Aalborg University Copenhagen

Department of Medialogy

Aalborg University Copenhagen
Lautrupvang 15, 2750 Ballerup,
Denmark

Semestercoordinator: Stefania Serafin
Secretary: Lisbeth Nykjær
Phone: 9940 2471
lny@create.aau.dk
https://internal.media.aau.dk/

Semester: MED 10

Title: Gesture Based Presentation Tool

A comparison of interaction methods in Slideshow presentations.

Project period: Spring 2011

Semester theme: Master Thesis

Supervisor(s): Bob L. Sturm

Project group no.:

Members:
Heino Jørgensen

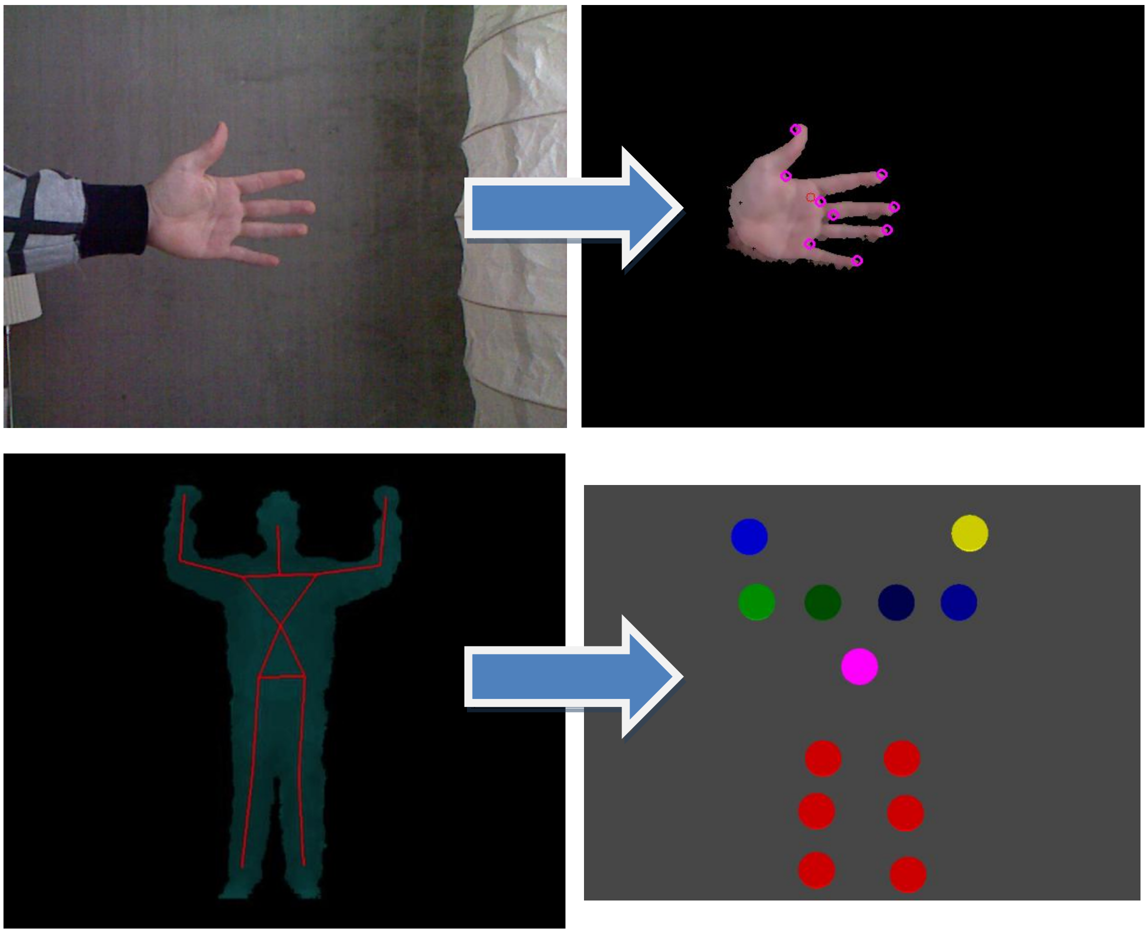
Abstract:

This project focuses on using natural gestures to control a slideshow presentation. Using the Microsoft Kinect an approach is given for tracking the users hand and fingers, in order to supply for a gesture based interaction with the presentation. The method proposed for finger tracking uses skin color detection combined with feature extraction of fingertips, implemented using OpenCV in combination with OpenKinect. For the final experiment though, an application only utilizing hand tracking was implemented, due to the instability of the proposed finger tracking. The final application is implemented in Unity, using the NITE library from OpenNI. It allows a presenter to move through a presentation by swiping as well as highlighting text and objects on the slides. The final experiment tested the audience perception of the gesture based presentation, compared to a traditional presentation using mouse and keyboard. The test showed that while the gesture based presentation was neither better nor worse for conveying information to the audience, it seemed to distract the audience more, because of the presenter's movement.

Copies: 3
Pages: 72
Finished: 26-05-2011

Gesture Based Presentation Tool

A comparison of interaction methods in slideshow presentations



AALBORG UNIVERSITET

Author: Heino Jørgensen

Supervisor: Bob L. Sturm

Preface

This project has been developed during the 10th semester of Medialogy at Aalborg University Copenhagen, in the period from the 1st of February 2011 to the 27th of May 2011.

The product consist of two parts: A finger tracking algorithm implemented in C++ and an application developed in Unity using skeleton tracking. The report will document and discuss the steps of the production process that the author went through during the project period. The report will cover topics such as natural gestures, skin color detection, feature extraction and use of Microsoft Kinect.

Reader's Guide

Every main chapter in this report will contain a small description of what the given chapter will cover as well as how it is structured.

The APA Quotation standard is used when referring to sources and quoting: "To be, or not to be, that is the question" (Shakespeare, 1603). When a source is used directly in the text, such as: "(Shakespeare, 1603) is one of the great pieces of literature in history", it means that it should be read as a part of the sentence, but still as a reference to the content of the source. The full list of sources can be found in *12 Bibliography*.

When referencing to other chapters or illustrations the text is italic, as it can be seen in the line above.

Illustrations are numbered X.Y, where X is chapter number and Y is the number of the illustration in the specific chapter. Illustration text is easily distinguishable because the font is blue and the font size is smaller than the body text, which is True type font "*Times New Roman*" in size 11.

Code examples and variables are formatted with `Courier New` and the background is blue. Furthermore when multiple lines of code are represented, a line number can be found on the left side.

13 Appendix contains the questionnaires used for testing as well as some of the test results. The full test results spread sheet and source code for the developed applications, can be found on the enclosed CD-Rom. In the folder called "Applications" on the CD-Rom, both of the developed applications can be found, ordered in the folder "Finger" and "Skeleton" respectively. For the finger tracking approach the source code is supplied directly as well as a PDF file containing the code formatted for easy reading. For the skeleton tracking approach the entire Unity project is supplied, which will require a copy of Unity to open. However code source files can be found in the folder "/Assets/Scripts" and a PDF file containing a well formatted copy of the code is also supplied in this folder.

Table of Content

1. Introduction

Within the last decade, more and more companies have been working on developing game consoles with very physical interfaces. First the EyeToy from Sony, then later on the Nintendo Wii and the most recent approaches are the Microsoft Kinect and the Move system from Sony. The systems are making players more and more accustomed to using physical interfaces to control their games, but so far the use of these systems have been limited outside the realm of games. This project will look into other usages of physical games systems like the ones mentioned, more specifically how to use them in a presentation context. The project will look into how to create a gesture based presentation tool and compare this to a traditional presentation in order to test if the watching audience will prefer one over the other.

This chapter will be an introduction to the general topic of the project and the motivation for working with the topic of gesture based control methods, both in and out of game applications.

1.1 Motivation

In the systems mentioned in the introduction, there are different technological approaches to achieve the desired control methods. The Wii and the Playstation Move are both using controllers to track the player's movement, although in different ways. The Kinect and Playstation's Eye Toy are using cameras to track the players and thus there are no controllers involved. The way these systems works technologically will be covered in more details in *2.2 Human tracking hardware*, but despite their differences in hardware and software, they share a common trait in the play experience: They all require a very active and physical interaction. In a gaming environment this form of interaction supplies for a physical engagement in the games that is not present in traditionally controlled games.

This form of engagement might have been part of the success these new game consoles have had, but maybe a physically engaging interface will also improve users' experiences with other applications than games. A classic example of such an interface is the one seen in the movie *Minority Report* (Spielberg, 2002), in which the main character is seen operating a – quite complex – application using gestures only. However, such an interface might not be suitable in an actual real world application. Some of the most obvious problems are how to isolate the user's gestures in a way that they will only interact with the application when intended and as intended. Another problem is that of fatigue in the user. If working a purely gesture based interface, it might quickly become both tiresome and exhausting for the user. Despite these challenges there have already been made several projects, which are trying to utilize the before mentioned hardware to produce applications with interfaces similar to the *Minority Report* example. For instance, the Wii Controller Interface Suit, or just Wiici, is a collection of tools for connecting the Wii controller with a PC, hereby making it possible to interact with a traditional Microsoft Windows system for instance (Wicii, 2010). Also the Kinect is being utilized outside of the gaming environment and the German company Evoluce recently released software that is utilizing the Kinect to produce a gesture based interaction with Microsoft's Windows 7 (Evoluce, 2010). Furthermore, there are also popular open source libraries such as OpenKinect (OpenKinect, 2011), which makes it possible to easily develop applications for a PC, using the Kinect hardware.

One of the reasons why it is becoming so popular to develop for these types of hardware and why there are so many projects popping up, based on motion tracking devices, is probably because of the low price and availability of these products. 10 years ago, before even the Eye Toy was released, it was both cumbersome and expensive to do interactions based on gestures. It was possible to do image processing on feeds from webcams through open source libraries such as OpenCV (OpenCV, 2011), released in 1999, but the webcams back then was usually providing low resolution and poor frame rate, at least on the webcams that were affordable to the average user. Furthermore, it would require complicated programming algorithms to acquire depth information from a regular webcam. Another option was to use special data gloves that would track the users' hands, but such gloves were (and are) expensive, which means not many users would buy the product. Such a glove is also an intrusive piece of hardware, because the user will have to wear the glove and most likely also be connected with wires to the hardware that is running the application. With the EyeToy a cheap and reliable piece of hardware were introduced to the market, which made it possible to produce gesture based interfaces to a wide audience. The Wii console further expanded the possibilities and with the Kinect and Playstation Move it became easy to obtain information about the users' movements and their position in space.

With this hardware available, at prices that allow casual users to buy it, it becomes interesting to investigate how to use the hardware also in non-gaming applications. As mentioned, it might not be suitable to make a purely gesture based interface for everyday tasks, such as browsing web pages or writing documents, since it will quickly make the user tired and because these tasks require an extensive list of actions that would be hard to achieve by gestures only. However, work has been done in this area and the work of German company Evoluce, mentioned previously in this chapter, did make it possible to both browse web pages and writing documents, through an onscreen keyboard controlled by gestures. Already back in 1995, work was done to produce a gesture based system for the everyday task of controlling your television (Freeman & Weissman, 1995). Freeman & Weismann managed to set up a system that would track simple gestures made by the users, to control i.e. the volume of the television. Although the system worked, it was also reported that some of the gestures "*was found to be somewhat tiring for extended viewing*" (Freeman & Weissman, 1995).

There are tasks though for which it might be prudent to use gestures and where gestures could perhaps even improve the way these tasks are performed. When doing presentations, the presenter will often be using hand gestures while talking and running through the slides of a slideshow. When doing such a presentation the traditional interfacing, using keyboard and mouse to move through the slides, might be a poorer interaction than what could be achieved using gestures. Several pieces of work have been done in this area too, ranging from simply using gestures to control the computer mouse (Argyros & Lourakis, 2006) to applications using gestures to directly influence a slide show (Baudel & Beaudouin-Lafon, 1993) (Husz, 2004). As seen from some of the research in this area, gesture recognition has been of interest to the academic environment for several years and presentations of all sorts are an important part of our lives today. In teaching, workshops, business propositions, etc. presentations are being used to convey a message in a simple and direct manner. The following chapter will look into the development of presentations and the technology surrounding them, in order to give an impression of what the technology has done to presentations today and how they can be further improved.

1.2 Related work

One of the first professional, digital presentation tools were the program Harvard Presentation Graphics (Harvard Graphics, 2011), which could be used to create slides, charts and graphs that could then be printed as transparencies. The first version of Harvard Presentation Graphics was published in 1986, at a time where computers and projectors were not easily available. The printed transparencies had to be placed on an overhead projector, which made it cumbersome to switch back and forth between slides, as well as remembering to put the slides on the projector with the right orientation, such that the content would not be mirrored or flipped upside down.

Only one year after the release of Harvard Presentation Graphics, Microsoft bought the rights for a presentation tool called Presenter, which was developed by Bob Gaskins and Dennis Austin, PhD students at the University of California-Berkeley. This program was later renamed to PowerPoint, due to copyright issues, and it is this software that is most widely known and used for presentations today (Hewitt, 2008). PowerPoint gains its strength by the use of PCs and projectors, which eliminates a lot of the problems that people were facing when using transparencies.

PowerPoint is traditionally controlled by mouse and keyboard where the presenter uses these traditional input devices to i.e. change slides or highlight things in the presentation. Through the years, other alternatives such as laser pointers and remote controls have been made to control the slide shows, but they are all more or less intrusive pieces of hardware that requires the presenter to still keep track of buttons on the remote or keep a very steady hand when using the laser pointer. Alternative approaches for interacting with presentations have been proposed, as described in some of the works mentioned previously in this chapter. These approaches most often includes hand gestures or speech recognition for controlling the slideshows, as in the work done by Baudel & Beaudouin-Lafon, in which they made an application for controlling a slideshow on a Macintosh computer, purely by hand gestures. However, their application was using a data glove and they note in their conclusion that "*The main problem remains the use of a DataGlove*" (Baudel & Beaudouin-Lafon, 1993) and they also report that their work could be improved by the use of video cameras for tracking the hands. However, they also mention that no matter what technology you use, the problem of the presenter having an instable hand is going to be present. Any little motion will be amplified by the system and it is something to consider when doing any sort of remote gestural interaction. This problem is also mentioned in the work done by Sukthankar, Stockton & Mullin about using a laser pointer or other pointing device for interacting with a slide show (Sukthankar, Stockton, & Mullin, 2001). In their work, it is possible to use your finger as a pointing device, tracked by the camera, hereby avoiding the use of a data glove. It is possible in this application to use your finger as a pointing device and drawing device. If the presenter hovers his finger over an area in the slide that is identified as a button, the activity mapped to that button will be executed. Furthermore, they also make it possible for the user to draw on the slides, although this action required the user of a laser pointer. When drawing, the instability of the presenters hand will again become an issue and Sukthankar et al. solves this by smoothing the motion, which according to their conclusion is a viable solution. What the system by Sukthankar et al. is lacking is the use of gesture recognition. Their system is limited to pointing, which as described requires the slide show to have buttons for performing the neccesary actions.

Several papers adress the issue of hand and gesture recognition, although not implenting it in a system like what Baudel & Beaudouin proposed in their 1993-paper. (Elmezain, Al-Hamadi, Appenrodt, & Michaelis, 2008) made an application that would track a user's hand motion and be able to identify the Arabic numbers (0-9) based on this motion. Elmezain et al. also uses a depth segmentation of the tracked image to be able to more easily identify the areas of importance for tracking. A method that is now easily available through the hardware of i.e. the Kinect. (Schlömer, Poppinga, Henze, & Boll, 2008) made use of another of the previously mentioned gaming consoles – the Wii – to do gesture recognition. The Wii gives input about the user's motion of his hand without the need to do any camera based tracking, but when using the Wii it is not possible to do tracking of the user's fingers. Only the general position and movement of his hand can be tracked. What Schlömer et al. did in their system though was to make it possible for users to train the system to recognize new gestures, hereby not limiting the system to a specific usage. In (Malik, 2003) a thorough description of a hand and finger tracking system is given, explaining how to use the OpenCV library to do gesture recognition. However, as with the study by Elmezain et al., the system was not put to use in any context, even if it would be very suitable for use in a presentation tool like that of Baudel & Beaudouin.

Common for the works mentioned in this chapter is that they all focus on how well the technology works and how well users were able to use the developed systems. Sukthankar et al. concluded that their system allowed users to "*interact with the computer as if it were a member of the audience*" (Sukthankar, Stockton, & Mullin, 2001). Elmezain et al. made an application with high recognition rates for gestures and in the same way Malik concluded how his hand tracker was able to track and find fingertips through the use of a webcam. Schlömer et al. and Baudel & Beaudouin-Lafon did include users in their tests and based their conclusions on the users' efforts, but the tests were only concerned with the people performing the gestures and not the ones watching. Therefore it becomes less interesting to investigate the use of a gesture based presentation tool from the point of view of the presenter. There has already been research showing that humans are indeed able to utilize natural gestures and perform meaningful actions with them. What is more interesting is to look at how the receivers of a gesture based presentation will react and to see if the use of gestures change anything in the audience's perception and understanding of the presented content.

1.3 Initial Problem Statement

Based on the previously done work in the area of gesture recognition and presentation tools, there seems to be a lack of systems that utilizes the natural gestures that a presenter uses doing a presentation, to interact with the presentation. Work has been done to apply gesture recognition in presentation tools, but they have required the use of intrusive hardware, such as data gloves. Other studies have investigated gesture recognition, but without applying it in a presentation context. None of the studies have looked into the consequences of using gesture based interaction with regards to the receiving audience. Therefore, this project will look into how gesture recognition, in a system that detects natural gestures of the presenter, can be utilized to create a presentation tool that will improve the presentation, as perceived by the audience, compared to a similar presentation done with traditional presentation tools, using i.e. keyboard and mouse. This leads to the following initial problem statement:

How can a presentation tool, detecting natural gestures of the presenter, improve the experience of the presentation, as perceived by the audience?

2. Pre-analysis

Based on the initial problem statement this chapter will focus on narrowing the problem down to be able to form a final problem statement. First of all it is necessary to narrow down what natural gestures are and then limit the work of this project to focus on identifying a few of these gestures, in order to keep the project achievable within the give time frame. Furthermore it is necessary to explore how the interaction with a traditional presentation tool works, in order to know what functionality needs to be mapped to gestures. This will be important, both to identify what gestures are most important to develop recognition for, but also to be able to do a valid comparison between the interaction types later on in the report.

The other major part of this pre-analysis will be to explore the hardware described in *1.1 Motivation*, in order to determine which of the systems are most suitable for the development of a gesture based presentation tool.

2.1 Gestures

One problem with gestures, especially with regards to human-computer interaction, is to classify the gestures and identify their purpose in a given context. Normally, gesture based interfaces have a list of gestures that the user can perform in order to trigger certain actions, but this requires the user to learn and memorize these gestures to be able to use the application. A gesture based interface would be more meaningful if the user could use his normal, natural gestures to perform the actions. (Wexelblat, 1995) gives an example of a potential house buyer touring a virtual building. If the system made for this tour has implemented a certain gesture for opening doors, but the user forgets this gesture and instead tries to open the door like he would open a door in a normal house, he would fail and be left with a door that refuses to open (Wexelblat, 1995, p. 181). In an ideal interaction method, the computer system would recognize and interpret the gestures made by a user, the same way a human would. What Wexelblat did was to make a gesture analyzer that would segment data and extract features that matched natural gestures. He was not able to produce a system that was as adaptable and predictive as a human's perception, but was at least able to supply an approach to such a system. Others have also examined the area of natural gesture tracking in HCI, such as (Pottar, Sethi, Ozyildiz, & Sharma, 1998), (Eisenstein & Davis, 2004) and (Perzanowski, Schultz, Adams, Marsh, & Bugajska, 2001), but the work is most often limited to observing natural gestures within a group of users or in general and successively integrating the recognition of these gestures – or a selection of them – in order to achieve the best possible interaction between computer and system. Furthermore, the systems can be designed to learn new gestures, such that users can teach the system to recognize gestures that he feels is meaningful with regards to a certain action.

In a presentation context, natural gestures would refer to the movements done by the presenter while explaining the content of his slides. The problem is that these gestures varies a lot from presenter to presenter and one gesture that would seem logically matched to a certain action by one person will not seem logic to another. In order not to cognitively overload the users by making them learn and remember a list of new gestures, the system should ideally adapt to the gestures that a given user finds logical for performing certain actions. However, some gestures are more universal than other and might be meaningful to a greater group of people than others. *Deictic* gestures are gestures that refer to something in the physical space in front of the presenter doing the gesture, for instance pointing at a certain object or using ones hands to represent entities or events in space (Cassell, 1998). Especially the pointing gesture is common for highlighting objects of importance and has been used in other gesture recognition systems, such as the one by Sukthankar et al. or in (Lenman, Bretzner, & Thuresson, 2002). Thus it would be natural to include tracking of a pointing gesture, in order for the presenter to point at and highlight objects in his slides.

