimensional gesture-based interaction, the building of the interface for Microsoft Kinect was exploratory. Three task sets were applied to the interface: camera controls, first person mode and menu. Camera controls and first person mode were both used to control the camera. The menu was used for activating commands through offline gestures. The first person mode was not as flexible as the normal camera controls were but it was faster and more natural to use than the regular camera controls were. Offline gestures worked well for activating commands. Different approaches were tried for each of the task sets, but the most natural and easiest way was achieved when the full body gestures combined with the pointing gestures were used. Generally Kinect and three dimensional gestures can be used in interacting with 3D world, but the space requirements and other limitations of Kinect and the interface itself limits its usage to really specific use scenarios.

Microsoft Kinect interface was implemented as a test of how the Kinect could be used in interacting with 3D world. The final implementation of the first person mode worked well, but considering the functions of the interface and the nature of vision-based interaction, the actual usage of this interface will be really rare. Gesture-only interfaces suit well for controlling smaller applications, but in 3D world's case, the software itself loses so much of its potential when only a little portion of it is in use through the gestures. The possibilities of Kinect interaction are high, but with this amount of development hours they could not be fully utilized. As a result of the previous thoughts, the development of this interface should be halted until some concrete research or application using this kind of interaction in an actually useful way exists.

3DConnexion SpacePilot PRO

3DConnexion's SpacePilot PRO is a movement controller, better known as 3D mouse. 3DConnexion provides a family of 3D mice and SpacePilot PRO is the product with the most supporting functions. The main function of all the 3D mice is the central 6 degree-of-free controller cap. The controller cap enables more natural control in 3D environ-

ments. 3D mice are commonly used in CAD environments to control the user view and manipulate object manipulations and rotations.

As CAD environments are similar to 3D world, the goal for this interface was to apply similar functionality to this interface that the most common CAD environments have. 3D mouse is used in two main ways in CAD environments: in camera mode; where the controller cap is used to control the movement of camera, and object mode; where the controller cap is used to translate and rotate objects in 3D space. These modes were applied to interface. Camera mode worked properly for controlling the camera in 3D world. Object mode translations worked equally well, but as the COM interface of 3D world limited the rotations to be used only in component's own coordinate system, some usability problems occurred.

Generally, building of the interface was successful. The only problem was the rotations, and it can be fixed if the interface is implemented inside the core of system as the required tools are available there. In the future, the 3DConnexion's certification program should be considered. The certification program helps in utilizing the 3D mouse in the best possible way, and it has other benefits as well. As a summary, the 3D mouse suits well for interacting with 3D world and the interface could be implemented inside the core of 3D with the fixes on rotations.

REFERENCES

[1] Siemens. Digital Manufacturing. [WWW]. [Accessed on 21.6.2011]. Available at: http://www.plm.automation.siemens.com/en_us/plm/digital-manufacturing.shtml.

[2] R. Brown. 2000. Driving digital manufacturing to reality. Simulation Conference, 2000. Proceedings. Winter, Orlando.

[3] T. Seino, Y. Ikeda, M. Kinoshita, T. Suzulu & K. Atsumi. 2001. The Impact of "Digital Manufacturing" on Technology Management. Management of Engineering and Technology, 2001. PICMET '01. Portland International Conference on, Portland, OR,

[4] V. I. Pavlovic, R. Sharma & T. S. Huang. 1997. Visual Interpretation of Hand Gestures, IEEE Transactions On Pattern Analysis and Machine Intelligence, vol. 19, no. 7.

[5] B. Buxton. 2011. Gesture Based Interaction. [WWW]. [Accessed on: 31.5.2012]. Available at: http://www.billbuxton.com/input14.Gesture.pdf.

[6] G. Kurtenbach & E. Hulteen. 1990. Gestures in Human-Computer Communications. B. Laurel (Ed.) The Art of Human Computer Interface Design, Addison-Wesley, pp. 309-317.

[7] M. B. Kaâniche. 2009. Human Gesture Recognition. PhD Thesis. Nice. University of Nice. 106p.

[8] W. Westerman. 1999. Hand tracking, finger identification, and chordic manipulation on a multi-touch surface. Delaware: University of Delaware.

[9] D. Archer. 1997. Unspoken Diversity: Cultural Differences in Gestures, QUALITATIVE SOCIOLOGY, vol. 20, no. 1, pp. 79-105.

[10] R. S.M.Alex, G. Sreelatha & M. Supriya. 2012. Gesture Recognition Using Field Programmable Gate Arrays. Devices, Circuits and Systems (ICDCS), 2012 International Conference on, Coimbatore.

[11] D. Kammer, M. Keck, G. Freitag & M. Wacker. 2010. Taxonomy and Overview of Multi-touch Frameworks: Architecture, Scope and Features. Engineering Patterns for Multi-Touch Interfaces.

[12] H. Ishii. 2008. Tangible Bits: Beyond Pixels. Proceedings of the Second International Conference on Tangible and Embedded Interaction (TEI'08), Bonn.

[13] B. Loureiro & R. Rodrigues. 2011. Multi-Touch as a Natural User Interface for Elders: A Survey. Information Systems and Technologies (CISTI), 2011 6th

Iberian Conference on, Porto.

[14] D. Wickeroth, P. Benolken & U. Lang. 2009. Markerless gesture based interaction for design review scenarios. Applications of Digital Information and Web Technologies, 2009. ICADIWT '09. Second International Conference on the, Koln.

[15] M. Karam, J. C. Lee, T. Rose, F. Quek & S. McCrickard. 2009. Comparing Gesture and Touch for Notification System Interactions. Second International Conferences on Advances in Computer-Human Interactions, Toronto.