Apart from the pointing gesture to highlight important parts of the slides, the probably most used actions in a slideshow is the action of moving back and forth between slides. Possibly, this will be the only action performed by some presenters, in cases where the slideshows are not animated in other ways. For this action it is hard to define a universal deictic gesture to symbolize switching to the next slide. There might however be *metaphoric* representations that would map a certain gesture to the action of switching slides. A metaphoric gesture is a representational gesture, but they represent something that has no physical form (Cassell, 1998). For instance, the metaphor of changing the transparencies on an overhead projector or the metaphor of changing pages in a book could be used to represent switching slides. If a user would change a transparency on a projector, he would most likely place his fingers by the side of the transparency and drag it off the projector, before placing the new slide on the projector. However, sliding the current slide “off” the screen in a slide show presentation might be confusing as to what slide will appear: The next or the previous of the slide show order? Using the metaphor of switching pages in a book will give a more clear impression of what will follow, depending on if you flip the page left or right. The problem with such a gesture is that it takes up a lot of physical space in front of the presenter, if he is to mimic changing the page of a book. Changing pages in a book might also be done with a “swipe” like motion though: Placing your finger on the right page of an open book and dragging it to the left will also change the page. It might not be very good for the book, but you do not have to worry about that in a slide show and the motion is both simple and meaningful. Furthermore, the swipe is well known by users of the iPhone (or other similar phones with touch screens) to change “pages” in a menu or a photo album. Swiping from right to left will change page forward and swiping from left to right will change backward.

According to Cassell, there are also iconic and beat gestures, which are used often in presentations. Iconic gestures are representational like the metaphoric ones, but represents real actions or events, such as demonstrating how a certain product is used, without actually having the product in your hands. Beat gestures are small motions with your hand that does not represent anything, but are movements you do while presenting to support or comment your speech, like little flicks or similar movements (Cassell, 1998). These two forms of gestures are not as useful in a gesture recognition system for presentations. The iconic gestures represent physical actions or events and do not map well to the virtual slide shows being projected and the beat gestures does not carry enough meaningful information with regards to the actions the user wants to perform. *Table 2.1* gives a brief description of the four types of gestures.

Gesture type	**Deictic**	**Metaphoric**	**Iconic**	**Beat**
Description	Relates to something directly in the physical space of the presenter (i.e. pointing at an object.	Representational, describing something of no physical form.	Representational, describing an action of event of physical form.	Small, often repeating motions, used to support or comment one’s speech.

Table 2.1: A brief description of the four types of gestures defined by (Cassell, 1998). Deictic and metaphoric gestures are the most suitable for a gesture based presentation tool.

In order to get information about whether or not the swiping gesture actually seems natural to the users who would potentially use the product or watch presentations done with this product and to find out if other actions are crucial for the application to be successful, it was necessary to gather data from possible users. In order to get these users’ opinions a small survey was conducted.

2.1.1 Online survey about use of digital presentation tools

The survey should first of all give some demographic information about the persons filling out the survey, in order to determine if it is expert, casual or novice users of presentation tools. There might be a big difference in what a novice and an expert user would require of a gesture based presentation tool. To clarify the respondents' experience, they were asked how often they use presentation tools, for what purpose they use them and also what tool they use. This should give an indication both about their experience with traditional digital presentation tools as well as give information about if certain actions are preferred for certain types of presentations. Another important piece of information is regarding what functionalities the persons use in their information. It is assumed that changing slides will be a functionality used by most, but in order to get an idea of what other functions and actions should be considered when designing a gesture based application, respondents were given a list of common functions from slideshow presentations to choose from.

Having determined the demography of the respondents and their usage of presentation tools, the other part of the survey is regarding gesture based interaction in presentation tools. In order to get an idea of what the users of presentation tools think is the most natural gestures for different actions, the survey contained questions about how the respondents would change slides back and forth as well as highlighting objects, using gestures. Furthermore, respondents were encouraged to give their opinion on other actions they thought would be natural to control through gestures. These questions supply information about if the metaphoric gesture of swiping seems natural to the respondents too and if there are any other gestures that are crucial to design for in the application. The final survey can be seen in *Appendix 13.1 Survey about use of digital presentation tools*.

The survey was distributed online and a total of 16 people answered the survey, out of which most persons (five) stated that they used presentation tools every 2-4 months. Four persons used them monthly and three used them weekly. This means that most of the respondents use presentation tools on a fairly regular basis and are assumed to be familiar with presentation software and the functionalities of it. Microsoft PowerPoint was the most widely used presentation tool, with a total of 13 people stating to be using this software. Two of these persons were also using Apple Keynote, while one person noted to be using PowerPoint and OpenOffice Impress. None of the respondents used the online presentation tools Prezi or Zoho. With regards to the purpose of the presentations, the answers were very widespread and included all the given options from the survey. There did not seem to be any correlation about the purpose of the presentation and how often the respondents used the presentation tools. However, almost all respondents answered that they were using keyboard and/or mouse for controlling the slideshow. 12 of the 16 respondents claimed to be using keyboard/mouse and three claimed to be using an automated slideshow. One person did not answer this question as he/she has stated to never be using presentation tools. With regards to what functions the respondents used in their presentations almost everyone stated to be using the functionality of changing back and forth between slides. 14 of 16 respondents answered this question and 13 of them stated that they used the functionality of changing slides back and forth. Five people stated to also be using the function of jumping back and forth between different slides in the presentation. The rest of the options were chosen by less than five persons each and was mainly concerned with selecting objects/graphs or text.

In the second part of the survey, which was about the gestures, the respondents seemed to agree on the metaphoric swipe gesture for changing slides. 12 of the 15 users who answered this question would want to change slides back and forth by swiping their hand in front of the slideshow. Even if this question was answered purely by text input, the respondents seemed to describe the swipe very similarly and several people referred to the swipe function known from the iPhone. One person even mentioned the book metaphor. There was however variations as to which direction the swipe should be performed, in order to change slides backwards and forwards, and there were no clear conclusion as to which direction was preferred by the respondents. Out of the eight people who indicated a direction of the swipe, there were equally many preferring a left-to-right swipe as a right-to-left swipe, for changing forward in slides.

With regards to highlighting objects in the slideshow, the answers were a bit more varied, although leaning towards the same solution. The majority of the respondents wanted to point at the object or text they wanted to highlight and then "clicking" in the air to complete the highlighting. Other approaches were also suggested, such as drawing a circle around the object to be highlighted or moving a finger back and forth several times under the object or text, as if to underline it. The users furthermore indicated that a highlighting should result in some sort of visual feedback, such as the highlighted object becoming bigger or "moving closer".

As for other suggestions for actions that could be controlled through gestures, the feedback was limited. Two respondents noted that the objects in a slideshow presentation are often appearing one after the other and not all at once. Thus it should be possible to have a gesture for making objects appear and one of the two respondents suggests using a vertical slide for this purpose. Other suggested functionalities include activation of hyperlinks and zooming.

To conclude on the survey it shows that people seem to be using the same functionality in presentation tools, independent of their experience and the purpose of the presentations. The change of slides is the most crucial functionality to implement and people seem to consider the metaphoric swipe gesture a natural choice for this function. Also highlighting of objects seems important to the users of presentation tools and they seem to consider the deictic pointing gesture a natural choice for this, combined with an "air click". Thus, these two gestures should be the most important ones to implement and possibly add functionality for free hand drawing or zooming.

Based on the information gathered in this chapter, the most important thing to note is that the most general actions in a slide show presentation should be covered by gestures for the application to be successful. These action should be mapped to natural gestures that are meaningful to as large a group of people as possible. Based on the survey conducted it was clear that at least the swiping and pointing gestures are natural gestures to the users. However, there were no clear indication towards the users needing more functionality from a gesture based presentation tool and thus the change of slides and highlighting should be the main focus for the application. Having covered what kind of gestures to focus on during development, the following chapter will investigate how to track and recognize these gestures from a technical aspect.

2.2 Human tracking hardware

As mentioned in *1.1 Motivation* there is several pieces of game hardware that can track its users in different ways. One of the earlier attempts was the Sony EyeToy, which was released in 2003 and was basically a webcam connected to a Sony Playstation 2 and allowed players to play without controllers. The Playstation 2 can record video at up to 60 frames/sec (fps) and a resolution of 320x240 (Scantleberry, 2003). Most midrange webcams today can match this and for the comparison of technologies in this chapter, the EyeToy technology will be regarded as using any modern webcam for doing the tracking.

A more recent approach to doing gesture recognition in commercial terms is the Microsoft Kinect that was released in the end of 2010. Kinect can stream color video inputs at 30 fps at 640x480, but compared to regular webcams it has the property of also being able to simultaneously stream a 320x240 16-bit depth image at 30 fps (Pankinkis, 2010). The depth image can be useful in doing tracking, because it allows for segmenting not only based on the information in the color image, but also based on the distance between the camera and the recorded objects. If developing games to be played on the Kinect's original platform – the Xbox – you need a development kit from Microsoft. However, for development on the PC there are open source code libraries and drivers available for using the Kinect. The most popular of these is the project called OpenKinect, which supplies both drivers and source code for using the Kinect on several platforms, including Windows, Linux and Mac (OpenKinect, 2011).

If looking outside the realm of directly tracking the user's hands, there are the Nintendo Wii and Playstation Move. Both of these uses wireless controllers, and are doing tracking in similar ways, although the Move is a bit more advanced. Compared to the Wii, the Move is able to also give depth information by tracking the relative size of a colored light bulb at the top of the controller, meaning that it is actually a mix of camera tracking and accelerometer tracking. The camera tracking is done through the Playstation Eye camera, which is a newer version of the EyeToy. The problem with the Move is that it is a more difficult technology to access as a developer. There are no open source libraries or drivers released and although some might be on their way, Sony is planning on keeping the Move a gaming dedicated controller (Kuchera, 2011). This makes it unsuitable for PC development, even for a prototype, since it will be too cumbersome to get the system working.

For the Wii there are different options available when it comes to open source libraries and one of the most used libraries for interfacing with the Wii controller is the OpenFrameworks (OpenFrameworks, 2011), which is also what the OpenKinect library is built upon. This makes it just as easy to start developing for as the Kinect. One of the pros of the Wii is that it directly supplies data about the user's hand movement, through its accelerometers. This means that I would be able to directly use the input from the Wii remote and start analyzing this data to identify gestures, instead of having to do initial image processing as it is the case on the Kinect.

To give a summary of the different technologies *Table 2.2* will give an overview of the different technologies listed in this chapter and their pros and cons with regards to their use in this project.

Technology/Platform	Pros	Cons
EyeToy (webcams)	Cheap and easy to develop for with open source image processing libraries.	The quality varies a lot in resolution and frame rate and they are often very dependent on light. Image processing needed before gesture recognition can be done.
Kinect	Specific open source libraries and drivers available. Supplies both regular color image and a depth image.	Not as cheap as regular webcams. Somewhat light dependent. Image processing needed.
Move	Both accelerometer and camera tracking.	No open source libraries available. Is dependent on a controller.
Wii	Open source libraries are available and the accelerometer based tracking, makes image processing unnecessary.	Controller dependent.

Table 2.2: Comparison of the different technologies presented in this chapter. Kinect and EyeToy are the only controller independent systems, but it is thus also more difficult to get access to data about the user.

Based on the information covered in this chapter about the technologies, it shows that Wii or the Kinect would be most suitable for the purpose of this project, primarily due to their accessibility. The Wii gives a more direct approach to the gesture recognition, since it is possible to use the data supplied by the controller directly for the recognition. However, using a Wii controller for presentation is just as intrusive as having to use a remote control or laser pointer. Furthermore, it is not possible to track the fingers of the user with the Wii remote, which is possible with the Kinect by doing image processing. The Kinect is a bit more expensive than a Wii controller, which might be an issue for some, if it was to develop into a commercial product. The

Kinect can also have problems in environments with lots of sun light, but in a presentation context that would rarely be a problem, since direct sunlight would probably also ruin the audience view of the slide shows. Thus, the Kinect will be the preferred technology for development in this project, due to it being a less intrusive technology than controller based systems and because of the open source libraries available for developing on PC.

2.3 Delimitation

Based on the chapters of the pre-analysis, the scope of the project will be limited, before forming a final problem statement. First of all, the application to be developed will focus on tracking only a few natural gestures, including a pointing gesture for highlighting and a gesture for changing slides. The application will furthermore be focused on metaphorical gestures that will be meaningful to a Western cultural group of people. The most common actions performed in a traditional slide show presentation should have gestural commands attached to them, in order for the users to be able to do a full presentation using gestures only.

With regards to the technology, the Kinect will be chosen as the preferred type of hardware, due to it being non-intrusive to the presenter and providing a good opportunity for tracking, using open source code libraries such as OpenKinect and OpenCV.

2.4 Final Problem Statement

How can I, using the Microsoft Kinect, create an application that will improve a slide show presentation towards the perceiving audience, using only a few, predefined, natural gestures?

3. Methodology

Having established the final problem statement of the project it is necessary to further analyze key aspects of the problem statement, in order to obtain knowledge that should be used in designing and implementing the product. However, before going into the analysis this chapter explains the methodology and structure of the following chapters of this report as well as explaining the work structure that was applied throughout the development of the product.

When developing user oriented applications, tests are an important part of evaluating the success of the application. Any tests of prototypes will bring new knowledge that will alter the requirements for the final product. This will lead to new topics having to be analyzed, which will again lead to a re-design and re-implementation of the product, according to the result of the previous test. *Illustration 3.1* shows a graphical illustration of how this process works.

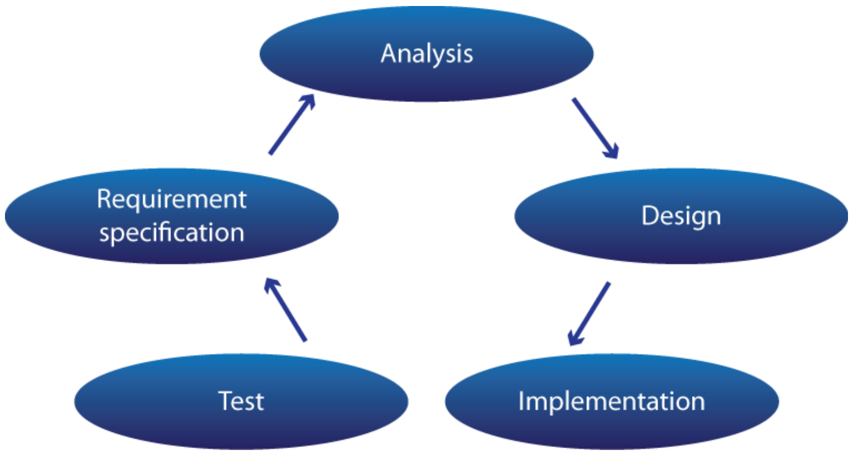


Illustration 3.1: The iterative design makes sure that product development adheres to results of user tests of previous prototypes.

At some point, development will have to exit out of the loop, due to the project deadline, but throughout the product development the work process will follow the flow of the model in *Illustration 3.1*. In order to keep the report structured and ordered it will only cover the most crucial re-iterations. Throughout the entire work process the iterative design was applied and even if most of the tests were simply debugging sessions, they were crucial in forms of re-specifying the requirements for further development. Problems of this nature will be explained throughout the chapters of the report as well as their solutions. However, there was one important change that happened throughout the project period, which changed the course of development noticeably. For the first part of the project the idea was to do skin color segmentation, extract hand features and perform gesture recognition based on this information, with the help of the Kinect's depth sensor, which will all be explained in the chapters to come. Unfortunately, this tracking method proved to be both too slow and too unstable to be able to get a reliable test setup within the time frame of the project. At the semester midterm presentation, I was made aware of another code library that would give direct access to the skeleton tracking that is also being used on the Microsoft Xbox for their Kinect games. With this library it was easy to get information about the presenter's position in space, although finger tracking was not possible. In order to be able to do a valid test, the final testable prototype was developed using this code library. To illustrate this work process, the report will be divided into a "Part I", which explains the first approach with skin segmentation, and a "Part II", which will explain the approaching using the skeleton tracking.

Part I

Part I will include an analysis of the theory and technology needed to perform the skin segmentation, feature extraction and gesture recognition. The design and implementation of this tracking method will also be covered here, in order to explain both the pros and cons of choosing this approach. Furthermore, Part I will also cover the analysis and design of the parts of the project that is independent of the tracking method, such as how to test the product and how the setup of the application will be.

4. Analysis I

This analysis will be primarily concerned with further investigating the most important parts of the final problem statement. First of all it is a matter of finding the best way of tracking the user's hand and extract information about gestures from this. Due to the choice of using the Kinect as the desired development platform, it is necessary to do image processing to find the user's hand or hands and then run the gesture recognition algorithm on the data extracted from this image processing. The image processing should take care of figuring out if a hand is present in a given image, segment the hand and give information about the state of the hand and its position in the image. The gesture recognition algorithm will then take care of analyzing this data in order to see if any valid gestures have been made.

4.1 Programming language

Since it is already known that some image processing is needed for the Kinect to be able to recognize the presenter's hand, it is necessary to choose a programming language and a programming library that supports image processing. As described in *1.1 Motivation* an example of an image processing library is OpenCV, which is developed for C++, C and Python. It does exist for other coding platforms too, such as ActionScript or C#, but they are modification of the official library and do not contain the full functionality of the original OpenCV library. The OpenKinect hardware, also mentioned in *1.1 Motivation* is the other part of getting the application working with the Kinect. This library is developed for a lot of different platforms and languages, including both C++, C#, Python and Actionscript. This opens up for development on several platforms and with several programming languages, since most of the common programming languages are supported both by OpenKinect and OpenCV. The C-style languages as well as Python has the advantage of being officially supported by both libraries and that they are all relative low level languages, which means a great deal of freedom during development. Furthermore, C++ is the language I am personally most familiar with and have the greatest experience with, which means that developing in this language will also be easier for me than other higher level languages, such as ActionScript.

When it comes to the production of the visual part of the application - the actual slideshows - there are other development environments that supply a more direct approach to setting up graphics. Adobe Flash is one example of a developing environment which gives a direct approach to creating and animating graphics and would supply the tools needed to create a small slideshow application, which could then be controlled through the image processing of OpenCV in a C++ application. Another approach is to use a game engine like the freely available Danish engine called Unity 3D, or simply Unity (Unity3D, 2011). Even if it is a game engine for developing 3D games it is very flexible and can be used to easily create smaller applications too. Unity supports JavaScript, C# and Boo for programming and from previous university projects I have experience working with this engine and C# programming. The problem with using either Flash or Unity is that data from the image processing application should be transferred to the graphics application, which might be a cause of latency. The transfer can happen either through input/output (IO) streaming through a local file or through a socket connection internally on the host PC. With the IO approach the image processing application will write to a file locally on the PC, while the graphics application will then read the data from this file at runtime. Using a socket connection data would be transferred directly over the chosen port number, which will improve the speed of the application, but is also more cumbersome to set up. However, compared to the time it will take to implement the visuals through C++, it is more prudent to develop the visual part of the application in a development environment that is suited for this purpose and then deal with the latency by making sure to send as little data as possible and as rare as possible.

Having decided upon development language and environment, the following chapter will look into how the image processing can be done, using the OpenCV library.

4.2 Image processing

The Kinect is in many ways functioning as a traditional web cam, except for one important difference. The Kinect also features depth tracking by using an infrared camera. This means that it is possible to not only get information about distances between objects in a room, but for the purpose of tracking specific object of known distance to the sensor, it is possible to discard information in the image about objects that are too close or too far away, hereby reducing the amount of analysis that needs to be done. However, even when limiting the amount of data by the depth, it is still necessary to do some image processing to be sure only the desired objects are tracked. In the case of this project, the desire objects are the users' hands, which mean that some way of finding hands through image processing is needed. In (Malik, 2003), an approach to finding a hand in a colored image is proposed, based on method described by (Jones & Rehg, 1999). The method is based on first doing a simple background subtraction and then a *skin pixel detector* as it is referred to by Malik. The background subtraction will compare images coming in from the video feed from the webcam (the Kinect in the case of this project) with an image that is classified as being background. Any pixel value in the foreground image that differs by some threshold in color value will be classified as foreground. The foreground pixels are then the ones being put through the skin pixel detector in order to filter away anything that is non-skin.

The skin pixel detector checks the probability that any given color is skin or not skin. It does so by being trained on a series of training images, in which skin color is manually being selected. A training program will run through all the pixels in the training image and save the distribution of color values in a histogram. In (Malik, 2003) it is suggested to arrange the pixels in 32 bins, meaning that each bin will cover eight values in the range of 0-255 for each color channel. The same procedure will be performed on a series of images containing non-skin pixels. This will give two histograms H_s and H_n for skin and non-skin respectively. With these histograms it is possible to calculate the probability that any given RGB color is skin or non-skin using the following equations:

$$P(rgb|skin) = \frac{s[rgb]}{T_s} \tag{4.1}$$

$$P(rgb|\neg skin) = \frac{n[rgb]}{T_n} \tag{4.2}$$

where $s[rgb]$ is the pixel count in the bin corresponding to the given RGB value in histogram H_s and $n[rgb]$ is the corresponding value for H_n. T_s and T_n are the total counts of the skin and non-skin histograms.

Using Bayes rule it is possible to calculate the probability of a given pixel of a certain RGB value being skin or non-skin. The probability of any given RGB pixel being skin is thus given by:

$$P(skin|rgb) = \frac{P(rgb|skin)P(skin)}{P(rgb|skin)P(skin) + P(rgb|\neg skin)P(\neg skin)} \tag{4.3}$$

$P(rgb|skin)$ and $P(rgb\,\middle|\,\neg skin)$ is already given by equation 3.1 and 3.2. The prior probabilities $P(skin)$ and $P(\neg skin)$ can be calculated by using the total count of the skin and non-skin histograms (Malik, 2003).