[16] A. Lorenz, M. Jentsch, C. Concolato & E. Rukzio. 2010. A Formative Analysis of Mobile Devices and Gestures. MELECON 2010 - 2010 15th IEEE Mediterranean Electrotechnical Conference, Aachen.

[17] J. Guenther, F. Volk & M. Shaneck. 2010. Proposing a Multi-touch Interface for Intrusion Detection Environments. Proceedings of the Seventh International Symposium on Visualization for Cyber Security, New York.

[18] Microsoft. Windows Touch Gestures Overview. [WWW]. [Accessed on: 22.8.2012]. Available at: http://msdn.microsoft.com/en-us/library/windows/desktop/dd940543(v=vs.85).aspx.

[19] K. Kin, M. Agrawala & T. DeRose. 2009. Determining the benefits of direct-touch, bimanual, and multifinger input on a multitouch workstation. Proceedings of Graphics Interface 2009, Toronto.

[20] S. Knoedel & M. Hachet. 2011. Multi-touch RST in 2D and 3D spaces: Studying the impact of directness on user performance. 3D User Interfaces (3DUI), 2011 IEEE Symposium on, Bordeaux.

[21] C. Guo & E. Sharling. 2008. Exploring the Use of Tangible User Interfaces for Human-Robot Interaction: A Comparative Study. CHI '08 Proceedings of the twenty-sixth annual SIGCHI conference on Human factors in computing systems, New York.

[22] L. Gallo, A. Placitelli & M. Ciampi. 2011. Controller-free exploration of medical image data: Experiencing the Kinect. Computer-Based Medical Systems (CBMS), 2011 24th International Symposium on, Bristol.

[23] K. Nickel, E. Scemann & R. Stiefelhagen. 2004. 3D-tracking of head and hands for pointing gesture recognition in a human-robot interaction scenario. Automatic Face and Gesture Recognition, 2004. Proceedings. Sixth IEEE International Conference on.

[24] C. Yu, L. Jianzhuang & T. Xiaoou. 2008. Sketching in the air: A vision-based system for 3D object design. Computer Vision and Pattern Recognition, 2008. CVPR 2008. IEEE Conference on, Anchorage, AK.

[25] A. Bigdelou, T. Benz, L. Schwarz & N. Navab. 2012. Simultaneous categorical and spatio-temporal 3D gestures using Kinect. 3D User Interfaces (3DUI), 2012 IEEE Symposium on, Munich.

[26] P. O. Kristensson, T. F. Nicholson & A. Quigley. 2012. Continuous Recognition of One-Handed and Two-Handed Gestures using 3D Full-Body Motion Tracking Sensors. International Conference on Intelligent User Interfaces 2012, Lisbon.

[27] J. Lourenço & H. Thinyane. 2011. An evaluation of a low-cost 3-dimensional gestural interface: Wii3D. SAICSIT '11 Proceedings of the South African Institute of Computer Scientists and Information Technologists Conference on Knowledge, Innovation and Leadership in a Diverse, Multidisciplinary Environment, New York.

[28] S. Sreedharan, E. S. Zurita, Edmund & B. Plimmer. 2007. 3D input for 3D worlds. OZCHI '07 Proceedings of the 19th Australasian conference on Computer-Human Interaction: Entertaining User Interfaces, New York.

[29] M. Toma, P. C.C. & A. C. 2010. Multi-modal Interaction for 3D Modeling, Bulletin of the Transilvania University of Brasov, vol. III, pp. 137-144.

[30] Kuo, Chien-Fu & M.-J. J. Wang. 2012. Motion generation and virtual simulation in a digital environment, International Journal of Production Research.

[31] J. O. Coplien. Design Pattern Definition: Software Patterns. [WWW]. [Accessed on: 4.6.2012]. Available at: http://www.hillside.net/patterns/222-design-pattern-definition.

[32] S. Bajaj. 2001. Design Patterns: Solidify Your C# Application Architecture with Design Patterns, MSDN Magazine, no. 7.

[33] J. Maioriello. 2002. What Are Design Patterns and Do I Need Them?. [WWW]. [Accessed on: 30.5.2012]. Available at: http://www.developer.com/design/article.php/1474561/What-Are-Design-Patterns-and-Do-I-Need-Them.htm.

[34] M. A. Jalil & S. A. M. Noah. 2007. The Difficulties of Using Design Patterns among Novices: An Explanatory Study. Fifth International Conference on Computational Science and Applications, Washington, DC.

[35] The Hillside Group. Design Patterns Catalog. [WWW]. [Accessed on: 5.6.2012]. Available at: http://www.hillside.net/patterns/patterns-catalog.

[36] W. Cunningham. Portland Pattern Repository. [WWW]. [Accessed on: 5.6.2012]. Available at: http://c2.com/ppr/index.html.

[37] P. Avgeriou & U. Zdun. 2005. Architectural Patterns Revisited – A Pattern Language.

[38] A. Seffah. 2010. The Evolution of Design Patterns in HCI: From Pattern Languages to Pattern-Oriented Design. PEICS '10 Proceedings of the 1st International Workshop on Pattern-Driven Engineering of Interactive Computing Systems, New York.

[39] C. Kruschitz. 2009. The Anatomy of HCI Design Patterns. Future Computing, Service Computation, Cognitive, Adaptive, Content, Patterns, 2009.

COMPUTATIONWORLD '09. Computation World:, Athens.

[40] Microsoft. 2010. 10262A Developing Windows Applications with Microsoft Visual Studio 2010, vol. I, Microsoft.

[41] Microsoft. Model-View-Controller. [WWW]. [Accessed on: 29.5.2012]. Available at: http://msdn.microsoft.com/en-us/library/ff649643.

[42] Sun Microsystems. Java BluePrints: Model-View-Controller. [WWW]. [Accessed on: 20.6.2012]. Available at: http://java.sun.com/blueprints/patterns/MVC-detailed.html.