Since a pixel in this case can only be skin or non-skin, it is given that $P(skin) + P(\neg skin) = 1$ and thus the two probabilities can be calculated by:

$$P(skin) = \frac{T_s}{T_s + T_n} \tag{4.4}$$

$$P(\neg skin) = 1 - P(skin) \tag{4.5}$$

Using the value of $P(skin|rgb)$ at runtime and only keeping pixels with a high skin probability, it is possible to further segment the image to leave out pixels that are not skin colored (Malik, 2003).

4.2.1 Noise reduction

When working with image processing from webcam feeds and similar hardware, there will often be different kinds of noise to take care of. Noise is often a problem when doing i.e. edge detection, because the edge detection algorithm will find edges along all the noisy pixels, which is undesired. A common way of getting rid of noise in an image is by using a low pass filter, which is the process of applying a convolution kernel to an image, in order to blur the image, hereby eliminating pixels which are varying greatly in color value, compared to its neighbors. An example of a simple low pass filter is a *mean filter*, which as the name implies takes the mean of the neighboring pixels in order to calculate the new value of a given pixel in an image. A 3x3 kernel for mean filtering would thus look as follows:

$$\begin{bmatrix} 1/9 & 1/9 & 1/9 \\ 1/9 & 1/9 & 1/9 \\ 1/9 & 1/9 & 1/9 \end{bmatrix}$$

Applying this kernel on an image would give each pixel a new value corresponding to the means of its neighbors and itself. This means that if a pixel – in a grayscale image – has a high value compared to its neighbors, it will be reduced to a value closer to that of its neighbors. This smoothes the image, but also removes details.

Another way of smoothing is by using a non-uniform kernel, meaning that not all pixels in the kernel are weighted equally. A common example of such a kernel is seen in a Gaussian filter (Efford, 2000), in which the coefficients of the kernel is calculated using a Gaussian function, meaning that the outermost neighbors will have little weight in the smoothing, and the centre will have the most influence. This will also smooth the image, but compared to the mean filter, fewer details are lost in the process. Using OpenCV it is possible to apply both a mean filter and a Gaussian filter, by using the function `cvSmooth`.

Another approach to noise removal is that of erosion and dilation. In these operations you apply a structuring element to the image in order to either expand or shrink blobs of non-zero pixel values. When doing erosion the pixel is kept wherever the structuring element fits the image and in any other cases, the pixel will be set to 0. This will erode the image, making any areas of non-zero pixels smaller, which can be used in situations where some segmentation has already taken place, but the previous operations has left small areas of non-zero pixels in the image that are not desired. For instance, after doing the skin color detection described previously in this chapter, the foreground image might consist primarily of the hand that was desired to be found, but small blobs of pixels might still be present in the image because they were falsely identified as being skin. By applying erosion, these small blobs (noise) will be removed, but the main blob (the hand), which is a lot bigger, will be kept although reduced in size. To restore the main blob to its original size, dilation can be applied. Dilation looks for a hit, which means that only one value of the structuring element has to be similar to the image at the given pixel coordinate. This means that any blobs of non-zero values in the image will be expanded, hereby restoring the remaining blobs to their original size. The process of applying erosion followed by dilation is called opening and doing the two operations in the opposite order is called closing. The reason for this naming is clearer when looking at what the closing operation does. By first dilating and then eroding, all blobs will first be expanded and then decreased in size. If a blob has holes in it of zero-values, these holes will be covered when doing the dilation (given that the holes are small enough and the structuring element is big enough) and the successive erosion will then restore the blob to its original size, although without the holes now. Both erosion and dilation can be directly implemented using the OpenCV functions `cvErode` and `cvDilate` respectively.

4.3 Feature extraction

When having achieved a reliable tracking of skin color in the input images, it is necessary to implement an algorithm for identifying whether the skin colored blobs are a hand or not. In (Malik, 2003) an assumption is made that the tracked blob of skin color is in fact a hand and thus feature extraction is done directly to identify fingers. In an application used for a presentation tool, it is not certain that only the user's hand will be in the image. When the user walks in front of the slide show presentation his head will also be visible for the camera and thus also be tracked as skin colored. The approach by Malik can be part of the process of identifying a hand in an image though, since it looks for features that should ideally be unique to the fingers of a hand. OpenCV provides a way of finding the contours of blobs in a binary image and given that the skin segmentation has been successful, the blobs in the skin segmented image should only consist of a head, hands and possibly arms. Ideally, that should provide an image similar to the one seen in *Illustration 4.1*.

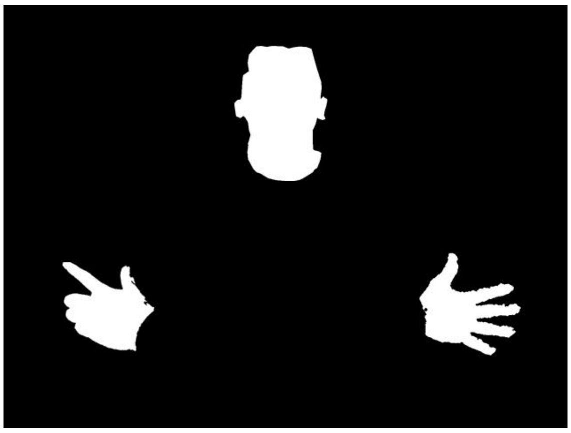


Illustration 4.1: The blobs seen in an incoming image after skin segmentation has been completed. The face and hands are the only visible parts left. This is an ideal situation and the image is not a result of the final application.

By looking at the contours of the segmented blobs, it is possible to look for certain features in the blobs that could help identify them as hands or heads. Looking at *Illustration 4.1* it can be seen how the fingertips are noticeable features in the image. This is the observation Malik made too and his method is based on tracing the contours in order to identify narrow angles, which will then be identified as fingertips. OpenCV supplies the opportunity of supplying coordinates for points around the contour and thus it is possible to look at these points, in order to calculate angles around the contour, as seen in *Illustration 4.2*.

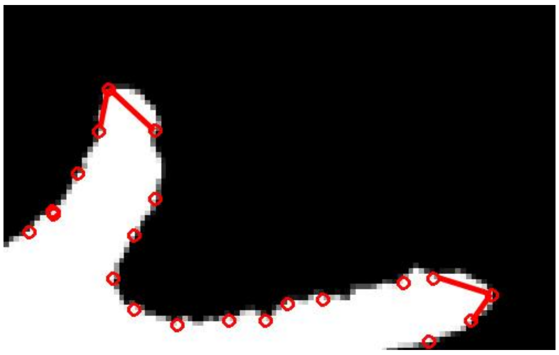


Illustration 4.2: By calculating the angles between the points along the contour, it is possible to find the narrowest angles and identify these as fingertips. In the picture above, the two narrowest angles are highlighted with lines to illustrate the angle. The image is an actual result from the application implemented.

The angles are found by looping through all the points and for each point making a vector between the point before and the point after. Let C be a contour in the image and i be a point on the contour. Then the angle θ can be calculated as:

$$\theta = cos^{-1}\left(\frac{[C(i) - C(i-1)] \cdot [C(i) - C(i+1)]}{|[C(i) - C(i-1)]||[C(i) - C(i+1)]|}\right) \tag{4.6}$$

If the angle is then below some threshold, the point of the angle will be considered a fingertip. Malik suggest a threshold for the angle of 30. However, this method will also find dips between the fingers, if the fingers are close enough. Malik solves this by placing the vectors in 3D space, lying in the xy-plane, and then compute the dot product. The sign of the z-component will then decide whether the point is a tip or a dip: Positive means a tip, negative a dip (Malik, 2003).

The problem with this approach is that there is a high potential for false positives, meaning that the algorithm will find finger tips where there are not supposed to be any, because the angle is below the given threshold. This can happen for example if a contour gets jagged along the edge due to a poor tracking, resulting in a lot of narrow angles or if the tracked hands are far away, resulting in a small contour where the number of points along the edge for doing the angular test is limited. The jagged edges can be removed by smoothing the image or by ignoring that certain part of the image, if a lot of narrow angles is found successively. If the tracked contour is too small to do a reliable finger tip tracking it can be expanded, which will result in more points along the contour, but which will also expand any noise or unwanted pixel data.

4.3.1 Face detection

Using the approach of finding finger tips may not be completely reliable, since there is a risk that narrow angles, which identifies a fingertip, might also occur in the blob that makes up the user's face. In order to ignore this blob before doing the fingertip feature extraction, a face detection algorithm can be used to detect the presenter's face and ignore that part of the image. OpenCV supplies a method for doing face detection using Haar-like features, which is an approach described in the work by (Viola & Jones, 2001). The approach is based on the relation of average pixel intensities between rectangular areas in the images. For instance, when tracking a face it is known that the area around the eyes is generally darker than the surrounding areas of the face. The problem with detection using this method is that it requires a lot of training data in order to obtain a so-called cascade of classifiers that can be used for a reliable detection of the desired object. For instance in the work of Viola & Jones they used 4916 pictures of faces (which were all mirrored vertically to produce a total of 9832 training images) and 9544 images of non-faces for training their face detector (Viola & Jones, 2001). Using OpenCV this training is unnecessary though, since cascades for different tracking purposes, such as frontal face and profile face detection, are already included in the OpenCV library. Hereby this approach becomes useful for doing face detection in the presentation tool, since it can be implemented quickly and that the algorithm itself is also fast and computationally light (Viola & Jones, 2001). In this chapter the theory behind the Haar-like feature approach will not be described, but instead an explanation of how to utilize it will be covered.

OpenCV supplies different classifier cascades in xml-format and these can be loaded into any given application. When using the OpenCV functionality for doing Haar-like feature tracking the function `cvHaarDetectObjects` will return an array of bounding boxes for the detected objects. Loading cascades for both frontal and profile faces it is thus possible to get an array of bounding boxes which surrounds the skin colored pixels of the face, as seen in *Illustration 4.3*.

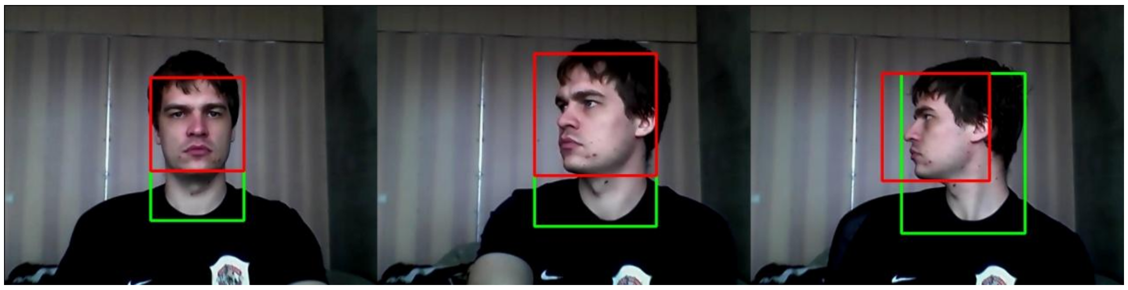


Illustration 4.3: Face detection using OpenCV's Haar cascades. The images show tracking with a combination of frontal and profile cascades. The red rectangles are the original bounding boxes produced by OpenCV and the green ones are scaled versions of these bounding boxes, made to fit the entire skin region.

By knowing the position of the face it is possible to ignore this area of the image when doing fingertip feature extraction. This can be done either by removing blobs that are intersecting with the area that are classified as a face region or by expanding and offsetting the OpenCV bounding boxes to fit over the entire skin region (as shown by the green boxes in *Illustration 4.3*) and then setting all pixel values to 0 within that area. The downside of this approach is that if the presenter moves his hand in front of his face while presenting, the hand will not be tracked either. However, this will not happen very often, since the presenter will rarely want to keep his hand in front of his face, but rather in front of his body or in front of the projected slideshow.

Theoretically the Haar-like feature approach could also be used to track hands, but it would require a great amount of training data for each pose the hand could be expected to take, which is not a prudent solution. However, for face detection Haar-like features are useful and combined with the skin segmentation and feature extraction described in *4.2 Image processing* and *4.3 Feature extraction* it will aid in making the detection and tracking of the hands more reliable.

4.4 Gesture recognition

When having successfully identified a hand in the incoming image, the application should use this information to be able to identify gestures. Depending on the complexity of the gesture, there are different ways of approaching the gesture recognition. For the deictic gesture of pointing, it is not necessary to do any gesture recognition, although pose recognition might be needed. In order for the software to determine when the user is pointing and not just moving his hand around with no intention of highlighting anything, some form of activation of pointing is needed. This activation can happen both through a gesture and a pose, but since the gesture involves movement, the activation might lead to unwanted pointing at objects in the slideshow. A pose is static and thus the users hand will not move until he wants to start the actual pointing. The problem with a pose is that in order to identify what pose the user is holding his hand in, a clear identification of the fingers are needed. As described in *4.3 Feature extraction* there can be problems with false positives in the finding of fingertips, despite doing different forms of error handling. Another approach of activating the pointing gesture is to have a certain area within the slideshow that will activate pointing if the user holds his hand within this area for a short amount of time. This will mean that it is not necessary to do finger tracking constantly, but only once pointing is activated. It is slightly more cumbersome for the presenter, but will improve the reliability of the application.

With regards to actual gestures that should be mapped directly to actions in the slideshow, it is necessary to track the movements of the presenter's hand. For the user to do a swipe gesture his hand will only move in one direction, which means the movement can be translated into a 2D vector. Given the magnitude and direction of this vector, it is possible to determine if the user's hand has moved in a way that fulfills the pattern of a swipe either left or right. The problem with this method is that the user might be rotated in space, according to the viewpoint of the Kinect. Had the user been operating a touch screen or similar 2D surface there would have been no depth factor to consider, meaning that a swipe could not be mistaken for another gesture: A left-to-right or right-to-left movement of the finger or hand would always happen along the x-axis. In front of the Kinect the user can move along three axes and a swipe would often be moving in the xz-plane, rather than just along the x-axis, as shown in *Illustration 4.4*.

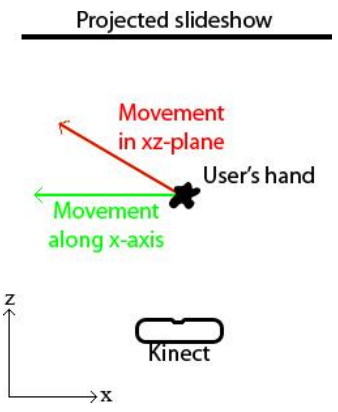


Illustration 4.4: When the user moves his hand around in front of the Kinect, his hand will often move in the xz-plane, which might be problematic for tracking a swipe gesture.

This could potentially lead to the swipe being recognized as something else, like for instance a flick of the hand towards or away from the slideshow (moving the hand rapidly along the z-axis), if such a gesture should be implemented too. Doing a swipe gesture towards the Kinect, given that the Kinect is facing the slideshow, will not be a natural mapping for changing slides though. Since the metaphor relates to flipping through the pages of a book, it must be assumed that users will move their hands with relation to the slideshow and a swipe will move along the axis that the slideshow is projected to. Looking at *Illustration 4.4* it is noticeable how even a movement in the xz-plane will have a higher magnitude in the x-direction than in the z-direction. Thus, a swipe gesture can be assumed to be a movement along the x-axis of a velocity that exceeds a certain threshold. A slow movement should not be confused with a swipe, but the threshold value of when the movement is fast enough is something that will have to be found through testing.

For more advanced gestures a more advanced system would have to be used. *Hidden Markov Models* (HMM) has traditionally been widely used in gesture recognition and examples of such can be seen in

(Freeman & Roth, 1994), (Wilson & Bobick, 1999) and (Schlömer, Poppinga, Henze, & Boll, 2008). However, for the purpose of recognizing the gestures that were indicated necessary in the survey conducted in *2.1.1 Online survey about use of digital presentation tools* such algorithms will not be necessary and will thus not be covered in this report.

4.5 Testing

With a well functioning algorithm implemented for detecting hands and recognizing gestures, it is necessary to set up an experiment for testing the application. The purpose of this project is to see if a gesture based presentation tool will work better for the audience than a traditional slide show presentation, controlled through keyboard and mouse. Thus, it is necessary to do an unbiased comparison between the two and in order to do so, there are several things to consider, which will be covered in this chapter.

First of all, it is necessary to set up a hypothesis about the experiment. Based on the final problem statement of this report, the null hypothesis of this project will be that no difference is found between a gesture based and a mouse/keyboard based presentation tool. Since the prior research in this area is limited, there are no strong indications towards one type of interaction method being superior to the other and thus the experiment will be based on a two-tailed hypothesis (Sharp, Rogers, & Preece, 2007, p. 661). The goal of the experiment is to prove or disprove the null hypothesis, by acquiring data from test subjects about their experience of the presentations. In order to acquire this data a suitable test setup must be chosen, which first of all includes how to group test persons and what test persons should be presented to what. In general there are three different approaches to assemble the group of test participants: *Different participants*, *same participants* and *matched participants* (Sharp, Rogers, & Preece, 2007, p. 663). With a 'different participant' approach, different test subjects are exposed to the different conditions of the experiment, such that no test subject is exposed to more than one experimental condition. This means that there is no risk of the results being influenced by training/learning effects, where test subjects would perform better or change their opinion between two experimental setups, because of knowledge obtained from their first experience. The disadvantage with this method is that a larger group of test participants are needed in order to get reliable results, since each participant will only supply data for one condition. If instead using the 'same participants' experimental design, only half the amount of test subjects are needed, since each participant will be testing both conditions. The disadvantage of this approach is that training/learning effects might influence the results of the experiment and the participant's prior knowledge about the subject being tested might also influence the results. The latter can be reduced by using a 'matched pair' experimental design, which is similar to the 'different participants' design, except that test subjects are grouped based on characteristics that might influence the results. For instance, participants could be grouped by age, gender, education or similar and then randomly being assigned to each of the two conditions. The problem with this approach, as with the 'different participants' design, is that it requires more test participants. Furthermore, in order for this method to have any advantage compared to 'different participants' design, it is necessary to be very certain about the influences that the ordering characteristics are having on the experiment. Thus, if a experimental setup can be designed, which minimizes the effects of training or learning between the two conditions, a 'same participants' approach will be a good approach for testing, since it requires less time for finding and testing participants. Furthermore, by making sure to switch evenly between what conditions the participants are first exposed to, effects of training will further be reduced.

When having acquired the data needed in the experiment, calculations need to be done in order to determine if the result of the experiment is reliable enough to reject the null hypothesis. According to Sharp et al., the most common way of doing statistical tests is through the use of a t-test (Sharp, Rogers, & Preece, 2007, p.

664). The t-test is a way of checking whether there is a scientifically significant difference between two results and is calculated as follows:

$$t = \frac{|\overline{X_1} - \overline{X_2}|}{\sqrt{\frac{{\sigma_1}^2}{n_1} + \frac{{\sigma_2}^2}{n_2}}} \tag{4.7}$$

Where $\bar{X}$ is the sample means, σ is the standard deviation of the two samples and n is the amount of participants in each of the samples. By calculating the t-value and the degrees of freedom, which is found by $df = (n_1 - 1) + (n_2 - 1)$, it is possible to look up the p-value for the test in a table of significance. The p-value is what indicates whether the result is significance and normally a value lower than 0.05 is considered as significant (Sharp, Rogers, & Preece, 2007, p. 664). This means that the chance that the result of the test happened by chance is less than 5%.

The t-test can only be done on quantitative data, such as the comparison of a scaled response in a questionnaire. However, qualitative data might also give important inputs about which method is the preferred one. Qualitative data can be obtained in several ways, for instance through simply observing the participants while performing the experiment. An observation can give information about what parts of the application participants have most trouble understanding or interpreting. Another way of acquiring qualitative data is through interviews and/or questionnaires, in which the test subjects are asked about their experience with the two presentation types and can thus express this in their own words, rather than on a predefined scale. Thus it will be prudent to design an experiment which mixes qualitative and quantitative data, in order to obtain as accurate an image of the test as possible. Quantitative data will aid in more easily comparing the different setups, while qualitative data will give a more in-depth look into the test subjects likes and concerns.

Based on the information covered in this chapter, the application to be designed and implemented should be made in such a way that both interaction methods can be compared equally and unbiased. The learning outcome of experiencing one presentation type before the other should be minimal, such that each participant can be used to test both conditions. The test should focus on both quantitative and qualitative data in order to obtain information that will give as detailed a result as possible. Since the test will mostly be about personal opinion and preference for the test subjects, it is important that the collected data will reflect these opinions.

4.6 Specification of requirements

To sum up the analysis chapter, this subchapter will cover the requirements that are needed in order to design the application in the coming chapter. Some of the requirements for the application are given by the final problem statement. This includes that the application should be using the Microsoft Kinect and that it should implement natural, predefined gestures. Further requirements are defined by the information gathered through this analysis. First of all the application should be able to find a human hand based on background subtraction and skin color segmentation. By further analyzing the segmented area the application should be able to find fingertips to use for pointing and to recognize a swipe gesture for changing slides. The image processing part of the application should send data to another application running the graphical part, the actual slideshow. In order to be able to do an unbiased experiment of the application in the end, a set of presentations needs to be designed, which will not influence the experiment in any direction, based on their content. The final experiment should compare a traditional PowerPoint presentation with the gesture based application designed and implemented in this project.