[43] Microsoft. 2008. Using the Model-View-Presenter (MVP) Design Pattern to enable Presentational Interoperability and Increased Testability. [WWW]. [Accessed on: 29.5 2012]. Available at: http://blogs.msdn.com/b/jowardel/archive/2008/09/09/using-the-model-view-presenter-mvp-design-pattern-to-enable-presentational-interoperability-and-increased-testability.aspx.

[44] Microsoft. 2006. Model View Presenter, MSDN Magazine, no. 8.

[45] M. Fowler. Passive View. [WWW]. [Accessed on: 20.6.2012]. Available at: http://martinfowler.com/eaaDev/PassiveScreen.html.

[46] M. Fowler. Supervising controller. [WWW]. [Accessed on: 20.6.2012]. Available at: http://martinfowler.com/eaaDev/SupervisingPresenter.html.

[47] J. Smith. 2009. WPF Apps With The Model-View-ViewModel Design Pattern, MSDN Magazine, no. 2,

[48] N. Polyak. 2011. MVVM Pattern Made Simple. [WWW]. [Accessed on: 30.5.2012]. Available at: http://www.codeproject.com/Articles/278901/MVVM-Pattern-Made-Simple.

[49] D. Benyon, P. Turner & S. Turner. 2004. Designing Interactive Systems: People, Activities, Contexts, Technologies, Addison Wesley.

[50] J. Preece, Y. Rogers & H. Sharp. 2002. Interaction design: Beyond human-computer, New York: John Wiley & Sons, Inc.

[51] Microsoft. Component Object Model (COM). [WWW]. [Accessed on: 28.5.2012]. Available at: http://msdn.microsoft.com/en-us/library/ms680573(v=vs.85).

[52] Microsoft. The Component Object Model. [WWW]. [Accessed on: 28.5.2012]. Available at: http://msdn.microsoft.com/en-us/library/ms694363(v=vs.85).

[53] Microsoft. COM Clients and Servers. [WWW]. [Accessed on: 29.5.2012]. Available at: http://msdn.microsoft.com/en-us/library/ms683835(v=vs.85).

[54] XML UK. .NET History and Information. [WWW]. [Accessed on: 21.5.2012]. Available at: http://www.xmluk.org/net-history-information.htm.

[55] Microsoft. Creating Components in .NET. [WWW]. [Accessed on: 4.9.2012]. Available at: http://msdn.microsoft.com/en-us/library/ms973807.aspx.

[56] Microsoft. 2010. .NET Framework Versions and Dependencies. [WWW].

[Accessed on: 21.5.2012]. Available at: http://msdn.microsoft.com/en-us/library/bb822049.aspx.

[57] Microsoft. .NET Framework Conceptual Overview. [WWW]. [Accessed on: 21.5.2012]. Available at: http://msdn.microsoft.com/en-us/library/zw4w595w.

[58] P. Faraday & B. Becker. Deciding when to adopt Windows Presentation Foundation. [WWW]. [Accessed on: 22.5.2012]. Available at: http://windowsclient.net/wpf/white-papers/when-to-adopt-wpf.aspx.

[59] C. Mosers. 2011. Introduction to Windows Presentation Foundation. [WWW]. [Accessed on: 22.5.2012]. Available at: http://www.wpftutorial.net/WPFIntroduction.html.

[60] Microsoft. XAML Syntax In Detail. [WWW]. [Accessed on: 22.5.2012]. Available at: http://msdn.microsoft.com/en-us/library/ms788723.

[61] Microsoft. WPF Architecture. [WWW]. [Accessed on: 22.5.2012]. Available at: http://msdn.microsoft.com/en-us/library/ms750441.aspx.

[62] Microsoft. 2010. C# Language Specification Version 4.0.

[63] Microsoft. Introduction to the C# Language and the .NET Framework. Microsoft, [WWW]. [Accessed on: 23.5.2012]. Available at: http://msdn.microsoft.com/library/z1zx9t92.

[64] W. Zeng. 2012. Microsoft Kinect Sensor and Its Effect. Multimedia, IEEE , vol. 19, no. 2, pp. 4-10.

[65] Adafruit industries. 2010. We have a winner – Open Kinect driver(s) released – Winner will use $3k for more hacking – PLUS an additional $2k goes to the EFF!, [WWW]. [Accessed on: 23.5.2012]. Available at: http://www.adafruit.com/blog/2010/11/10/we-have-a-winner-open-kinect-drivers-released-winner-will-use-3k-for-more-hacking-plus-an-additional-2k-goes-to-the-eff/.

[66] Microsoft. 2011. Microsoft Releases Kinect for Windows SDK Beta for Academics and Enthusiasts. [WWW]. [Accessed on: 23.5.2012]. Available at: http://www.microsoft.com/en-us/news/press/2011/jun11/06-16MSKinectSDKPR.aspx.

[67] Kinect for Windows Team. 2012. Starting February 1, 2012: Use the Power of Kinect for Windows to Change the World. [WWW]. [Accessed on: 23.5.2012]. Available at: http://blogs.msdn.com/b/kinectforwindows/archive/2012/01/09/kinect-for-windows-commercial-program-announced.aspx.

[68] Microsoft. Skeletal tracking. [WWW]. [Accessed on: 24.5.2012]. Available at: http://msdn.microsoft.com/en-us/library/hh973074.

[69] M. Buchanan & J. Johnson. 2010. Deep Inside Xbox 360 Kinect and Why It's the Future of Microsoft. [WWW]. [Accessed on: 24.5.2012]. Available at:

http://gizmodo.com/5604308/deep-inside-xbox-360-kinect-the-interface-of-microsofts-dreams.

[70] D. Gantenbein. 2011. Helping Kinect Recognize Faces. [WWW]. [Accessed on: 23.5.2012]. Available at: http://research.microsoft.com/en-us/news/features/kinectfacereco-103111.aspx.

[71] Microsoft. Natural User Interface for Kinect for Windows. [WWW]. [Accessed on: 24.5.2012]. Available at: http://msdn.microsoft.com/en-us/library/jj131023.

[72] N. Cohen. 2011. Timeline: A History Of Touch-Screen Technology. [WWW]. . [Accessed on: 25.5.2012]. Available at: http://www.npr.org/2011/12/23/144185699/timeline-a-history-of-touch-screen-technology

[73] J. Prader. Touch Screen Definition. [WWW]. [Accessed on: 24.5.2012]. Available at: http://www.ehow.com/about_6757418_touch-screen-definition.html.

[74] D. Braue. 2011. Inside touch: How touchscreen technologies work. [WWW]. [Accessed on: 24.5.2012]. Available at: http://apcmag.com/inside-touch-how-touchscreen-technologies-work.htm.

[75] Planar Systems, Inc.. Touchscreen Technology for Displays & Monitors. [WWW]. [Accessed on: 27.8.2012]. Available at: http://www.planarembedded.com/technology/touch/.

[76] MaximumPC. 2008. White Paper: Touch-Screen Technology. [WWW]. [Accessed on: 25.5.2012]. Available at: http://www.maximumpc.com/article/white_paper_touch_screen_technology.

[77] 3DConnexion. SpacePilot PRO: The Ultimate Professional 3D Mouse. [WWW]. [Accessed on: 28.5.2012]. Available at: http://www.3dconnexion.com/products/spacepilot-pro.html.

[78] 3DConnexion. SpacePilot PRO: The Ultimate Proffessional 3D Mouse: Feature Guide. [WWW]. [Accessed on: 28.5.2012]. Available at: http://www.3dconnexion.com/nc/media.html.

[79] C. Dias, O. Fernandes, A. Cunha & L. Morgado. 2011. Planning of a usability test for 3D controllers in Second Life / OpenSimulator virtual worlds. Serious Games and Applications for Health (SeGAH), 2011 IEEE 1st International Conference on.

[80] 3DConnexion. 2012. 3DxWare SDK 3.0 for Windows (x86 & x64).

[81] A. Tsao. 2010. Multi-touch in WPF 4 Part 3 – Manipulation Events. [WWW]. [Accessed on: 22.8.2012]. Available at: http://blogs.msdn.com/b/ansont/archive/2010/01/11/multi-touch-in-wpf-4-part-3-manipulation-events.aspx.

[82] R. A. Stebbins. 2001. Exploratory Research in the Social Sciences, Thousand Oaks, CA: Sage Publication, Inc.

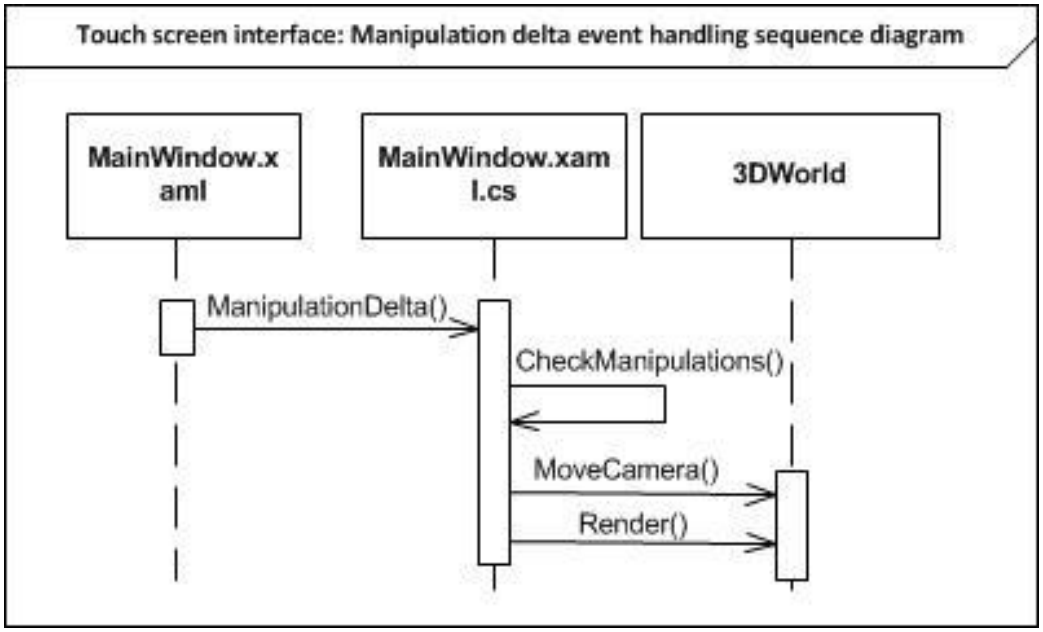
[83] Deltakosh. Codeplex: Kinect toolbox. [WWW]. [Accessed on: 16.7.2012].

Available at: http://kinecttoolbox.codeplex.com/.