5. Design I

In this chapter the design for the overall setup of the application will be presented. Even if this design is based on the finger tracking approach it will also be valid for the skeleton tracking, which will be described in *Part II*. The design will primarily be concerned with designing the setup and interfacing for the gesture based presentation tool. The interfacing for keyboard and mouse is already in place, since the traditional Microsoft Windows control methods will be used, i.e. left clicking or pressing right arrow to move one slide forward. It is necessary though to find out the exact approach to doing the gesture recognition and how to use the gestures in the best possible way during the presentation.

5.1 Setup

Since the application should work in correlation with a slideshow presentation being projected on a surface behind the presenter, most of the setup is already given. The thing that needs to be considered is the position of the Kinect, with respect to the presenter and the presentation. Basically, there are three main positions that the Kinect could be placed, to be able to track the presenter: Behind the presenter, besides the presenter (left or right) or in front of the presenter. All three positions have pros and cons which will be briefly covered here, in order to determine which position is the best.

Placing the Kinect behind the presenter will mean that the Kinect should be placed above or below the projected slideshow, facing the presenter. If placing the Kinect on top of the slideshow, it gives the advantage that the Kinect can look down at the presenter and use the floor as background, which in most cases is non-changing surface. This makes it easier to detect the presenter and hereby the presenter's hand. Furthermore, gestures like the swipe, in which the presenter will only move his hand along one axis, will be easy to detect from a top view from behind the presenter, since there will be an unobstructed view of the presenter's hand for the entire duration of the swipe, given that the presenter faces the presentation when doing this. The disadvantage of this position is that it is difficult to determine what the presenter is pointing at, if using the pointing gesture to highlight something in the slideshow. Even using the depth tracking of the Kinect, it will be difficult to get an acceptable estimate of the presenter's hand in space. Another disadvantage is that it can be rather cumbersome to get the Kinect positioned above the slideshow, especially in big lecture or conference rooms, where the projection area can be several meters high. Placing the Kinect below the projected area is easier, but the problem here is that there is a risk the Kinect will be "blinded" by the powerful light of the projector, since it will be pointed almost directly towards the light bulb of the projector. Furthermore, the audience might also be tracked from this perspective, which is first of all a very inconsistent background (which is bad when doing background subtraction) as well as a background with lots of skin color, which might confuse the skin color detector, described in *4.2 Image processing*.

Having the Kinect placed besides the presenter removes the risk of the Kinect being exposed to the audience – in most cases – as well as preventing the Kinect from being exposed to the strong, direct light of the projector. The disadvantage is that the presenter may easily obstruct the Kinect's view of his hands, which means that he has to be very aware of his position with regards to the Kinect, making the technology rather intrusive. Furthermore, it is also difficult to determine where the presenter is pointing on the slideshow, when this gesture is used. The same is the case with the swipe gesture, unless the presenter makes the gesture while facing the Kinect, which seems very unnatural to both presenter and audience.

Placing the Kinect in front of the presenter makes it possible to detect the presenter's hand when pointing, since the Kinect will track the same plane that the slideshow is being projected to. The xy-coordinate of the hand, tracked by the Kinect, in this position can be mapped directly to the slideshow presentation. This will be described in more details in *5.2 Gesture interaction*.

There is a chance that the presenter might partially cover the Kinect's view of his hand when doing for instance a swipe gesture, if the presenter is facing the slideshow presentation. However, the problem is not as great as with side-view, since the presenter will often, at least partially, face the audience and hereby the Kinect. The front position of the Kinect also poses a problem when doing skin tracking. Since the part of the presenter's hand that is facing the Kinect is also facing the projector, the hand will also be exposed to the projected image, which will cause it to change color and thereby possibly not be detected by the skin color detector. Also, since the slideshow becomes the background in the tracked image, the background will change whenever the slide show change, which makes background subtraction troublesome. In *Table 5.1*, the pros and cons of the different positions of the Kinect are listed:

Kinect position:	**Behind**	**Besides**	**In front**
Pros	Steady background (if placed above slideshow). Visible part of hand is not exposed to projector light.	Steady background.	Interpolation for pointing is easy. Low risk of presenter occluding the view of his hand.
Cons	Difficult to interpolate position of hand when pointing. Cumbersome to place the Kinect.	Difficult to interpolate position of hand when pointing. High risk of presenter occluding the view of his hand.	Unsteady background. Difficult to track skin due to changing colors, caused by the projector.

Table 5.1: List of pros and cons of placing the Kinect behind, besides or in front of the presenter respectively.

Despite the possible problems of tracking the presenter's hand, the position in front of the presenter will be the preferred one and the one used for the purpose of development in this project. Having the slideshow visible from the Kinect's point of view makes it easy to figure out where the presenter is pointing and it supplies for the clearest view of the presenter's hand, which gives the presenter the biggest freedom of movement. The problem of possible bad tracking can be minimized with more training data and the background subtraction can be improved by ignoring the pixels in the image that is at the distance of the projected slideshow. It is safe to assume that the presenter's hand will always be in front of the slideshow, so all pixel data that is at the distance of the slideshow or further away can be filtered out before doing background subtraction, using the data of the Kinect's depth image.

5.2 Gesture interaction

Having decided upon the positioning of the Kinect it is necessary to decide which gestures should be tracked and how. As mentioned in *1.2 Related work* pointing has been part of most other works involving gesture controlled interfaces, such as in the work of (Baudel & Beaudouin-Lafon, 1993) and (Sukthankar, Stockton, & Mullin, 2001). In a normal PowerPoint presentation, the presenter would normally use the mouse, a laser pointer or some other form of pointing device to point out important parts of his presentation. By tracking the user's hand, it is possible to directly map the position of the presenter's index finger to a position on the projected slideshow. Since the depth position of both the presenter's hand and the slideshow is known, based on the information from the Kinect's depth image, it is possible to project a vector from the Kinect to the projected slideshow, passing through the presenter's hand, as seen in *Illustration 5.1*.

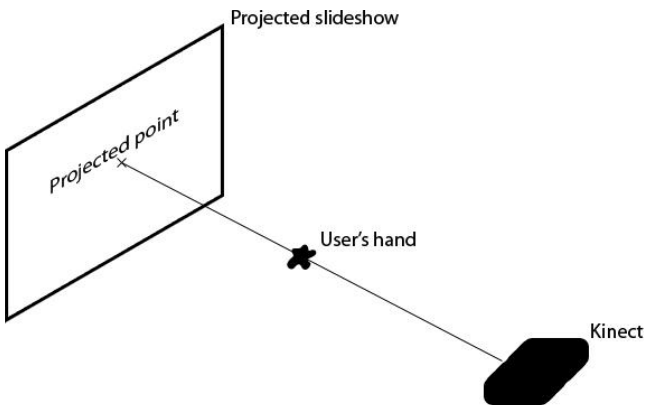


Illustration 5.1: Pointing is achieved by tracing a line from the Kinect camera to the slideshow projection, which passes through the presenter's hand.

Using this approach makes it possible for the presenter to point at areas in the projected slideshow that it would be difficult for him to point at using just his hand in a traditional presentation. If for instance the projected image was much taller than the presenter, he would still be able to point at the top part of the projected slideshow, since the movement of the projected point will be scaled according to the user's distance from the projection area. The downside of this is that the closer the user gets to the Kinect, the more his involuntary movements will be amplified. It is difficult for a human to hold his or hers hand completely still and the small movements of the hand will appear bigger when projected on the slideshow. Furthermore, the tracking might not be completely reliable and might also be the cause of noisy movement. Some of this unwanted movement can be smoothed out by averaging the movement over a given time period – as described in (Sukthankar, Stockton, & Mullin, 2001) – but this will also make the movement slower and the balance between accurate and smooth movement should be balanced through testing.

For the swipe gesture the recognition will happen through the approach described in *4.4 Gesture recognition*, meaning that the velocity along the x-axis will determine whether a swipe has been performed or not. However, in order to avoid that the presenter will accidently change slides during the presentation it is necessary to find a way of determining if the movement of the hand should be identified as a gesture. There are two main approaches to doing this:

1. Using modes where a single gesture or hand pose will change between pointing mode and slide change mode.
2. Using hand poses during gestures, such that a certain movement will only be recognized as a gesture if the hand is in a certain pose.

Each of these approaches has pros and cons. The first approach has the advantage that the swipe gesture recognition can be ignored as long as the user is in pointing mode. The application will only have to do the pointing projection and check to see if the mode change gesture/pose is performed. As described in *4.4 Gesture recognition* this mode change can be as simple as having the presenter move his hand to a certain position in the slideshow. If the mode is changed to slide change mode, pointing can be disabled such that the presenter does not accidently point at something in the presentation while doing the swipe. The downside of this approach is that doing the mode changing action (whether it is a pose or a gesture) might not seem very natural to neither presenter nor audience. Furthermore it is also breaking the flow of the presentation a bit by requiring several steps of interaction from the presenter to i.e. change the slide. The second approach is smoother and more natural since the interaction can happen completely through natural gestures and no time is "wasted" on changing modes. This is also the approach used by (Baudel & Beaudouin-Lafon, 1993) –

mentioned in *1.1 Motivation* – for their data glove based presentation tool. Baudel & Beaudouin-Lafon mentions how finger and wrist positions should be non-usual, but not unnatural when doing a gesture or pose. For the purpose of skipping ahead to the next chapter for instance (basically a slide change), they use an approach where the presenter should stretch all fingers of his hand, then move his hand towards the right and in the end bend all fingers, except the thumb, as if grabbing an object. According to Baudel & Beaudouin-Lafon this approach works well, but they had the advantage of using data gloves, which directly supplies data about which fingers are bent and stretched as well as the fingers' position relative to each other. It can be difficult to obtain data about this using camera tracking for several reasons. First of all, as explained in *4.4 Gesture recognition*, finger tips are found by looking for narrow angles along the contour of the hand, which can give a good indication about what the presenter's index finger is pointing at for instance, but can be unreliable when looking for more detailed information. The pointing is not as sensitive to errors, since pointing can be mapped to the hand's centre of mass if no finger is found and if the application loses track of the hand completely, the only consequence is that the pointer will disappear from the slideshow. If a hand pose like the one proposed by Baudel & Beaudouin-Lafon should be recognized it would be necessary to recognize all five fingers and the tracking of these fingers should be consistent for the duration of the entire gesture. If not, the presenter will not be able to change slides, which will be a lot more frustrating than not being able to point for a short period of time. Another problem is that the fingers might be occluded from the camera's point of view, because they move behind other parts of the presenter's body. Therefore the first approach will be chosen instead. Despite the fact that it might seem more unnatural to some users, the greater reliability of this approach is making it a better choice.

In order to avoid relying heavily on the finger tracking for changing modes in the presentation, a gesture that can be recognized without finger tracking is desired. Several options are possible, including a wave gesture (rapid change in velocity along the x-axis and the hand moving along an arc), a "push" gesture resembling pushing a button (high velocity along the y-axis) or a flick gesture (high velocity along the z-axis). The wave gesture is already used for Kinect on Xbox in order to reactivate tracking, a metaphor for attracting the attention of the Kinect. However, since the Kinect is looking at the presenter from the same point of view as the audience (and not from the projected image, as it is the case when playing Kinect games), it will appear as if the presenter is waving at the audience, which might be very confusing to the audience. Furthermore, when in slide change mode a wave could also easily be misinterpreted by the application as a swipe. The two other gestures do not suffer from this problem, since movement will be measured along the y- and z-axis, which is not used when looking for a swipe. The problem with the push gesture is that it can occur while pointing if the presenter is quickly moving his hand from the top to the bottom of the slideshow, but the advantage is that it is not directed at the audience and the risk of the audience perceiving the gesture as being directed at them is smaller. The flick has the advantage of being unlikely to happen during the presentation as the presenter will rarely move his hand along the z-axis neither for pointing nor swiping. The disadvantage is that it also suffers the risk that the audience might perceive the gesture as being directed at them. The flick is a more discreet gesture than a wave though, since the presenter's hand is moving less, and with a little practice from the presenter the gesture could be performed without distracting the audience noticeably. However, the advantage of the gesture being unlikely to happen during pointing or swiping is stronger than the disadvantage of the risk of distracting the audience. A continuous, involuntary change between presentation modes might prove to be even more distracting to the audience than the short flick.

To conclude the application will be controlled by having two modes: Pointing and changing slides. In pointing mode the user will be able to move a pointer around on the slideshow, which is a direct projection from the Kinect to the projected slideshow moving through the presenter's fingertip. Doing a small flick of the hand towards the Kinect will change the mode and the presenter will be able to swipe left or right in order to change slides forth or back.

5.3 Presentations

Having established the setup of the application and how to control the presentation through gestures, this subchapter will look into the actual presentations and the way the presentation will react to the gestures. As described in the beginning of this chapter the traditional presentation will be controlled using traditional keyboard and mouse interaction and will be presented using Microsoft PowerPoint. As described in *2.1 Gestures* the most used presentation tool amongst the participants of the survey conducted was PowerPoint and in order to supply a good base for comparing the gesture based presentation a PowerPoint presentation will be used to represent a traditional presentation.

For the gesture based presentation an application should be designed that is similar to a traditional presentation. Since the interaction method should be the decisive parameter for the outcome of the test, the gesture based presentation should not differentiate itself greatly from the traditional presentation in style or graphical quality. However, the functionality of the gesture based presentation might have other visual cues to illustrate their execution. In a traditional PowerPoint presentation it is possible to animate the change of slides using different built-in effects, such as the slides fading out, dissolving or moving out of the screen. The default way of changing slides in PowerPoint is an instant change though, meaning that as soon as the presenter presses a button for the next slide, the old slide will disappear and the new one appear. The same could happen as soon as a swipe was completed in the gesture based presentation, but the effect does not match the gesture very well. Pressing a keyboard or mouse button is a binary interaction and thus mapping this action to an instant change of slide is natural, but the swipe is a continuous interaction and an according mapping should be chosen. On the iPhone, and other systems using swipe gestures, the action that the swipe is mapped to is animated to follow the swipe. This furthermore highlights the metaphor of flipping through a book and aids both presenter and audience in seeing the effect of the swipe. The downside is that there will be a short delay while the slides are moving where the presenter cannot continue his presentation. However, this delay will be so small that it should not affect the presentation negatively.

For the pointing an appropriate substitute to the mouse cursor should be found. Even for a traditional presentation the mouse cursor is usually quite small, but as the presenter needs to be operating the mouse on a surface it is usually clear to the audience when they should be paying attention to the cursor. When the presenter is simply using his hand it might not be as clear to the audience that they should be paying attention to the small mouse cursor, rather than the actual position of the presenter's hand. Therefore a bigger visual representation is needed for illustrating the point in the slideshow that the presenter is aiming to highlight. To further satisfy the requests of the respondents in the survey conducted in *2.1 Gestures*, objects that are pointed at will also slightly increase in size to illustrate them being highlighted. There is a risk that this added visual feedback will bias the results towards the gesture based presentation as far as highlighting goes: Pointing at objects using the mouse cursor is not as strong a visual cue as it is to have the objects increase in size. However, since the presenter will be standing in front of the slideshow for the gesture based presentation there is also a risk that he might block the view of the projected point for some of the audience, which will not happen if the presenter is operating a mouse on a table next to the presentation. By highlighting the entire object or sentence that the presenter is pointing at, even people in the audience who cannot see the projected point will be able to know what is highlighted, which justifies the need for such a highlighting.

Since the presentations should work equally well for all content, the design will not cover the specific content of the slideshows to be used in the final experiment. This will instead be covered later on in *9.1 Slideshow content*.

5.4 Sum-up

Based on the information of this chapter it has been determined that for the physical setup of the gesture based presentation, the Kinect will be placed in front of the presenter, looking at the slideshow from the same perspective as the audience will. This will ease the projection of the pointing as well as minimizing the risk of the presenter occluding the view of his hand.

Pointing will be implemented through projecting a vector from the Kinect to the canvas where the slideshow is projected and moving through the presenter's hand. Changing slides will happen with a swipe gesture, right-to-left for changing forward and left-to-right for changing backwards. To sum up the functionality of the presentation tool, the flowchart in *Illustration 5.2* will explain the general flow of the application and how the presenter will use it.

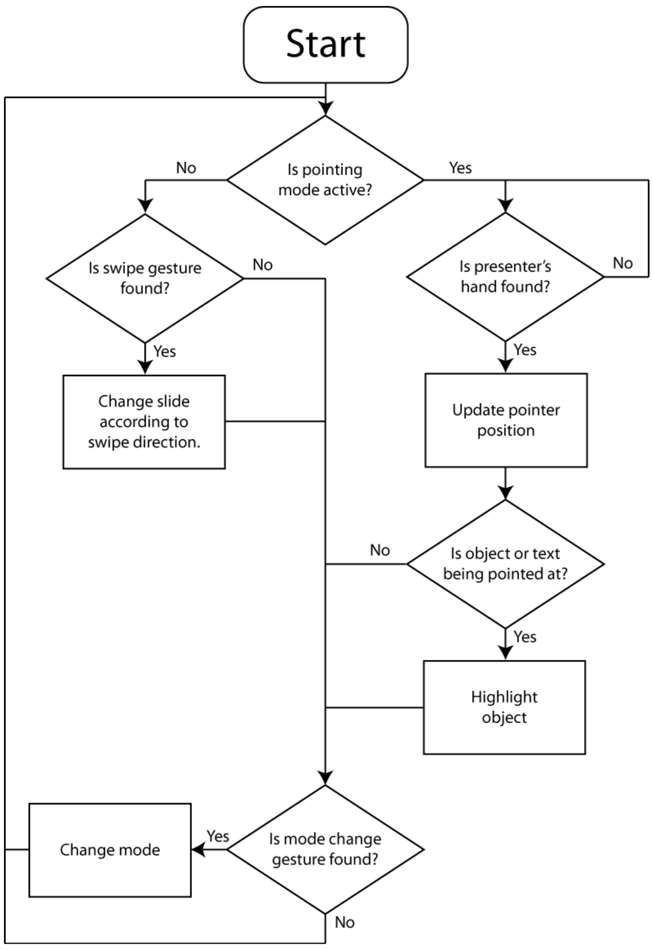

Start
Is pointing mode active?
No
Yes
Is swipe gesture found?
No
Yes
Change slide according to swipe direction.
Is presenter's hand found?
No
Yes
Update pointer position
Is object or text being pointed at?
No
Yes
Highlight object
Is mode change gesture found?
Yes
No
Change mode

Illustration 5.2: Flowchart of the general functionality of the gesture based presentation tool.

In the following chapter (6 Implementation I), the implementation of the finger tracking method will be explained. Since this method was not finalized and connected to an actual slideshow presentation, the implementation chapter will only be concerned with the image processing and feature extraction needed for doing the finger tracking. *Part II* will describe the implementation of the presentations.

6. Implementation I

In this chapter the process of implementing the finger tracking algorithm will be covered. The first part in this process is the training of the skin detector, which is based on processing a number of images containing skin and non-skin, as described in *4.2 Image processing*. Thus, the first part of this implementation chapter will be concerned with describing the program used for training the skin detector to recognize skin. After that, the actual program for tracking the hand and potential gestures will be described.

6.1 Skin detector training

Before using the program for training the skin detector, it is necessary to have manually obtained images for training beforehand. The training images should ideally be taken in the environment in which the skin detector should be used. The non-skin images should contain no skin and the skin-images should contain nothing but skin. Grabbing a non-skin image can thus be done by simply taking one or several pictures of the room in which the setup should be used. The skin-images require a bit more work, since it is necessary to manually cut out the portion of the pictures that are skin colored, using an image editing software. *Illustration 6.1* shows an example of a few images from a training set, including only skin colors.

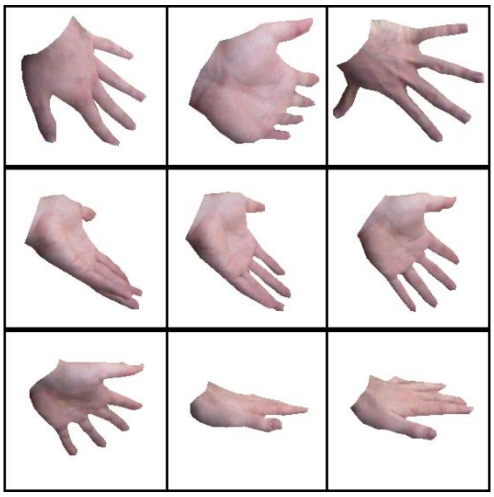


Illustration 6.1: Nine skin-images used for training the skin detector. All images contain only skin colored pixels.

With the images captured, the training program can be executed. What the program does is to run through every pixel in the images and save them in a histogram, based on their color value. As described in *4.2 Image processing*, Malik suggests a bin size of 32 for each color channel, which means that each color value will be assigned to bins ranging over 8 values in the range from 0-255. In the program, the images that are to be trained on are loaded one by one and their pixel values are assessed one by one through a nested for-loop.

```
for(int y = 0;y<trainingImage->height;y++)
{
    for(int x = 0;x<trainingImage->width;x++)
    {
          uchar r, g, b;
          r = datasrc[y * step + x * channels + 2];
          g = datasrc[y * step + x * channels + 1];
          b = datasrc[y * step + x * channels + 0];
//[…]
```

In the code above, `datasrc` is the array of byte values that makes up the image. This array can be accessed through the `IplImage` class, which is OpenCVs class for storing images. The member `imageData` of the `IplImage` class returns a pointer to the array of data in the image. Since OpenCV uses a BGR format rather than an RGB format, the red channel is the last value for each set of three values that makes up a pixel.