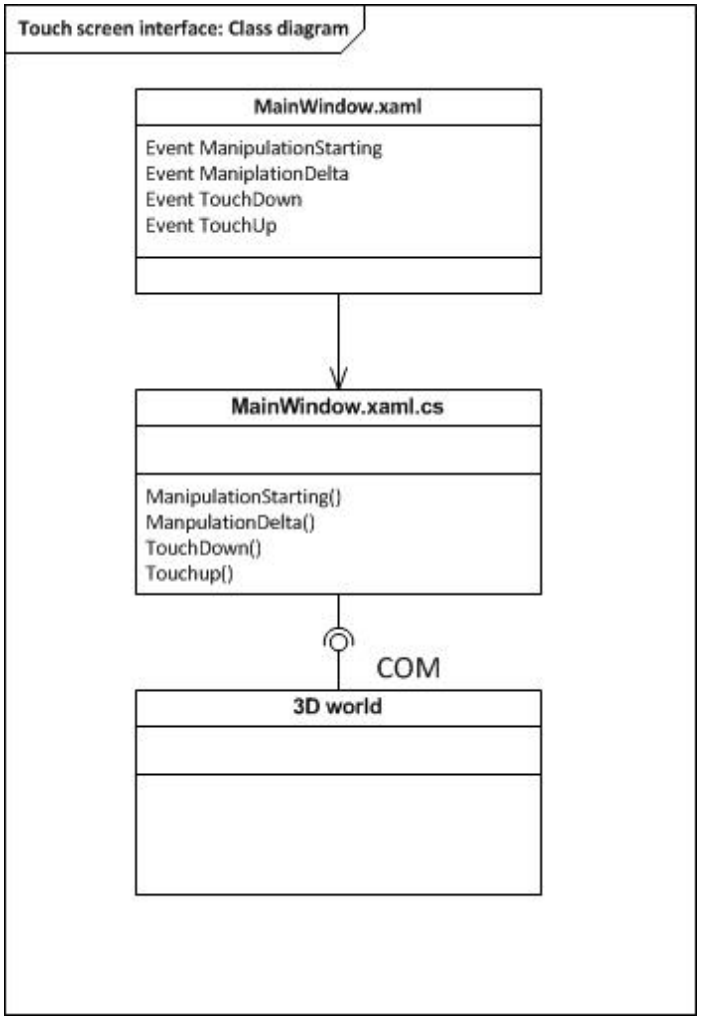
[84] 3DConnexion. 3DConnexion Certification Program. [WWW]. [Accessed on: 19.7.2012]. Available at: http://www.3dconnexion.com/company/3dx-certification-program.html.

[85] Leap Motion, Inc. Leap Motion. [WWW]. [Accessed on: 2.10.2012]. Available at: https://leapmotion.com/.

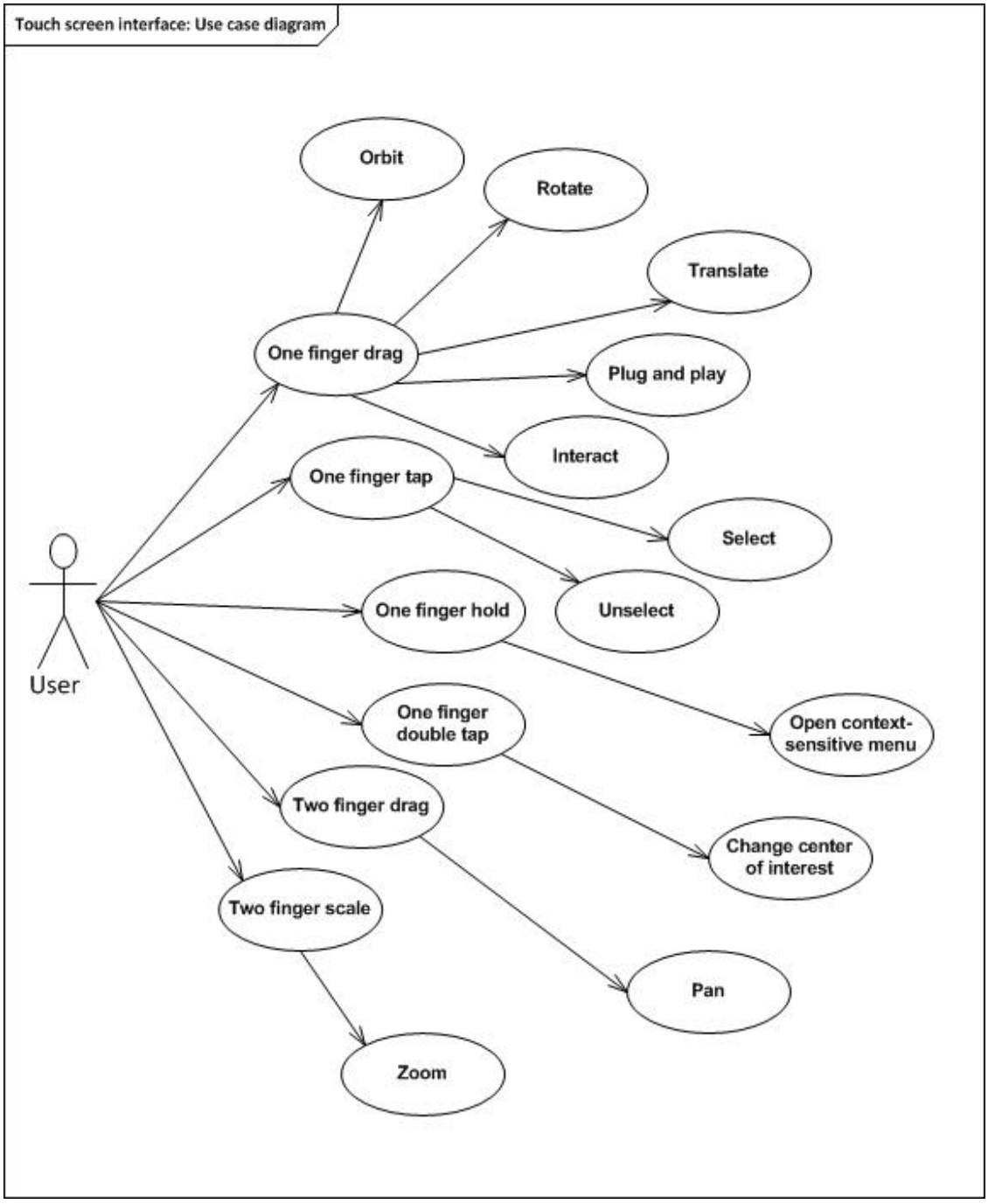
APPENDIX 1: TOUCH SCREEN INTERFACE: MANIPULATION DELTA SEQUENCE DIAGRAM



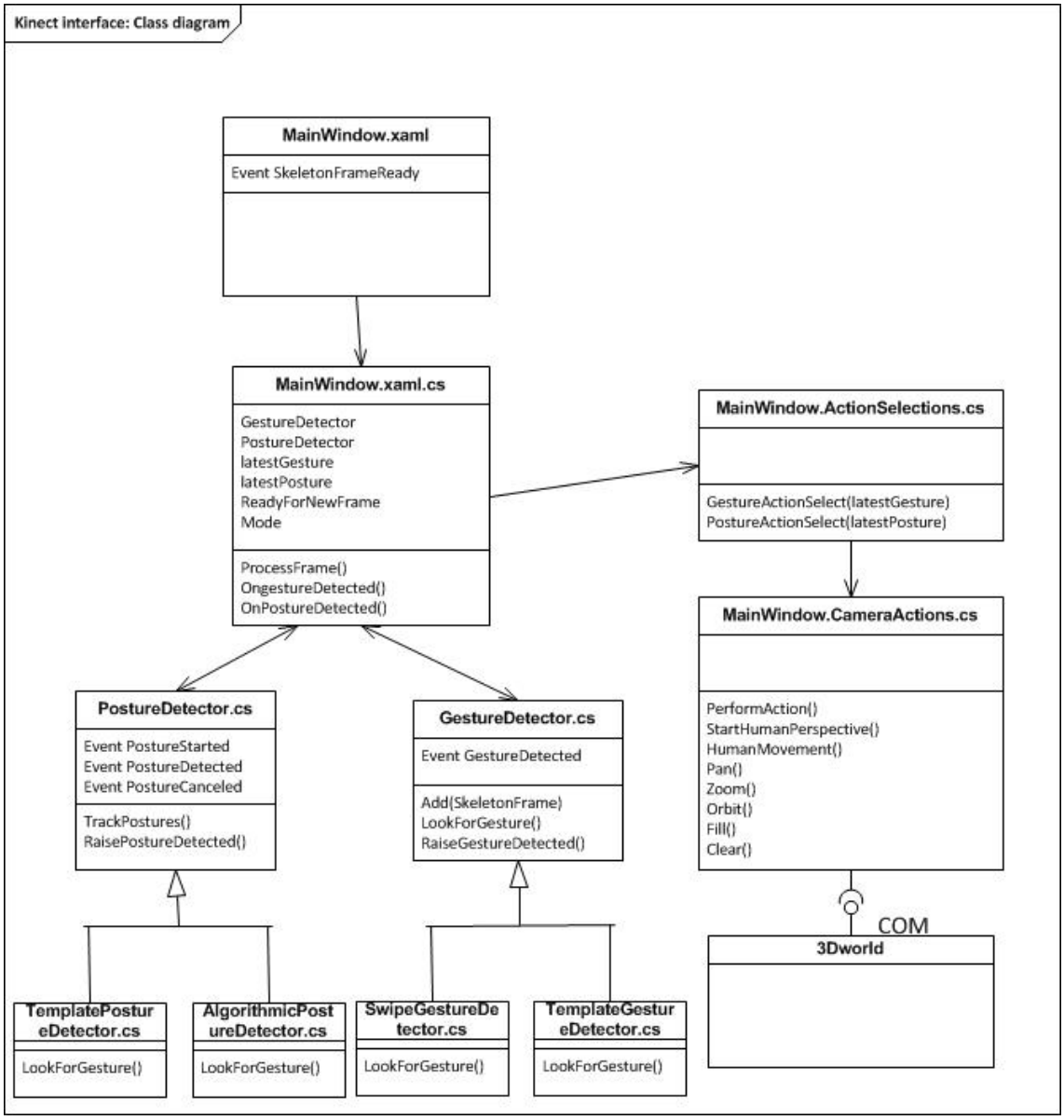
APPENDIX 2: TOUCH SCREEN INTERFACE: CLASS DIAGRAM