With the RGB-values for a given pixel collected, it is possible to figure out the position in the histogram that needs to be incremented. Since a bin size of 32 is used, the following code will determine which bin to use for red, green and blue respectively:

```
rBin = (int)(((float)r/256)*32);
gBin = (int)(((float)g/256)*32);
bBin = (int)(((float)b/256)*32);
```

By dividing the values by 256, which is the entire range of each channel, and multiplying by 32, which is the bin size, the resulting number will be a decimal number. However, by typecasting the value to an integer, all decimals will be capped off and the remaining number will be the correct position for the given color in the histogram. For instance, a value of 155 will result in $155/256 * 32 = 19.375$ and by typecasting to `int`, the number will be 19, which is correct since the range of bin 19 is 152-159. After these calculations, the histogram has to be incremented by one at the right position. Since the histogram is a three dimensional array, the right position is accessed as follows:

```
histogramMatrix[rBin][gBin][bBin]++;
```

When all pixels in all images have been processed, the histogram is saved to a file. For this purpose a CSV (Comma Separated Values) file is used, since it is easy to read and write to such a file. All values are separated by a semi-colon, which makes it easy to read in the histogram later on, when it is needed in the actual skin detector. Furthermore, programs such as Microsoft Excel are able to read CSV files and in Excel, each value that is separated by a semi-colon will be put in a new cell, which also makes it easier to debug the produced histogram after training. The training needs to be done on both skin and non-skin images, such that two different histograms are created, H_s and H_n respectively.

6.2 Skin detector

With the histograms in place for skin and non-skin, it is now possible to do the actual skin detection in a video stream. For this implementation the skin detector was build upon a sample code project from the OpenKinect library, which means that there are parts of the code which are there only to be able to keep the Kinect running and supply video feeds from the Kinect. These parts of the code will not be explained in details here, but can be seen in the file called *glview.c*, which is included on the attached CD. There are mainly two functions from the *glview.c* file that will be explained here: The `main` function and the function called `ImageProcessing`. Furthermore, a list of other functions is included in the *ImgProcMethods.h* and *ImgProcMethods.cpp* (both files can be found on the attached CD), which will be explained throughout this chapter. This chapter will explain the code in the way it is executed in the program and thus the first part is the `main` function, since that is the starting point of the code.

6.2.1 Function: main()

The main purpose of the `main` function is to get the necessary variables initialized and getting the various streams opened. First of all, an output stream is opened for writing to a text file, which is to be used by the graphics application for knowing when certain gestures has been made. However, due to the change of approach briefly described in *3 Methodology*, the file stream was not used in the final implementation and thus will not be described any further in this report. This chapter will instead mostly focus on the skin detection algorithm, since that is what could be utilized in future implementations, combining the approaches from *Part I* and *Part II*.

The next important part of the `main` function is the loading of the trained histograms (line 605 and 606 of *glview.c*). The function `readHistogram` can be found in *ImgProcMethods.cpp*. The function is given a path input for where to find the histogram CSV file and a three-dimensional array for storing the array. It then runs through the file, storing the values from the CSV file into the array, which can be used in the skin detection. Having read in the histograms it is possible to get the total count of the skin and non-skin histograms by looping through the arrays, adding the values of all bins to the total count, which can then be used to calculate the values $P(skin)$ and $P(\neg skin)$ as described in equation *4.4* and *4.5* from *4.2 Image processing*. The rest of the main function is primarily concerned with setting up threads for running the Kinect video streams and will not be covered in more details here, since it was part of the original OpenKinect sample code. Once the threads are running, the function `DrawGLScene` (the sample code uses OpenGL to render the video streams) is called each frame and from this function the `ImageProcessing` function is called, which is the function doing the skin segmentation and feature extraction.

6.2.2 Function: ImageProcessing()

The first parts of the `ImageProcessing` function is, like the main function, primarily concerned with initializing variables, such as the `IplImage` pointers used to store the image data. The `GLTex2CVImg` function (called in line 134 and 135 of *glview.c*) is converting the OpenGL textures into OpenCV images, which can then be used for the necessary image processing. The function is declared in *ImgProcMethods.cpp*.

The first step of the image processing is to resize the input image. The original size of the RGB video stream from the Kinect is, as described in *2.2 Human tracking hardware*, 640x480 pixels, but in order to increase performance, the image is reduced to a 320x240 image. This improves the frame rate and does not significantly reduce the quality of the tracking. It might be possible to go to even lower resolutions in some cases, but for this implementation the 320x240 resolution seemed to be a good balance between speed and accuracy of tracking. After the resizing, the input image is smoothed using a 5x5 Gaussian kernel, before doing the background subtraction.

Background subtraction

The background subtraction is a simple comparison between a previously acquired image of the background and the current image of the video feed. Pressing space at any point during the application will save the current picture of the feed as a new background and smooth the image, also using a 5x5 Gaussian kernel. The following lines of code show how the background subtraction is performed using OpenCV functions:

```
cvAbsDiff(fg, bg, bgDiff);
cvCvtColor(bgDiff, bgDiffGray, CV_RGB2GRAY);
cvThreshold(bgDiffGray, bgDiffGray, 6, 255, CV_THRESH_BINARY);
cvCopy(fg, result, bgDiffGray);
cvCopy(result, fg);
```

In the code above, `fg` is the current image of the video feed and `bg` is the saved background image. `result`, `bgDiff` and `bgDiffGray` are locally declared `IplImage` pointers. First of all, the absolute difference is found between the current image and the background. Since an `IplImage` is essentially just an array of values representing the image, the difference is found for each pixel value stored in the two arrays. *Illustration 6.2* shows an example of two single channel, four pixel images being subtracted using this function.

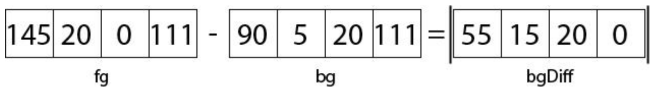


Illustration 6.2: Two single channel, four pixel images being subtracted using the OpenCV function cvAbsDiff.

The resulting image is then transformed into a grayscale image in order to be able to do a thresholding of the image using `cvThreshold`. When using this function, the source image (first parameter) is thresholded according to the method given by the fifth parameter. In this case a binary threshold is done, which means that every value below the threshold value, given by the third parameter in the function call, will be set to 0, while the rest of the values will be set to the maximum value given by the fourth parameter. This means that at every pixel position with a small difference in color value, the binary image will be black while the rest will be white. The threshold value might change according to conditions, but a value of 6 proved to be working well for the situations in which this application was tested. The binary image can now be used as a mask for finding the foreground. Using the `cvCopy` function the content of one image is copied into another image of the same size and if a mask is specified, only the pixel values that are above 0 in the mask array will be copied. So copying the content of `fg` into the empty image `result` using the `bgDiffGray` as a mask will complete the background subtraction. In the end, the content is copied back to `fg`, which is the pointer for the image that will be used throughout the rest of the image processing.

After the background subtraction further noise reduction is done, performing an opening operation by first eroding and then dilating using the default OpenCV values, meaning a 3x3 rectangular structuring element and using it only once. This is done since the background subtraction might produce noisy pixels where the difference between the current image and the background was higher than the threshold, although that part of the image should have been considered background.

Skin segmentation

Having a background subtracted image the actual skin color tracking can begin. This happens in the function called `SegmentSkin`, which is declared in *ImgProcMethods.cpp*. The function takes in eight parameters: A source and a destination image, the skin and non-skin histogram, the total count of the histograms and the $P(skin)$ and $P(\neg skin)$ values. The source image is the background subtracted image, which is the one skin detection is performed on, and the destination image is where the resulting segmentation will be stored. Through a nested for-loop every pixel value is checked for the entire image, in order to test for the probability that it is skin colored, using the approach described in *4.2 Image processing*. First the color value of the given pixel is checked for in the two histograms, using the function `getProbability`, which finds the value in the three-dimensional histogram-array that represents the count for that given color. This value is then divided by the histograms total count, which gives the value for $P(rgb|skin)$ and $P(rgb|\neg skin)$. Having calculated these values, all the variables of equation *4.3* are found and the value of $P(skin|rgb)$ can be calculated. The code below shows how this was implemented.

```
// Nested for loop, that goes through every pixel in the webcam feed
for(int y = 0;y<sourceImage->height;y++)
{
    for(int x = 0;x<sourceImage->width;x++)
    {
        uchar r, g, b;
        r = datadst[y*step+x*channels+2] = datasrc[y*step+x*channels+2];
        g = datadst[y*step+x*channels+1] = datasrc[y*step+x*channels+1];
        b = datadst[y*step+x*channels+0] = datasrc[y*step+x*channels+0];

        pRGBs = getProbability(r, g, b, histogramMatrixSkin, sTotal);
        pRGBns = getProbability(r, g, b, histogramMatrixNonSkin, nTotal);

        if (pRGBs == 0 && pRGBns == 0)
            pSkinRGB = 0;
        else
            pSkinRGB = (pRGBs*pS)/((pRGBs*pS)+(pRGBns*pN));

        if (pSkinRGB <= 0.6/*sigma*/)
        {
            datadst[y * step + x * channels + 0] = 0;
            datadst[y * step + x * channels + 1] = 0;
            datadst[y * step + x * channels + 2] = 0;
        }
    }
}
```

As it can be seen in the code above an error check needs to be performed before the calculation of $P(skin|rgb)$. Because division is involved, it is necessary to check that division by 0 does not occur, since that will give invalid values for $P(skin|rgb)$. Division by 0 can occur if a color is found in the processed image, which was found in neither the skin nor non-skin histogram. In this case, both $P(rgb|skin)$ and $P(rgb|\neg skin)$ will be 0 and thus there will be division by 0 in calculating $P(skin|rgb)$. However, if such a color value is found it is safe to assume - given that the training data has been extensive enough in covering skin color - that the given color is not of skin color and can thus be rejected. Thus, if the values of `pRGBs` (representing $P(rgb|skin)$) and `pRGBns` (representing $P(rgb|\neg skin)$) are both 0, `pSkinRGB` (representing $P(skin|rgb)$) will be set to 0 too. If either `pRGBs` or `pRGBns` is different from 0, the calculation of `pSkinRGB` is done according to equation *4.3*. Knowing this value it is possible to determine whether the pixel color should be considered skin colored, based on the probability threshold. According to (Malik, 2003) a value of 0.6 was effective in their work and a similar value seemed to work well in this application too.

In the end opening and closing operations are performed again, in order to first get rid of any noise and afterwards close holes in remaining blobs. In this case a 4x4 cross shaped structuring element is used, since the effect of using opening and closing operations with a cross shaped structuring element are not as powerful as when using a rectangular one. Fewer pixels are affected by the cross and thus the shape of the contour is more accurately preserved.

Feature extraction

After the skin segmentation it is possible to do the feature extraction, which finds the fingertips. Using the function `cvContour` will produce a pointer of the type `CvSeq`, which can hold sequences of different OpenCV data types. In this case it will hold a list of all the contours that OpenCV finds in an image. For the initial prototype the assumption was made that the biggest contour would be the desired hand. This is not a safe assumption to make in a final prototype, but the calculations done in the algorithm can be used in an application where the hand is correctly found in the image. Given a contour stored as a `CvSeq*` it is possible to run through the elements of the contour and get the coordinates of the points that makes up the contour. The following code shows how this is done:

```
for(int i=0; i<result->total; i++)
{
      CvPoint pt = *(CvPoint*)cvGetSeqElem( result, i );
      pointList.push_back(pt);
}
```

The `pointList` is a C++ vector of `CvPoint`s and `result` is the contour that the coordinates should be found in. This means that at the end of the loop, the `pointList` will contain all coordinates of the points around the contour. The function `FindTipsAndDips`, declared in *ImgProcMethods.cpp*, will run through such a list of points and look for angles above a certain threshold. As described in *4.3 Feature extraction* the angles are found by going through the points and creating the vectors v_1 and v_2, where v_1 is the vector between the current and the previous point and v_2 is the vector between the current point and the next point in the list. The angle to check is then the angle between these two vectors. However, as it can be seen in *Illustration 6.3*, the list of point is quite extensive and it would make no sense to check the angle between vectors of three consecutive points in the list, since they are placed so close together.

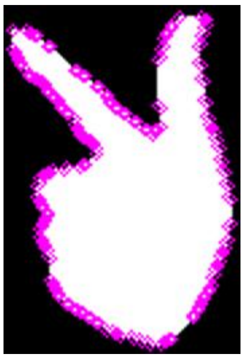


Illustration 6.3: Having found all the points along an OpenCV contour produces quite an extensive list of points. All the magenta colored circles are points along the contour, returned by OpenCV.

Instead of only going one position back and forth in the `pointList` for each calculation a constant value k is used to decide the amount of steps to step back and forth in the list for each calculation. In (Malik, 2003) a value of 16 is used and since their work is also based on an OpenCV approach, the same value was used in this application. The angle between the vectors is found through the cosine relation described in *4.3 Feature extraction*. For the sake of more easily setting the correct threshold for the angle, the value of θ is converted from radians to degrees and the absolute value of the angle is stored, since it does not matter if the angle is negative or not. The following piece of code calculates the angle between vectors:

```
v1 = cvPoint(p1.x - p2.x, p1.y - p2.y);
v2 = cvPoint(p3.x - p2.x, p3.y - p2.y);

double angle = acos(dotProduct(v1, v2) / (sqrt(dotProduct(v1,v1)) *
sqrt(dotProduct(v2,v2))));
angle *= 57.2957795;
angle = abs(angle);
```

In the work by (Malik, 2003) the threshold for deciding if an angle should be considered a fingertip is 30 degrees, however it seemed that in the case of this application the threshold should be set somewhat higher in order to identify fingertips. A threshold of at least 60 degrees seemed to be necessary in order to reliably detect fingertips and dips.

In the end the application was able to detect a hand and correctly identify fingertips and dips, as showed *Illustration 6.4*. At this point there was not differentiated between tips and dips for the fingers.

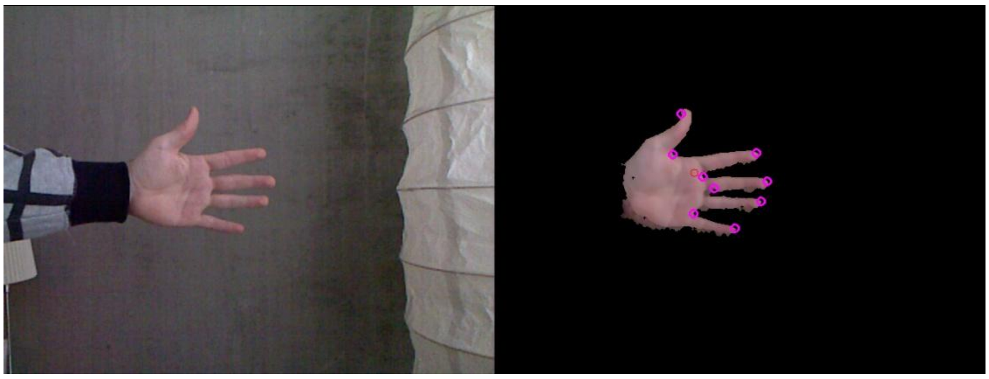


Illustration 6.4: Using the code described in this chapter will be able to track a hand and identify fingertips and dips as seen in the image above. The picture on the left is the original input image and the picture on the right is the segmented image. Magenta colored circles are detected tips or dips and the red circle is the centre of mass.

6.2.3 Problems and issues

The code described in this chapter did work, given the right conditions. However, there are a number of problems and issues that needs to be corrected or improved in order for the application to work flawlessly. First of all there are latency problems, which is due to the large amount of image processing being done. This was partially improved by reducing the image sizes and also reducing the amount of times the skin segmentation is done, such that this process was only being performed every second frame. Furthermore, for the prototype the actual presentations was planned to be made as Flash applications and communication with Flash through writing and reading to/from a text file was not very efficient either. Communication through sockets would probably work better performance wise.

Another problem was the actual skin segmentation and background subtraction. The background subtraction was very sensitive to changes in light condition, because the background was based only on a single image of the background. In a presentation context and for the purpose of an application like this, background lighting will be changing often because of the change of content of the slideshow and thus a more robust and reliable background subtraction should be found. An approach to a more reliable background subtraction can be seen in (Ivanov, Bobick, & Liu, 2000). The problem with the skin segmentation is related to the background subtraction. If the background subtraction is not reliable enough a lot of pixels that should not be considered foreground would be included in the skin segmentation. During the testing of the application it was discovered that a lot of pixels in the background had color values that were in the range of skin color and thus would be detected as skin color if the pixels were not discarded during the background subtraction. However, even if the background subtraction was successful it might be necessary to supplement the skin segmentation with some sort of pattern recognition to get a reliable tracking of the hand.

6.3 Sum-up

To conclude on this implementation it was found that the algorithm could work under ideal conditions. It could find a hand and fingertips and the information about the position of the hand and fingers could be used to do gesture recognition. However, the application proved to be too unstable to do a reliable experiment with this implementation and with the time frame of the project it was not realistic to produce a reliable implementation. Thus it was necessary to take another approach and as described in *3 Methodology* I was made aware of another code library that would aid in producing a stable, reliable application to be used for testing. This is what will be described in the upcoming chapters, called *Part II*.

Part II

Part II will be concerned with the skeleton tracking method, which is supplied through a coding library from the organization OpenNI. The analysis in this part will mostly be concerned with analyzing and describing the code library. Since the design will be based on the design of Part I, no design chapter is included in this part, but a new implementation chapter is describing the application that will be used for the final test is.

7. Analysis II

This analysis chapter will be covering the OpenNI code library and the drivers needed to utilize this library in correlation with the Kinect. Due to the OpenNI library including an implementation of skeleton tracking that can directly be used for development, it is not necessary to do any image processing. Furthermore, the chapter will look into approaches as to how to make the actual presentations that should be controlled through gestures.

7.1 OpenNI

OpenNI is a non-profit organization established in November 2010 with the goal of introducing natural interaction applications into the marketplace (OpenNI, 2011). The organization was established by four companies working within the realm of natural interaction, robotics, games and hardware. OpenNI supplies open source code for using different sensor types that can be used in correlation with developing natural interaction application, including 3D sensors, RGB camera, IR camera and microphones. One of the founding companies of OpenNI – PrimeSense – has developed a middleware library called NITE, which was originally created for PrimeSense's own 3D tracking device called PrimeSensor (PrimeSense, 2010). The PrimeSense is in many ways similar to the Kinect - since it also supplies an RGB image, depth image and audio stream - and in fact Microsoft developed the Kinect system in correlation with PrimeSense, based on the technology of the PrimeSensor (McElroy, 2010). That means that PrimeSense has been able to develop and publish drivers for the Kinect to work on a PC. Using these drivers in correlation with the NITE middleware library makes it possible to get access to full skeleton tracking in 3D as it is known from the Kinect games.

The NITE library is written in C++, which fits well with the choice of programming language chosen in *4.1 Programming language*. However, PrimeSense also developed a wrapper for NITE, which makes it usable directly in the Unity engine. Hereby it becomes unnecessary to send data from a C++ application doing the tracking to a Flash or Unity application showing the graphics. Not only does this ease development, but it is also decreasing latency. Doing development in Unity, using the PrimeSense wrapper, requires programming in C#, but as described in *4.1 Programming language* I have experience from previous projects with working in Unity and with C# programming, so the change of programming language is not a major drawback.

7.2 Skeleton tracking in Unity

Using the NITE library from OpenNI it is possible to directly access data about the skeleton of the tracked user and it only takes a few lines of code to access the needed data. The actual wrapper for Unity is a class called `NiteWrapper`, which is supplied in a sample code provided in the PrimeSense middleware. The sample is designed to be used in a Unity application, where users can take control of two soldier avatars. The purpose of the sample is to show how to get the users calibrated with the skeleton and how to access data about the skeleton. The `NiteWrapper` class is setting up some structures and functions needed for doing the calibrating and tracking. The functions are imported from a dll-file (UnityInterface.dll) also supplied in the library. The functions include methods for looking for new users, getting options about joint transformations and other useful functionality. With the wrapper class in place it is possible to start making the actual application. The sample code shows the setup of how the initialization is done and will not be covered in details here. It is worth noting that in order to initialize the tracking in NITE, the user is required to stand in a certain pose, which can be seen in *Illustration 7.1*.

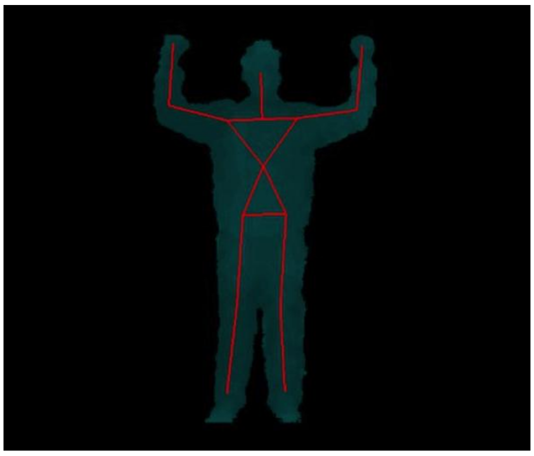


Illustration 7.1: In order for the NITE library to correctly initialize the skeleton, the user is required to stand in the pose shown here. After a few seconds the NITE library will have mapped the skeleton to the user.

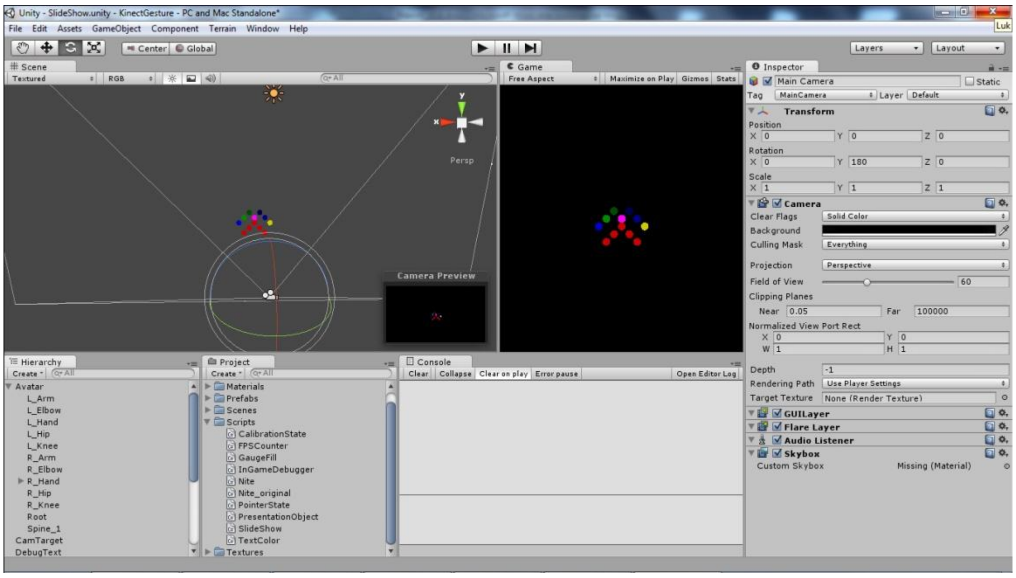
Once the user is calibrated it is possible to get information about the joints of the skeleton. Accessing joint data is possible using only two lines of code:

```
NiteWrapper.SkeletonJointTransformation trans = new
NiteWrapper.SkeletonJointTransformation();
NiteWrapper.GetJointTransformation(userId, joint, ref trans);
```

The `SkeletonJointTransformation` is a struct defined in the `NiteWrapper` class and contains two other structs called `SkeletonJointPosition` and `SkeletonJointOrientation`. The struct for position contains three float values for position in space (x, y, z) and a value for confidence, in the range [0;1], which determines how secure the tracking is. The orientation struct also contains a confidence value, but instead of x, y and z values it contains a 3x3 matrix where the orientation is stored. Calling the `GetJointTransformation` function will fill the `SkeletonJointTransformation` called `trans` with the data for the skeleton joint specified by joint for the user specified by `userId`. The `joint` parameter is an enum also specified in the `NiteWrapper` and `userId` is an integer value identifying the desired user.

Looping through all available joints with the two lines of code described above will thus supply information about the user's position in space. To use this in Unity, the script needs to be attached to a so-called GameObject in the "game world", which is called the *scene* in Unity. In *Illustration 7.2* the development environment for Unity can be seen and the scene is what can be seen in the upper left window. Unity works in the way that all scripts that are attached to an object in the scene view will be activated when the application is started. In the hierarchy view (lower left) all game objects in the scene are listed. As described previously in this chapter the NITE sample code is designed to work with a soldier avatar. It does this by looking for objects with key names to identify what is for instance an elbow or a knee and apply the position and transformation to these objects. For instance, it will look for a GameObject called "L_Elbow" and apply transformation and orientation data about the left elbow to this object. By doing this, the NITE script can be attached to any object in the scene and still be able to find the joints that it needs to modify, based on the names. The downside of this approach is that the game objects making up the skeleton have to exactly match the names that the script is looking for and that only one object can be called i.e. L_Elbow.

Essentially it is not necessary to have a skeleton in the scene in order to get data about the skeleton joints, but it supplies good debugging information as it is possible to see that the skeleton in the game world moves according to the user.


Unity - SlideShow.unity - KinectGesture - PC and Mac Standalone*
File Edit Assets GameObject Component Terrain Window Help
Center Global
Layers Layout
Scene Textured RGB
Persp
Camera Preview
Game Free Aspect Maximize on Play Gizmos Stats
Console Clear Collapse Clear on play Error pause Open Editor Log
Luk

Hierarchy
Create
- Avatar
  - L_Arm
  - L_Elbow
  - L_Hand
  - L_Hip
  - L_Knee
  - R_Arm
  - R_Elbow
  - R_Hand
  - R_Hip
  - R_Knee
  - Root
  - Spine_1
- CamTarget
- DebugText

Project
Create
- Materials
- Prefabs
- Scenes
- Scripts
  - CalibrationState
  - FPSCounter
  - GaugeFill
  - InGameDebugger
  - Nite
  - Nite_original
  - PointerState
  - PresentationObject
  - SlideShow
  - TextColor
- Textures

Inspector
Main Camera Static
Tag MainCamera Layer Default

Transform
Position X 0 Y 0 Z 0
Rotation X 0 Y 180 Z 0
Scale X 1 Y 1 Z 1

Camera
Clear Flags: Solid Color
Background
Culling Mask: Everything
Projection: Perspective
Field of View: 60
Clipping Planes: Near 0.05 Far 100000
Normalized View Port Rect: X 0 Y 0 W 1 H 1
Depth: -1
Rendering Path: Use Player Settings
Target Texture: None (Render Texture)

GUILayer
Flare Layer
Audio Listener
Skybox
Custom Skybox: Missing (Material)

Illustration 7.2: The Unity development environment. In order for scripts to become active on execution, they need to be attached to an object in the scene. All scene objects can be seen in the hierarchy.

As it can be seen in *Illustration 7.2* a skeleton is made up of 12 spheres, each named to match the names the NITE script is looking for. They are collected in a parent object called "Avatar". Another object in the scene contains the script, but this object is "empty", meaning that it does not contain any model, mesh, texture or similar. It is invisible in the scene and is thus useful for containing scripts that should control several other game objects. With the skeleton objects in the scene – named correctly – and the NITE script attached to an empty GameObject the application will be able to track a user and apply his movement to the skeleton.

In order not to get too detailed about the mechanics of Unity, this chapter will not cover Unity development in more details. Other elements of development will be covered during implementation, but the basics have been covered in this chapter. The implementation of the application using skeleton tracking will be built on the same setup and design as the finger tracking approach. Minor changes was made throughout the implementation, because of issues discovered through implementation, but since the basic design remains the same for both approaches a new design chapter will not be introduced in this part. The following chapter will instead explain the implementation of the skeleton tracking application.

8. Implementation II

In this chapter the development of the skeleton tracking application will be covered as well as how the tracking was combined with controlling a slideshow presentation. The first part of the chapter will be concerned with tracking of the user's hand and recognition of gestures and finally how this is connected to the control of the slideshow.

8.1 Hand tracking

Unlike the finger tracking application it is not necessary to start coding from a main-function, since the Unity engine takes care of all the basic necessities to make the program run. When scripting in Unity and making classes to be used directly in the application, the classes should inherit from a base class called `MonoBehaviour`, which supplies some basic overridable functions, such as an update function conveniently named `Update`, which is called by the Unity engine every frame. In the file called Nite.cs (found on the CD attached to this report) a class called `Nite` is defined, which is the class derived from `MonoBehaviour`. Thus it also contains the update function. Different variables are initialized in the function called `Start`, which is another overridable function from the MonoBehaviour class called at the beginning of program execution. However, this is part of the original NITE sample code and will thus not be described in details here. The update function of the `Nite` class looks as follows:

```
void Update()
{
    // Next NITE frame
    NiteWrapper.Update(false);

    // update the visual user map (comment out this for increased performance)
    //UpdateUserMap();

    // update avatars
    foreach (KeyValuePair<uint, SoldierAvatar> pair in calibratedUsers)
    {
        pair.Value.CastArmRay();
        pair.Value.UpdateAvatar(pair.Key);
    }
}
```

First of all the function makes sure to update the `NiteWrapper` class, which updates the position and orientation of the skeleton joints. The `UpdateUserMap` function is for debugging purposes and is commented out for increased performance. If used it will display a video feed of what the Kinect is tracking. Lastly a loop is running through all tracked users and their attached avatars. Since there will only be one presenter present it is not strictly necessary to loop through users, but the code piece is also part of the sample code and is left as is. This is also the reason why the `SoldierAvatar` class is named the way it is, since the original sample was designed to control a solider avatar.

The `CastArmRay` of the `SoldierAvatar` class is the one taking care of the pointing and will be covered later on in *8.1.1 Pointing*. The `UpdateAvatar` function is what updates the joints of the avatar in the scene, which is the only thing it is responsible for in the sample code. However, in order to get gesture recognition working for the purpose of the application developed in this project, another class was developed, named `GestureRecognizer`. An instance of this class is created in the `SoldierAvatar` class and a call to the `GestureRecognizer`'s update function is called from the `SoliderAvatar`'s update function, supplying the newly acquired information about the position and orientation of the presenter's hands.

As described in *4.4 Gesture recognition* the swipe gesture will be recognized by checking if the velocity along the x-axis is above a certain threshold. Whenever the update function of the `GestureRecognizer` class is called it will update the velocity of the hands in the x-, y- and z-axis, based on the previous position of the hands. For instance, the calculation of the velocity of the right hand is done as follows:

```
currVelRH.x = (currPosRH.x - lastPosRH.x)/deltaTime;
currVelRH.y = (currPosRH.y - lastPosRH.y)/deltaTime;
currVelRH.z = (currPosRH.z - lastPosRH.z)/deltaTime;
```

Based on this calculation the gesture recognition can happen and is done by checking the movement along the x-axis to recognize a swipe gesture. The threshold for velocity was found by testing and in the end a threshold of 1000 units/second was found to work well for identifying a swipe.

According to the design described in *5.2 Gesture interaction* the application should have two modes: One for pointing and one for changing slides. The mode control is also taken care of in the update function of the `GestureReconizer` class, as seen below:

```
Gesture perfGest = checkGesture();

if (!isPointingActive)
{
    if (perfGest == Gesture.RSWIPE)
    {
        SlideShow SlideShowControl = projArea.GetComponent<SlideShow>();
        SlideShowControl.MoveForward();
        isPointingActive = true;
    }
    if (perfGest == Gesture.LSWIPE)
    {
        SlideShow SlideShowControl = projArea.GetComponent<SlideShow>();
        SlideShowControl.MoveBackward();
        isPointingActive = true;
    }
}
```

The function `checkGesture` checks to see what swipe is performed (left or right) and if the application is not in pointing mode, defined by the variable `isPointingActive`, the application will perform the action connected to the gesture. For the purpose of the presentations to be done in the experiment of this project – and likely in most presentation situations – the presenter will only want to move one slide ahead and then start explaining something on the new slide. Thus the application will go back to pointing mode as soon as a slide change has been made. In this way the user does not have to make the mode change gesture to go back to pointing after changing slides.

The `checkGesture` function was also supposed to take care of tracking the mode change gesture, but this was complicated by the fact that the NITE library cannot actually supply information about the position and orientation of the hand directly. Instead it is using the information about the elbow joint and considering the hand as a natural extension of the elbow. This is in most cases a correct assumption – at least for the position - because the length of the user's lower arm is constant. The distance of the hand joint from the elbow can be changed to fit a given user, making the position correct for most cases. However, the rotation of the wrist cannot be found by the same method and the orientation of the hand joint is simply applied the same orientation as the elbow. This means that it is not possible to identify gestures where only hand rotation is involved. Ideally the flick gesture should work by measuring the change of velocity of the hand along the z-

axis, but through development it became obvious that it would require to powerful a movement of the hand to be able to recognize the flick by velocity alone. The presenter would have to make a long, fast flick towards the Kinect (and hereby the audience) in order to get a robust recognition that would not be confused with the presenter simply moving his hand around in front of the slideshow. Since it was not possible to track the correct orientation of the hand with the skeleton tracking the activation area approach described in *4.4 Gesture recognition* was used instead. In this approach no defined pose was used for changing mode, but instead the presenter would have to hold his hand over a certain area in the presentation for a short period of time, similar to the approach used by (Sukthankar, Stockton, & Mullin, 2001) for activating their drawing mode.

8.1.1 Pointing

The pointing was also changed slightly, compared to the original design described in *5.2 Gesture interaction*. It was not possible to get a tracking of the finger tip in the skeleton tracking approach. Furthermore the NITE library does not directly supply information about the position of non-human objects in the scene, which made it cumbersome to do the projection approach of tracing a vector from the Kinect to the slideshow moving through the presenter's hand. Instead the information about the elbow joint was used to compute a vector of pointing direction. Using the position and orientation of the elbow it was possible to use the presenter's lower arm as a pointing device, essentially drawing a vector from the elbow through the hand and onto the plane in the Unity scene that made up the presentation, as seen in *Illustration 8.1*.

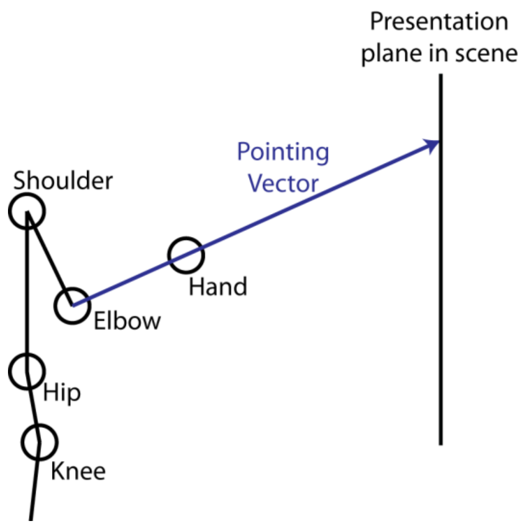


Illustration 8.1: Pointing in the skeleton tracking approach happens through creating a vector from elbow, through the hand and onto the presentation plane in the Unity scene.

This way of pointing has the advantage that the presenter can mostly stand in one place to do the pointing. It only requires movement of the arm in order to point at the extremities of the presentation, such as the top corners. The original pointing approach would require the user to move around in front of the Kinect, in order to be able to point at the corners for instance. The disadvantage is that the presenter cannot move the pointer simply by moving his finger or rotating his wrist. It requires the presenter to move his hand around.

The way to implement this pointing in Unity is through the use of ray casting, which is a built-in functionality of the Unity engine. Unity provides a class called `Physics`, which contains several functions for doing physics-related operations, including ray casting. The function is called `Raycast` and has a total of 12 overload functions, but in this chapter only the method used in the implementation of this application will be explained. The call to the `Raycast` function used in this implementation looks as follows:

```
if (Physics.Raycast(rightHand.position, rightElbow.right, out hit, Mathf.Infinity)
){
```

The first parameter is the origin of the ray, which should be casted from the position of the right hand. The next parameter is the direction of the ray and it uses the `right` vector of the elbow. In Unity any object is equipped with a `Transform`, which is a class containing information about the object's position and orientation in space. Included in the Transform are three variables called `up`, `forward` and `right`, which represent the object's orientation in space. The NITE library rotates the joints such that the right vector aligns with the orientation of the bone that follows. This means that the right vector of the elbow points in the direction of the user's lower arm, towards the hand, so this is also the vector to be used for casting the ray. The third parameter is a reference to a variable of the type `RaycastHit`, which contains any information about the collision point, including what object the ray has hit, if any. Lastly the distance of the ray is set to infinity, since there should not be any limitation on the pointing ray. The `Raycast` function will return true if an object was hit and otherwise false, so only when it returns true further analysis will be done.

In case the pointing ray hits an object there are three different cases that should be considered: If the ray hit the change mode area (top of the presentation), if it hit a text item or if it hit an object that should be highlighted. For checking if the ray hit the change mode area, it is checking if the name of the colliding object is "Switch". The area that will change the mode when being pointed at is defined by an object in the scene, placed on top of the presentation area, called "Switch". The object is transparent so it cannot be seen during the presentation, but the ray can still intersect with the object. Each time a ray hits an object, a reference to that object will be stored for the next frame and if the ray hits the object called "Switch" two frames in a row, a timer is incremented. Once the timer reaches one second the mode is changed. If the presenter moves the pointer away from the switch object the timer is reset. The code below shows the implementation of this:

```
if (collider.name == "Switch" && lastCollider.name == "Switch")
{
    pointerStateControl.Animate();
    switchTime += Time.deltaTime;
    if (switchTime > 1.0f)
    {
        gr.SetPointing(!gr.IsPointingActive());
        switchTime = 0.0f;
        pointerStateControl.SetActive(gr.IsPointingActive());
        stateChangeTimeOut = 2.0f;
    }
}
```

Apart from changing the mode of the presentation a small visual indicator is also changed. This object is controlled by the `pointerStateControl` object, which is being animated (a crosshair rotating in the upper left corner) while the timer is incrementing. Once the mode changes the indicator change, either to the rotating crosshair with a red cross over, if the mode is changed to slide change mode, or simply disappear if the presentation go back to pointing mode. This is done only to indicate to the presenter when he is changing modes. Furthermore the `stateChangeTimeOut` variable is set, which is a way to secure that the presenter does not change mode back by keeping his hand over the switch object too long. Once this variable is above 0, it will be decremented every frame by the time passed since last update and pointing will be disabled for the duration of the timeout (2 seconds) and thus the mode cannot be changed either.

For pointing at objects or text in the presentation the same approach is being used as for pointing at the change mode object. The difference is that instead of looking for all the names of the objects that could possibly be highlighted, the algorithm will look at the colliding objects layer. Layers are another part of Unity and basically work as a tag that can be applied to objects in order to group them. Text would be highlighted by increasing the size, but when highlighting for instance objects in a graph, a size increase is not a prudent way for highlighting, since it will make the graph display incorrect information, as exemplified in *Illustration 8.2*.

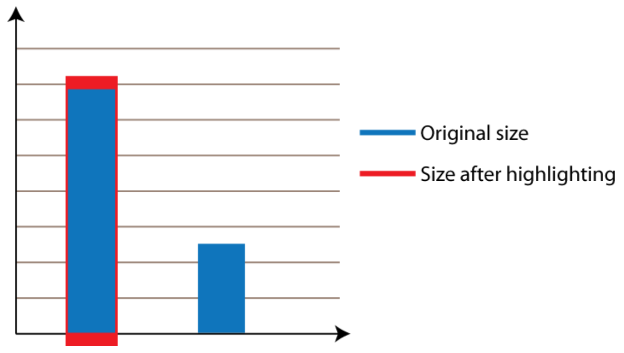


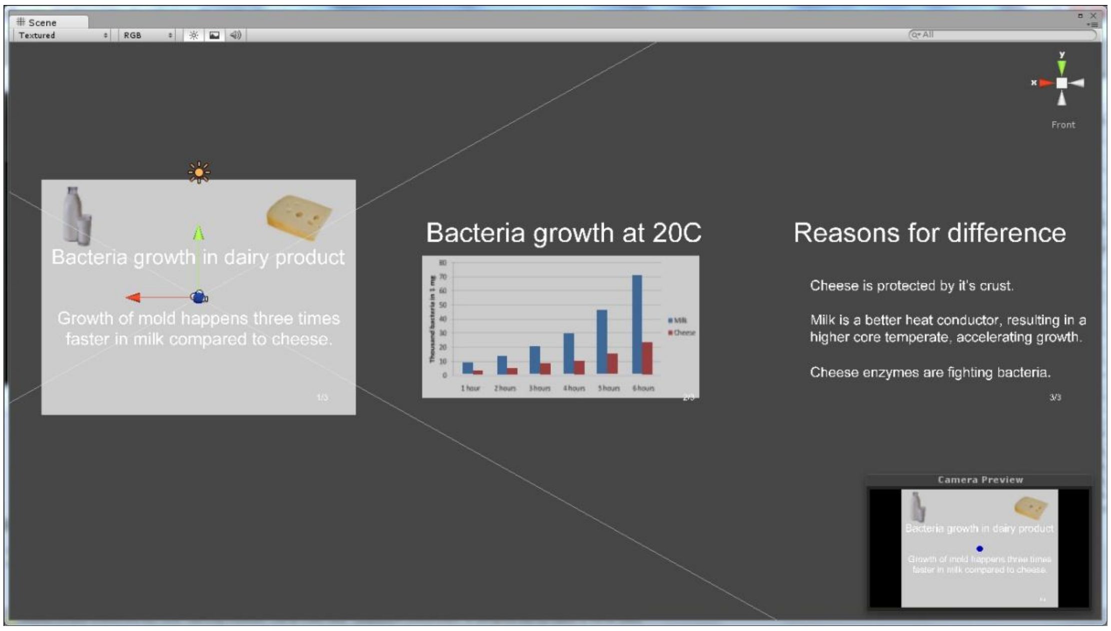
Illustration 8.2: If highlight for instance bars in a graph by increasing their size, the new size might confuse the audience about what value the bar is really representing.