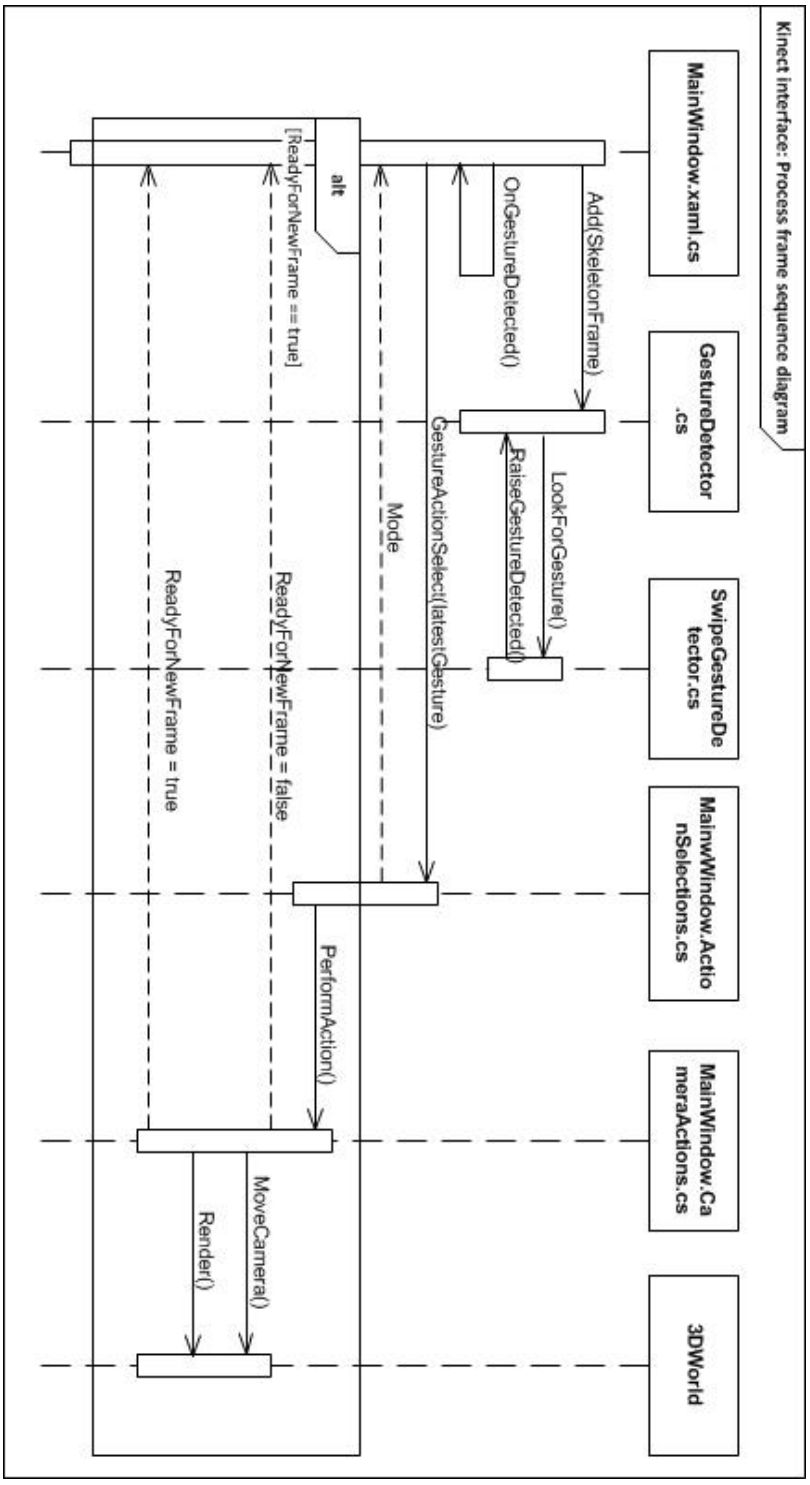
APPENDIX 3: TOUCH SCREEN INTERFACE: USE CASE DIAGRAM



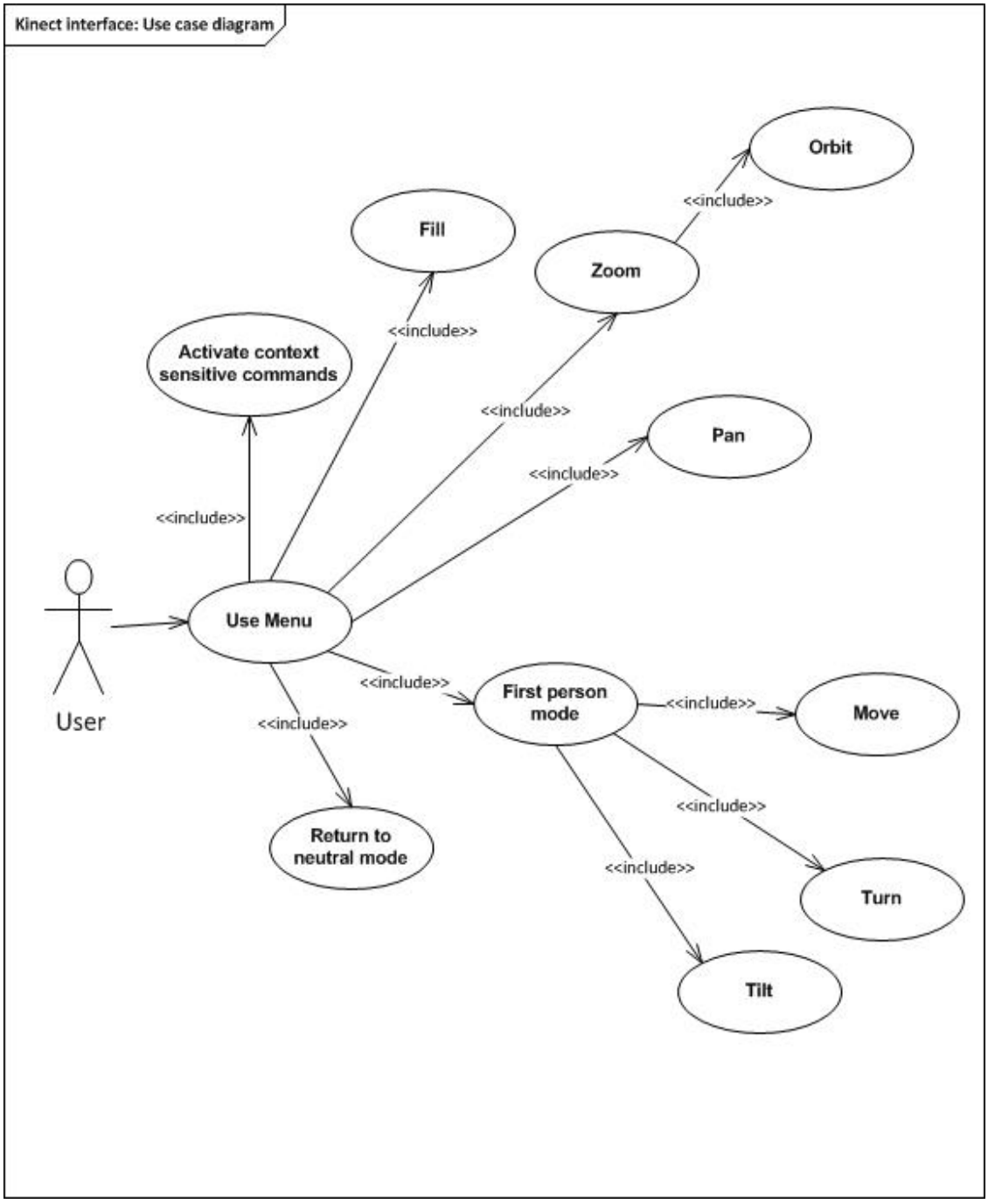
APPENDIX 4: KINECT INTERFACE: CLASS DIAGRAM