Due to this issue objects and text were placed in different layers. Text layers would, upon being pointed at, increase 10% in size while the objects would instead change color. Hereby the size of graph objects would not confuse the audience by changing in size, but it would still be noticeable that the object is highlighted. When the presenter moves the pointer away from a text or object it will return to its original size/color.

This concludes the implementation of the gesture recognition in the skeleton tracking approach. Swiping gestures were implemented for changing slides back and forth and a pointing method using the information about hand position and elbow orientation were created. As the last part of the implementation chapter the Unity scene will be described, which is what will display the actual presentations.

8.2 Unity scene

In order to display presentations to the audience, using the gesture based approach described previously in this chapter, the presentation content should be placed in the Unity scene. In order for the swipe to work as described in *5.3 Presentations*, with the slides moving out of the screen, the content were arranged side by side in the order it was supposed to be presented, as shown in *Illustration 8.3*. By collecting all the content in one parent object, all the content will move when the parent is moved, such that when swiping it will look as if the current slide is moving out of the screen in one side and the new one enters from the other side.



Bacteria growth in dairy product

Growth of mold happens three times faster in milk compared to cheese.

Bacteria growth at 20C

Reasons for difference

Cheese is protected by it's crust.

Milk is a better heat conductor, resulting in a higher core temperate, accelerating growth.

Cheese enzymes are fighting bacteria.

Illustration 8.3: By arranging presentation content on a line in the order it was supposed to be presented, it was possible to animate the content when the presenter performed a swipe. The window in the lower right shows what would be projected on the canvas.

For indicating where the presenter is pointing a blue dot is used. This dot is moved around on the slideshow according to where the ray of the ray casting hits an object. The position of the blue dot pointer is being set every frame to the coordinates of the collision point between ray and object. If no object is hit by the ray the pointer will stay at the last collision point.

Apart from the objects seen in *Illustration 8.3* a series of other objects are present in the scene, but not visible. This is for instance the change mode object, which is a rectangle covering the top part of the presentation area as well as debugging objects, such as an indicator to the presenter about whether or not he is calibrated with the skeleton. This information is only visible on the screen of the laptop running the presentation and will not be described in more details.

8.3 Sum-up

Using the Unity engine an application has been developed that can show a slideshow presentation, which can be controlled by gestures. The application has two modes: Pointing mode and slide change mode. In pointing mode the presenter can point at items in the presentation, which will be highlighted either by an increase in size (for text) or by a change of color (objects). In slide change mode the presenter can change slide by swiping. Swiping from right to left will move ahead to the next slide, by the current slide moving out to the left of the presentation and the new slide entering from the right. The opposite is true when doing a left to right swipe. The presenter can change from pointing mode to slide change mode by pointing at the very top of the presentation and keep pointing there for one second. Whenever a slide change has been made the application will automatically return to pointing mode.

With this application in place it is possible to compare the application to a traditional presentation using PowerPoint. The following chapter will cover this experiment and the results of it.

9. Test

In order for the final experiment to answer the final problem statement, it is necessary to look at the perceiving audience and their experience of the given presentations. As stated in *4.5 Testing*, the experimental setup should use a 'same participants' approach, comparing the application designed and implemented during this project with a traditional presentation tool, in a way that none of the approaches are biased to give a better experience than the other. The null-hypothesis will be that the two methods are equally good, i.e. gives the test subjects an equally good understanding of the presentation content and an equally good experience of the way it is presented. Furthermore, the questions asked before, during or after the experiment should be directed towards clarifying the test subjects preferred style of presentation.

In order to compare the gesture controlled presentation to a traditionally controlled presentation in a 'same participants' approach, the content of the two presentations should be similar, but not identical. It is important that the test subjects are not biased towards the presentation they are shown last, due to them knowing the content beforehand. When doing a presentation it is often to explain, teach or introduce a certain topic to the audience and therefore the gesture based presentation should also be evaluated based on its capability of conveying the necessary information to the audience. The test subjects in this experiment might be influenced by seeing the presenter doing gestures to control the slides and focus more on the movement of the presenter than on understanding the content of the presentation. Therefore it is important that the test subjects are presented to different materials in each test case, but that the material should be structured in the same way, i.e. using the same amount of slides and approximately the same amount of picture and text. In this way, it is possible to afterwards ask the users about the understanding of each topic and see which of them they showed the greatest understanding of. By then randomizing which presentation type shows which topic, it is possible to minimize the influence of both learning effects and differences between the two presentation topics.

As described in *2.1.1 Online survey about use of digital presentation tools*, the most popular presentation tool amongst the respondents of the online survey was Microsoft PowerPoint. In order to compare the gesture based presentation tool to something that is already familiar for the test subjects, a PowerPoint presentation will be used to represent the traditional presentation tools in the experiment. Thus for each group of test subjects to perform the test, they will each be given a presentation using PowerPoint with keyboard/mouse and a presentation using the application created in Unity with gestures as the interaction method.

The test should be set up in a way that resembles a realistic presentation situation as close as possible, in order to get an accurate image of the test subjects' perception of the presentation tools. Thus the test setup should be performed in a room where it is possible to use a projector for projecting the slideshow presentation onto a canvas and where there is room for a group of people to watch the presentation. With this setup the test subjects will also possibly feel less monitored and less stressed, since it does not seem as if they are the ones being tested. They are simply asked to watch and follow the presentations. The problem with the setup is that it is not possible to get feedback from the test subjects during the presentation and therefore all questions will have to be asked before and/or after the test session. In order to avoid training users to use the systems and avoid having them practice presenting the slideshows, I will be performing the presentations in all test cases.

9.1 Slideshow content

In order to create two presentations that are not biasing the test in any specific direction, it is important to find content that will not give advantages to test subjects who would happen to have knowledge of the given topic. Therefore, the two presentations used in the test are made up stories and facts, which should appear to be realistic, such that the test subjects believes in them and are interested in listening. Furthermore the presentations should contain facts and information that can be illustrated through traditional slideshow elements, such as graphs, bullet points, images and plain text. These are often used elements in slideshows and should be familiar to the test subjects. Furthermore it supplies the opportunity to put details into the presentations that can be used to test the subjects' awareness of the content, such as the numbers in a graph or important information from a bullet point list.

The two presentations created are about book sales in USA and bacteria growth in dairy products, but all information are completely fictional. Even if test subjects should have prior knowledge about either of the topics, they will not stand a better chance of answering the questions correctly afterwards, because numbers and information are not taken from real sources. Both presentations will consist of three slides, arranged as follows:

1. Front page including a headline, a short descriptive text and two pictures related to the topic.
2. A headline and a graph comparing two parameters over a given timescale.
3. A headline and a bullet point list, concluding the information given in the previous slide.

The slideshow will be presented in this order and approximately one minute will be spent on each slide in total. When reaching the final slide, the presenter will move one slide back to once again point out something in the graph. This is done in order to be sure that both changing slide forwards and backwards is utilized in the presentation. This is one of the most common activities in a normal presentation and there might be differences in changing slides forward compared to backwards that can reveal problem areas in either of the two presentations. The presenter will also highlight text and parts of the graph in order to be able to compare highlighting objects with a mouse, compared to doing it with a pointing gesture.

9.2 Questionnaire

The questions to be asked in correlation with this test should primarily focus on two main areas: The users' understanding of the presented content and their opinion about the presentations. Even if a user should state that he preferred one presentation tool over the other, the presentation might not have succeeded in conveying the necessary information to the user about the presentation topic and as such the preferred presentation might not actually be an improvement compared to the other presentation method.

9.2.1 Understanding

With regards to the understanding of the content, it is possible to ask the test subjects questions about this in the break between the two test cases. The advantage of this approach is that the test subjects have the presentation fresh in memory in both test cases, when answering questions about understanding. If questions of understanding are asked after the test subjects have been through both test cases, there is a risk that their memory of the presentation they were shown first might be blurred by the information they are given during the second presentation. On the other hand, test subjects might be annoyed by the fact that they asked to answer questions in the break between the two presentations and test subjects might be more aware of details in the second presentations, because they might have an idea of what to look for after having answered questions about the first presentation. However, in order to give an unbiased impression of the test subjects understanding of the presentations, the questions of understanding will be asked after each presentation.

In order to get an indication about how big an understanding the test subjects have of each presentation, the questions should test the test subjects in varying details about the content. First of all, the test subjects should simply be asked to briefly refer what they were told during the presentation. If the test subjects has paid attention to what was said during the presentation they should be able to briefly describe what they had been presented to. Apart from subjective understanding of the presentations' content there should also be more specific questions regarding the content. These questions can be formed as multiple choice questions, where users are asked specifically about certain things of varying importance in the presentation. The more questions the test subjects can answer correctly, the more successful the presentation has been in conveying the information to the users. These questions will concern pictures in the presentation, units on the graph shown and information given on the bullet point list.

The questionnaire concerning the test subjects' understanding can be seen in *Appendix 13.2 Final test questionnaire*.

9.2.2 Opinion

With regards to the test subjects' opinion about each of the presentation styles, the questions should be more focused on getting comparable data that can clarify which of the two presentations were preferred. By using a scale ranging from 1-5 for instance, it is possible to compare the scores about each of the presentations, in order to get an indication about which one the users preferred. Asking the test subjects about which presentation they preferred overall is one part of this assessment, but the comparison should be more detailed than that, in order to conclude what exact parameters made one presentation preferred over another. For instance, the test subjects might prefer one presentation because of its content. Even if trying to make the presentations equally interesting, there might be things in the presentation which makes test subjects prefer one over the other. Therefore it is important to get information about the test subjects' opinions about the content, to be able to evaluate the contents influence on test subjects' preferences. Furthermore the presenter's movement might also have an influence on the test subjects' opinion of the presentation. Since the gesture based presentation will likely be a new experience to most of the test subjects it might be distracting or confusing to see the presenter perform gestures to navigate the slideshow, which could negatively influence their opinion of the gesture based presentation. This information is important with regards to the final problem statement, since it will be difficult to make a gesture based presentation that will be perceived as better by the audience – compared to traditional presentations – if the audience is generally annoyed by seeing the presenter doing the gestures. Even if the application is using natural and, according to the survey described in *2.1.1 Online survey about use of digital presentation tools*, meaningful gestures for the functionality implemented, it might still be confusing for users that these gestures are actually affecting the slideshow. Finally, the test subjects should be asked towards their opinion about the functionality that has been mapped to gestures, i.e. changing slides and pointing. It is also important with regards to the final problem statement to figure out if test subjects prefer the way the gesture controlled presentation is displayed. For instance, slides change by moving sideways in the gesture controlled presentation, to accentuate the metaphor of flipping through a book, whereas slides change instantly in a normal PowerPoint presentation. Users might prefer the instant change of slides, because that is what they are used to and there is no delay in waiting for the new content to show up.

The questionnaire concerning the test subjects' opinion can be seen in *Appendix 13.2 Final test questionnaire*.

9.3 Conduction

The test was setup in a lecture room at campus in which there was a projector as well as room for a small audience. The projector was hanging from the ceiling and on a table below the projector the Kinect was placed, both facing towards the canvas on which the slideshow was projected. On a table to the left of the presenter the laptop that was running the presentations was placed. This way the presenter could follow the presentation on the laptop screen and use the mouse for pointing at objects and text in the traditional presentation. The test subjects were sitting at a table further back. An illustration of the entire test setup can be seen *Illustration 9.1*.

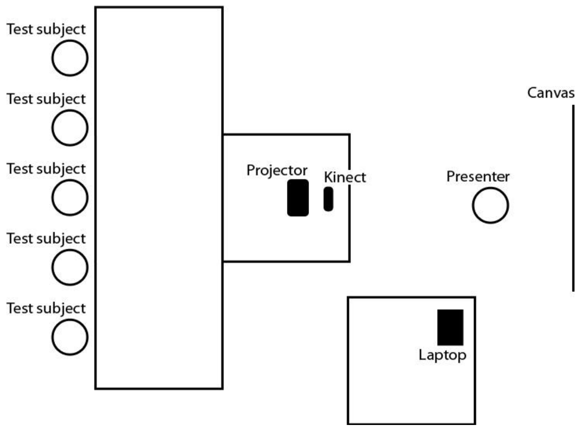


Illustration 9.1: The setup for the test of the presentation tools. The presenter would be standing in the position depicted above only for the gesture based test case. For the traditional presentation he would be standing at the laptop to be able to control the presentation.

For each test the test subjects would first be placed at the table they were supposed to sit at and were then instructed to the experiment. In order not to bias their thoughts in any direction they were only told that they were going to watch two presentations and that they, after each presentation, would be given a questionnaire about the content of the given presentation. Furthermore they were told that they would be given a third questionnaire about their opinions of the presentations after they had seen both presentations.

After those initial instructions the presenter would present the first presentation and afterwards the test subjects were each given the questionnaire about understanding of the content that matched the given presentation. When all test subjects had filled out the questionnaire the next presentation would be given and at the end of this presentation, test subjects were given the according questionnaire about content understanding. When that questionnaire was filled out test subjects were given the questionnaire about their opinion of the presentations.

9.4 Test results

The test was conducted on a total of 20 people, five for each of the four combinations of tests, which were:

1. Traditional presentation about bacteria growth followed by gesture presentation about books sales.
2. Gesture presentation about book sales followed by traditional presentation about bacteria growth.
3. Traditional presentation about books sales followed by gesture presentation about bacteria growth.
4. Gesture presentation about bacteria growth followed by traditional presentation about book sales.

All test subjects were student at Medialogy from 2nd to 6th semester and were thus used to being presented to new knowledge through slideshow presentations, due to the classes they have been taking during their education. In this chapter the data from the test will be described and afterwards analyzed. As described in *4.5 Testing* the test conducted will be based on a rejecting a two-tailed null hypothesis and thus an according value of p should be chosen. For this test, the table of significance found in (NIST/SEMATECH, 2011) will be used and a significance level of 0.05 is chosen as that is the most common level used (NIST/SEMATECH, 2011). Since the test is two-tailed a p value of $0.05/2 = 0.025$ will be used.

9.4.1 Understanding

With regards to the understanding of the presentations, the purpose of the questions was first of all to determine if the presentations had been unbiased as they were designed for. Each test person was able to give a brief summary of what they had been presented to, so none of them were completely distracted from the presentations. However, there were differences when looking at the other questions asked about understanding. When comparing the presentations based on their content, the users were able to answer more questions correct about the presentation of the growth of mold in milk and cheese (from here on referred to simply as MC) compared to the presentation about the decline in American book sales (from here on referred to simply as BS). The difference was especially noticeable in the recognition of pictures, where all of the 20 test persons were able to remember the pictures in the MC presentation, while only 11 where able to correctly remember the pictures of the BS presentation. For the rest of the questions, the amount of answers was more evenly distributed. 17 of the test subjects in the MC presentation was able to remember the unit of the axis of the graph they were asked about, compared to 15 in the BS presentation. 16 test subjects were able to remember what the columns in the graph represented in the MC presentation, compared to 17 in the BS representation. Finally, 19 test subjects were able to give at least one reason from the slideshow about the cause of growth of mold, were 17 were able to name at least one reason about the decline of book sales.

When comparing the users understanding based on the presentation method, the results were slightly less biased, although test subjects seemed to be more aware of the content in the normal presentation. The remembrance of pictures were distributed with 14 correct answers for the gesture based presentation and 17 for the normal presentation. The most significant difference in this case was with regards to remembering the unit of the data on the axis. 19 of the test subjects were able to remember the unit in the normal presentation, while only 13 was able to do the same in the gesture based presentation. As for comparing which presentation were given first and last, the amount of correct answers were almost evenly distributed with regards to all questions.

Looking at the test subjects own rating of the slideshows, given on the graded scale in the opinion questionnaire, the two presentations were rated almost equally in all cases. Generally the test subjects rated the presentations rather low, when looking at how interesting they thought the content was, but none of the presentations seemed to be significantly more interesting than the other. Comparing between the gesture based presentation and the traditional mouse/keyboard based presentation, the mean values were 2.4 and 2.7 respectively ($t = 0.92; p > 0.025$). *Illustration 9.2* shows the distribution of the test subjects' rating of how interesting the presentations was, when comparing gesture and traditional presentation.

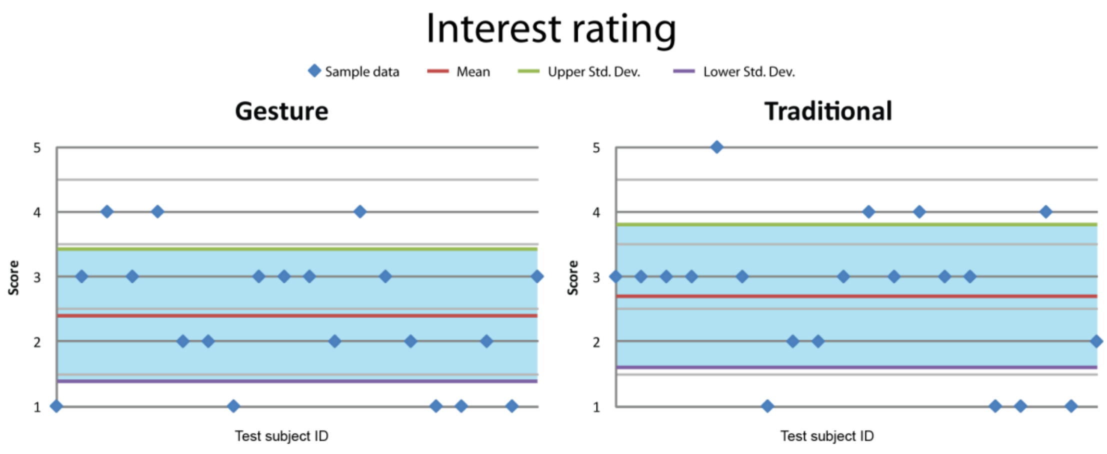


Illustration 9.2: On the question about how interesting the test subjects found the presentation, there was only a small difference between the test subjects' rating in the gesture based presentation compared to the traditional one.

When comparing between the order of the presentations, there was not a significant difference either, when looking at how interesting the test subjects found the presentations. On average, the score given to the presentation that was showed first was 2.65, while it was 2.45 for the presentation shown last ($t = 0.61; p > 0.025$). The same pattern seemed to be the case when comparing the content. For the MC presentation the mean rating was 2.5, while it was 2.6 for the BS presentation ($t = 0.31; p > 0.025$).

Another indicator about the presentations being unbiased was the test subjects responses about how easy it was to understand the content of the presentations. The test subjects generally rated the presentations to be easy to understand and there was no significant difference when comparing in any of the cases. Comparing between the gesture and traditional presentation, the mean rating of how easy the presentations was to understand was 4.4 for the gesture presentation and 4.5 for the traditional presentation ($t = 0{,}32; p > 0.025$) on a scale were 5 was very easy and 1 was very difficult. When comparing the presentation between if they were shown first or last, the mean rating was exactly the same at 4.45. Lastly, when comparing between the content of the information, the mean rating was 4.35 for the MC presentation and 4.55 for the BS presentation ($t = 0.64; p > 0.025$).

9.4.2 Opinion

The first question on the opinion questionnaire was asking the test subjects which of the two presentations they preferred. 55% preferred the traditional presentation and 35% preferred the gesture based presentation. The remaining test subjects did not prefer one presentation over the other. However, 55% also preferred the BS presentation over the MC, while 50% preferred the last presentation compared to 40% preferring the one they saw first.

The first of the rated questions, designed to supply data about test subjects' opinions of the presentations, was the question about how distracting the test subject's found the presenter's movement during the presentation. The scale for this question ranged from 1 (not being distracting at all) to 5 (being very distracting) and for this question there was a significant difference between the gesture and traditional presentation. On average, test subjects rated the presenter's movement at 3.05 for the gesture presentation, while rating it at 1.55 for the traditional presentation ($t = 4.49; p < 0.025$). The distribution of answers for this question can be seen in *Illustration 9.3*.

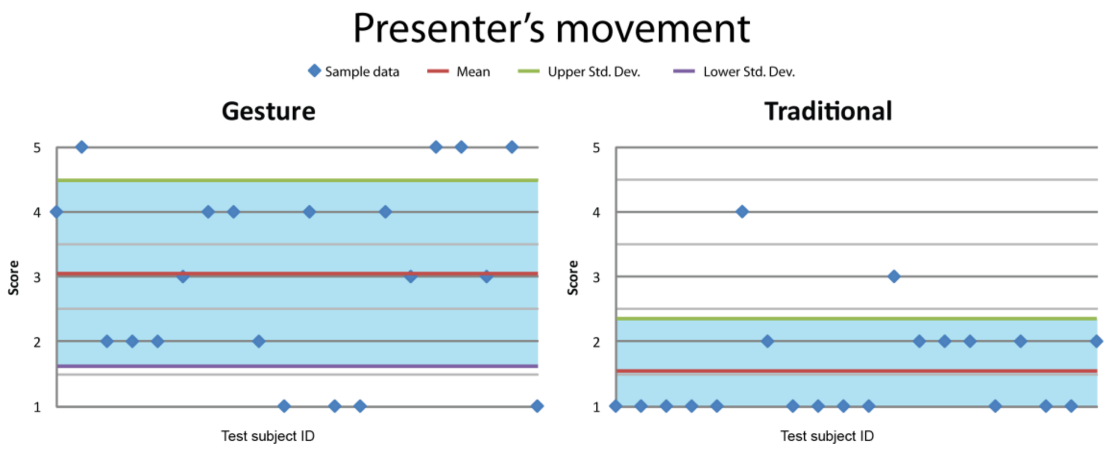


Illustration 9.3: Despite a greater standard deviation, the mean value of the responses about the presenter's movement in the gesture presentation was higher than in the traditional presentation (a score of 5 means very distracting), meaning that test subjects in general found the presenter's movement more distracting in the gesture presentation.