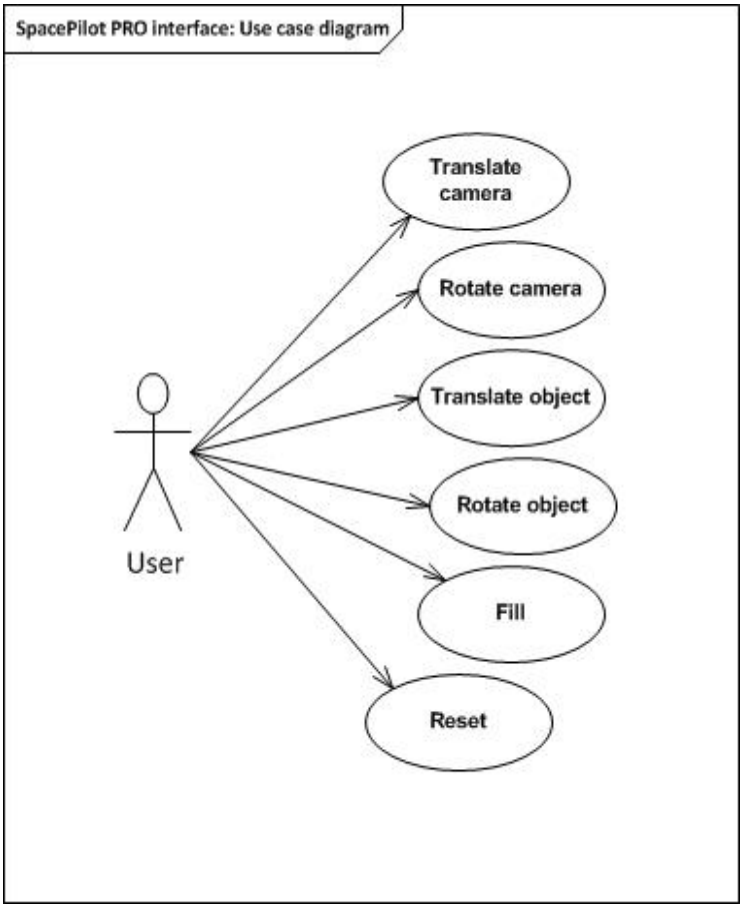
APPENDIX 5: KINECT INTERFACE: PROCESS FRAME SEQUENCE DIAGRAM



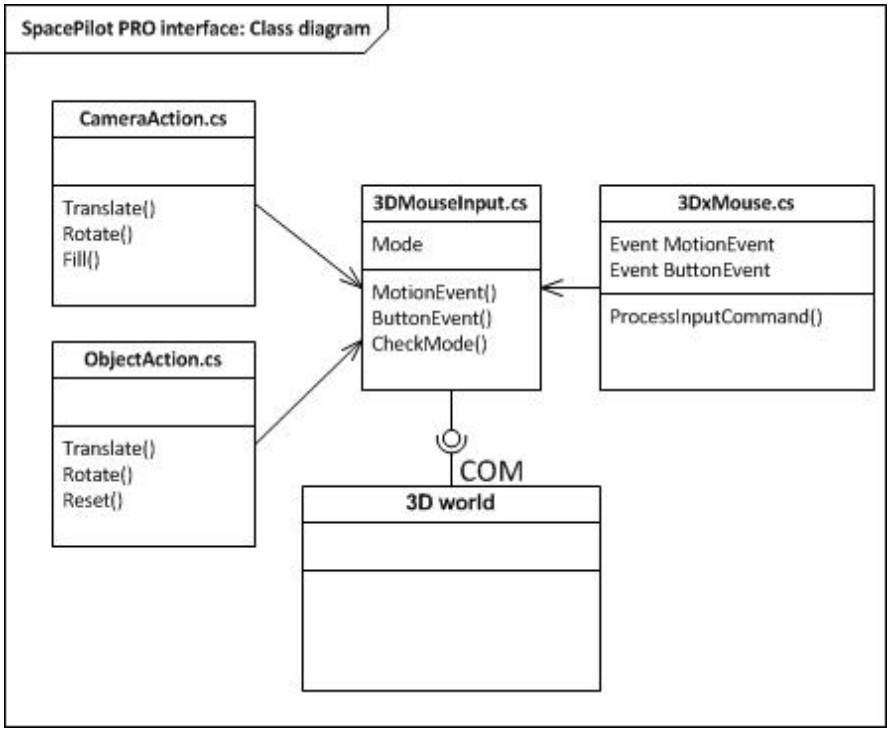
APPENDIX 6: KINECT INTERFACE: USE CASE DIAGRAM



APPENDIX 7: SPACEPILOT PRO INTERFACE: USE CASE DIAGRAM



APPENDIX 8: SPACEPILOT PRO INTERFACE: CLASS DIAGRAM



APPENDIX 9: SPACEPILOT PRO INTERFACE: PROCESS INPUT COMMAND SEQUENCE DIAGRAM