When comparing the responses to this question between which presentation was shown first and last, there was no significant difference. The mean score for the presentation showed first was 2.4 and 2.5 for the presentation shown last ($t = 0.54; p > 0.025$). There was no significant difference when comparing between presentation content either, with both the MC and BS presentation being rated at an average of 2.3.

The next part of the questionnaire was about rating the different functionalities of the presentations, including the change of slides, the highlighting of objects and the highlighting of text. For the change of slides there was no significant difference between the gesture based and traditional presentation. However, there was an indication towards the traditional way of changing slides being preferred, with a mean rating of 4.2 compared to the mean score of 3.6 for the gesture based presentation ($t = 1.90; p > 0.025$) on a scale were 5 meant a very effective change. *Illustration 9.4* shows the distribution of scores given in the question about rating the change of slides.

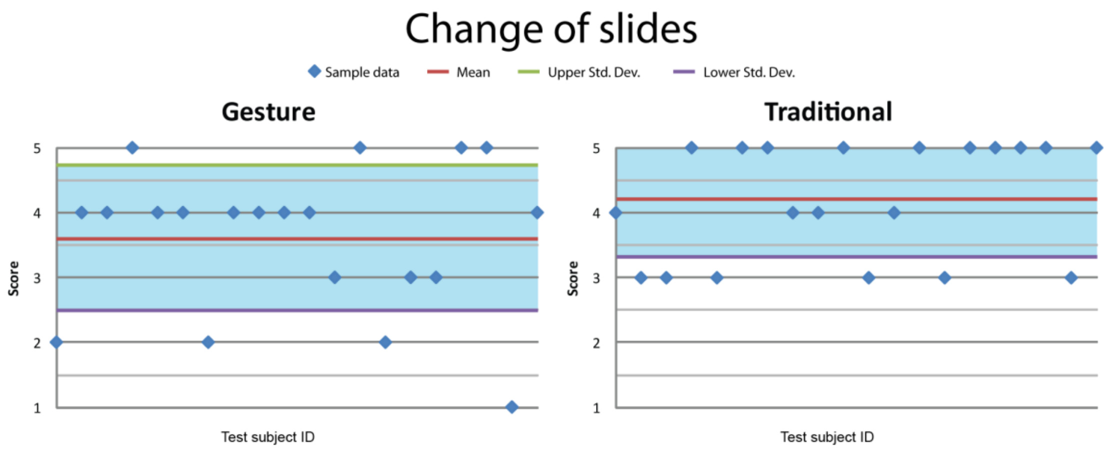


Illustration 9.4: The rating of the way the slides were changed in the two presentations showed no significant difference, although there was an indication towards the test subjects preferring the traditional way.

When comparing the change of slides based on which presentation was shown first and last and on the content of the presentations, there was no significant difference either. However, the presentation being showed first scored 4.1 on average, where as the presentation shown last scored 3.7 on average ($t =$

$1.25; p > 0.025$). The difference was less noticeable when comparing between the content, where the MC presentation scored a mean of 3.8 and the BS presentation scored a mean of 4.0 ($t = 0.62; p > 0.025$).

When looking at the way objects and text was highlighted, there was a significant difference between the gesture based and the traditional presentation in both cases. For highlighting objects the mean rating given by the test subjects was 3.55 for the gesture based presentation and 2.1 for the traditional presentation ($t = 4.28; p < 0.025$) on a scale were 5 meant very effective.

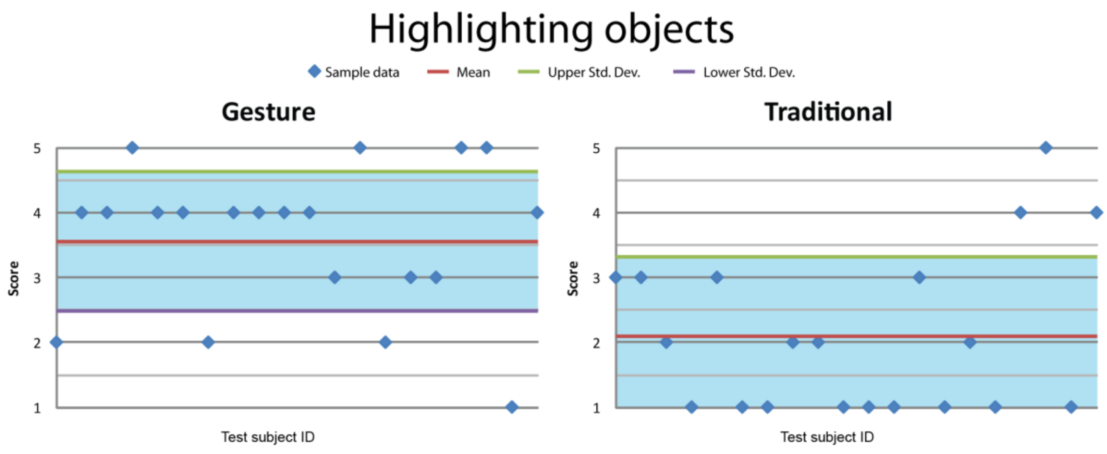


Illustration 9.5: For highlighting objects there was a significant difference between the gesture based and traditional presentation, with the gesture based presentation scoring the highest.

The highlighting of objects did not seem to have been influenced by order or content of the presentations. The mean score given to the presentation that was showed first was 2.80, while it was 2.85 for the presentation showed last ($t = 0.14; p > 0.025$). Comparing between the content of the presentations the MC presentation scored an average of 2.65, while the BS presentation scored an average of 3.0 ($t = 0.96; p > 0.025$).

The same pattern could be seen when looking at the highlighting of text. Again, there was a significant difference between the gesture based and the traditional presentation, with the gesture based presentation scoring a mean of 3.65 and the traditional one scoring a mean of 2.05 ($t = 4.85; p < 0.025$).

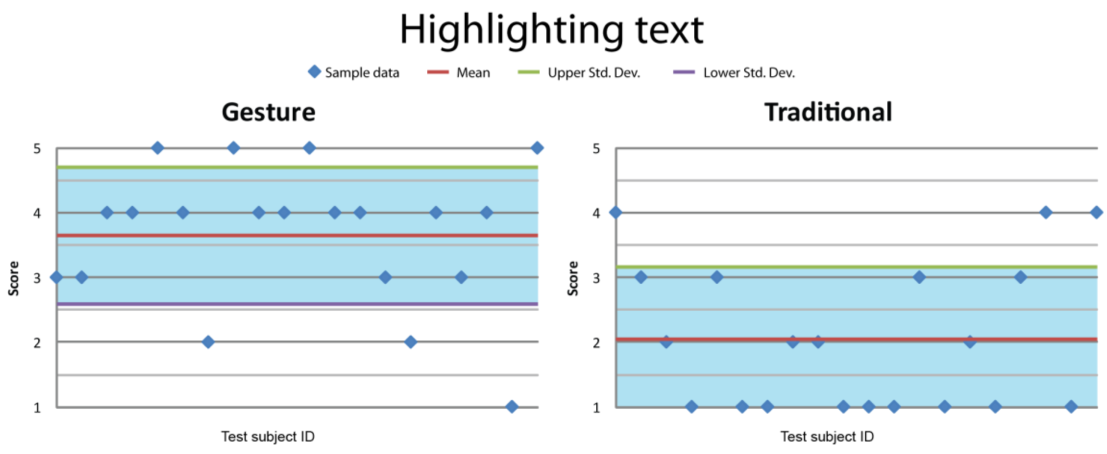


Illustration 9.6: The highlighting of text also showed a significant difference between the two types of presentations, with the gesture based presentation scoring higher on average.

Again, the order and content of the presentations did not seem to influence the test subjects' rating of this question. For the order of the presentation, the presentation that was shown first scored a mean of 2.85, which was the same mean rating obtained by the presentation shown last. When comparing between the presentations' content the difference in mean rating was similarly low, with MC scoring a mean of 2.75 and the BS presentation scoring an average of 2.95 ($t = 4.85; p < 0.025$).

Apart from the quantitative data presented in this chapter there was also some qualitative data collected, based on the test subject's response to the presentations in general. The replies can be seen in *Appendix 13.3 Qualitative test data* and some of the most important comments will be referred to during the analysis of the data in the following chapter.

9.5 Test analysis

In order to make an analysis that will supply any valid information about the hypothesis it is important to determine whether or not the two presentations being compared were unbiased in the test. Looking at the questions about the test subjects' understanding of the two presentations, there were differences in how many correct answers were given when comparing the presentations based on their content. The biggest difference was found in how many of the test subjects were able to remember the pictures shown, where almost only half of the test subjects were able to correctly remember the pictures in the BS presentation, where as all test subjects remembered the pictures of the MC presentation. This difference is difficult to give a good explanation for, since pictures were placed in the same position and were of the same size in each of the two presentations. Possible reasons might be that the test subjects were more interested in the content of the BS presentation and took more time to read and interpret the text. However the two presentations were rated almost equally by the test subjects when it comes to how interesting the content was. None of the other parameters that was measured on for comparing the two presentations showed differences that big and in total it seems the presentations were equally interesting and easy to understand, based on both the questions of understanding and the test subjects' replies in the questionnaire about their opinions of the presentations. Some users commented that the test might be uncertain, because they were paying more attention to the presentation they were shown last, but as explained in *9.2 Questionnaire* the randomizing of order and content between groups of test subjects were designed to minimize this effect and the results based on the understanding questionnaires show that this seem to have worked out.

As seen in *9.4 Test results* there were only significant differences when comparing between gesture based and traditional presentation. Neither the order nor the content of the presentations seemed to matter for the test subjects, which is further indications that the presentations were indeed unbiased. In general the presentations were rated poorly (2.4 and 2.7) for how interesting they were, but that was to be expected given that the content was not related to anything normally taught at Medialogy. On the other hand, the test subjects seemed to be able to easily understand the content and since both presentations were rated similarly on both these questions, it is not expected to have influenced the test in the favor of one of the presentations.

The way of presenting the content – i.e. using gestures or mouse/keyboard - did not seem to make it any more interesting or easy to understand. In general the traditional style of presentation was rated a little higher on both how interesting and how easy to understand it was, although it was not a significantly higher score than the gesture based presentation. What did significantly differ between the two styles was the test subjects' rating of highlighting objects and text, which both scored significantly higher for the gesture based presentation. However, it is worth noting that one test subject commented that he did not notice any highlighting in the traditional presentation. In this presentation text and graphs were highlighted by pointing with the mouse cursor, which is the normal way of pointing out something in a PowerPoint presentation (unless other devices like laser pointers are used), but it might also have been unclear to the test subjects that this pointing was supposed to highlight things. In the gesture based presentation text was highlighted by being slightly expanded and objects were highlighted by changing their color, which is a clearer visual cue.

A problem with the gesture based presentation is that a good visual representation is needed to let the presenter know where he is pointing the hand in the presentation. The blue dot chosen to represent where the presenter was pointing in this setup seemed to annoy the users, based on the comments they gave afterwards. Partly because of the size, bust mostly because of the jagged movement caused by the uncertainty of the tracking and the unstable movement of the presenter's hand. This might be solved by smoothing the movement even more, but the system should also remain responsive to the presenter. Another option could be to show the pointer only on the laptop screen from which the presentation were executed, but that would require the presenter to keep his eyes on the laptop screen while presenting. Despite the fact that test subjects' were annoyed by the chosen cursor in the gesture presentation, some of them still stated afterwards that they preferred the gesture based presentation.

The action of changing slides in the gesture based presentation also had a visual cue attached to it, since it was necessary for the presenter to know when it was possible to change slides. Without the possibility of finger tracking, the change of slides happened by raising the arm for a second and then swiping left or right, as described in *8 Implementation II*. The visual indication to the presenter about the presentation changing between pointing mode and slide change mode was also noted by one of the test subjects as being distracting. Furthermore, the short pause while waiting for the mode of the presentation to change also made it difficult for the presenter to keep the flow of talk fluent. Given a robust and reliable finger tracking, the slide changing might be improved significantly, since a change of slide could be done by the presenter making the right gesture with his fingers and then swiping. This would make it possible to get rid of the visual representation in the slideshow about the current mode of the presentation.

When looking at the test subjects' response to the presenter's movement during the presentations, there was a significant difference between the two presentation styles. Test subjects thought that the presenter's movement was significantly more distracting in the gesture based presentation than the traditional presentation. During the traditional presentation the presenter was standing behind the laptop, controlling the presentation with the mouse, which caused very little movement. During the gesture based presentation, the presenter was not walking around more than in the traditional presentation, but in order to control the slideshow it was necessary to move the arms more, which might have confused test subjects. Furthermore, the presenter was required to move his hand around more noticeably when using the gesture based presentation compared to what would have been necessary if using for instance a laser pointer. Due to the pointing being mapped to the vector running from the presenter's elbow to his hand, it was necessary to move the entire arm, rather than just rotating the wrist in order to move the pointer around on the slideshow. The presenter's movement during the presentation seems to thus be the biggest drawback for the gesture based presentation, based on the perception of the audience. Even if the audience did not have a poorer understanding of the content in the gesture based presentation, it is still a significant problem for the gesture based interaction method that the method in itself seems distracting to the audience.

9.6 Test conclusion

The test did not prove that one form of presentation was better than the other. There was a slight preference amongst the test subjects towards the traditional presentation tool when asked directly about which presentation they preferred. On the other hand, there were elements that were preferred in the gesture based presentation tool compared to the traditional one. Based on the replies from the test subjects, some of the areas that need to be improved are the stability of the pointer that indicates where the presenter was pointing in the slideshow as well as fewer visual inputs. The movement of the pointer should either be averaged more to produce a more smooth movement or not be shown at all, such that audience would only see the objects or texts being highlighted. The visual indicator telling the presenter when he was changing modes was also confusing to some of test subjects and a way of showing this indicator to the presenter only would be a good solution. This could be done by collecting all information to the presenter on the laptop screen only. Even though that would require the presenter to look at the laptop screen more, it might be a better solution for the audience watching. However, if a reliable finger tracking could be combined with the skeleton tracking approach, the presenter could be confident in the system working as he expects and thus would not need to be shown as much information about the state of the presentation.

In general it seems that the problems that needs to be solved for a gesture based presentation tool is more related to working out the best way of utilizing the presenter's natural movements during the presentation that the technological limitations. For me personally – as the presenter in all test cases – it seems that it would be possible to relatively fast be able to learn the control methods of such a presentation tool. For most presentation purposes it is sufficient to be able to highlight objects and text as well as changing slides, which has been achieved by the implementation done in this project. The problem seems to be that the audience might be confused by the presenter when his body is actively controlling the slideshow and that the need for feedback to the presenter can confuse the audience. Based on the experiment conducted it seems that a presentation tool using gesture based interaction can indeed be as good as a traditional presentation tool and perhaps better, but that a better and more fluent way of implementing that natural gestures of the presenter needs to be developed.

10. Conclusion

The motivation for this project was to utilize the popular physical game systems of today in another context. Physical games are becoming increasingly popular and the technology for controlling them is becoming both cheap and easily accessible. Other research has been done in the area of gesture recognition and many studies have looked into the usage of gestures for different purposes. However, none of them had looked into how the audience would react to such gestures being used in a presentation context, which made it an interesting perspective for this project.

Throughout the pre-analysis it was investigated what gestures would be natural and meaningful in a presentation context. Especially deictic and metaphoric gestures proved to be suitable for use in presentations. Through investigations of these gestures as well as a survey conducted, it was decided that the deictic gesture of pointing and the metaphoric gesture of swiping to change slides would be the two main gestures to focus on during development. Furthermore, through investigations of the different game systems available it was chosen to use the Microsoft Kinect for development. The Nintendo Wii and Playstation Move are both limited by having to use controllers, where as the Kinect and regular webcams does not require the use of intrusive hardware. However, the Kinect has the further advantage of having a depth sensor and having open source libraries for development. The investigations of the pre-analysis lead to the following final problem statement:

How can I, using the Microsoft Kinect, create an application that will improve a slide show presentation towards the perceiving audience, using only a few, predefined, natural gestures?

In the analysis it was researched how to implement a reliable tracking method using the Kinect. With inspiration from especially (Malik, 2003) it was discovered how a finger tracking algorithm could be implemented through the use of the open source image processing library OpenCV. A skin color detector was described, which would be able to find skin colored areas based on the probabilities of a given pixel's color value being skin or non-skin. Feature extraction of fingertips would be done by finding narrow angles around the contours of skin areas. It was also investigated how a so called Haar-like feature approach could be used for face detection, in order to eliminate the face area before doing the detection of fingertips.

The application was designed to be able to track the presenter's hand and use the pointing and swiping for controlling a slideshow. The setup would be that the Kinect would track the presenter from the audience's viewpoint in order to get the best view of the presenter's movement as well as making it easier to do pointing projection. It was decided to have two different modes for the presentation tool: One for pointing and one for changing slides. In this way an accidental swipe gesture would not be changing slides. The pointing was designed to happen through projection a vector from the Kinect to the projected slideshow through the fingertip of the presenter's hand and the swipe gesture would be detected through the velocity of the hand along the x-axis.

Using the fingertip tracking approach it was possible to track a hand based on skin color and find fingertips and dips under ideal light conditions. However, the algorithm was very light dependent and also had issues with latency, which made it unsuitable for use in an actual experiment of a gesture based presentation tool. Therefore another approach was taken, using a skeleton tracking open source library from the organization OpenNI. Using this library it was possible to get data about the position and orientation of the presenter's joints as it is used in the games produced for the Kinect on Xbox. The skeleton tracking does not supply finger tip tracking though, so it lead to small changes in the design. The pointing would be done through calculating a vector going from the elbow joint through the presenter's hand and onto the slideshow. The implementation of the swipe remained the same though and was done by checking the velocity of the centre of the hand. The skeleton tracking approach was implemented using the game engine Unity, which made data transfer between different applications unnecessary. This, combined with the well optimized library of OpenNI, supplied for an application that was both fast and accurate in tracking the user and could thus be utilized in the final experiment of the presentation tool.

The final experiment was conducted on a total of 20 test persons, being tested on groups of five people, who were watching two presentations with fictional information. The test compared a traditional presentation (using Mircosoft PowerPoint) to the gesture based presentation in order to figure out which tool was preferred by the user. The test results showed that there was no significant preference towards one presentation over the other, with 55% preferring the traditional presentation tool, 35% preferring the gesture based presentation and the remaining 10% not preferring either of them. The traditional presentation tool seemed to have its advantages in the presenter's movement being less confusing to the user, while the gesture based presentation seemed to more clearly be able to highlight objects and text. The gesture based presentation tool was significantly more confusing to the audience watching than the traditional presentation tool. The presentation method did not seem to have any influence on the test subjects' ability to understand the content and the presentations did not become more interesting to the audience, using one interaction method compared to the other.

The application produced in this project did succeed in making a presentation tool, using the Microsoft Kinect, which implemented a few, predefined, natural gestures, but in its current state it is not significantly improving the way of presenting slideshow presentations, as perceived by the audience.

11. Future work

For further improving the gesture based presentation tool developed in this project, the first step would be to combine the two methods described in this chapter. Combining the accuracy of the skeleton tracking with a reliable finger tracking approach will supply for more options of tracking the presenter. The latency of the finger tracking approach might be reduced noticeably if only a small sub-image was analyzed, based on the position of the hand as tracked by the skeleton tracking approach. One of the main problems, which were also noted by the test subjects, was the need for visual feedback to the presenter about the state of the presentation. Given a reliable finger tracking – that the presenter could trust in – the need for visual feedback to the presenter would decrease and hereby also distract the audience less.

A gesture based presentation tool can profit from some of the elements implemented in this project, such as the highlighting. However, the movement of the presenter should be carefully considered during future implementations. A swipe gesture does seem to be a natural and meaningful action for changing slides, but it should be implemented in a way that does not distract the audience. The implementation used in this project where the presenter should change mode in order to change slides seems to be too distracting.

Finally it should be noted that a gesture based presentation tool is unlikely to make presentations more interesting or easier to understand, according to the experiment conducted in this project. Thus, future implementations should be done with the goal of improving the interaction and ease-of-use for the presenter as well as improving the experience for the audience.

12. Bibliography

Argyros, A. A., & Lourakis, M. I. (2006, May). Vision-Based Interpretation of Hand Gestures for Remote Control of a Computer Mouse. *Proceedings of the HCI'06 workshop (in conjunction with ECCV'06)* , pp. 40-51.

Baudel, T., & Beaudouin-Lafon, M. (1993, July). CHARADE: Remote Control of Objects using Free-Hand Gestures. *Communications of the ACM* (36), pp. 28-35.

Cassell, J. (1998). A Framework for Gesture Generation and Interpretation. (R. Cipolla, & A. Pentland, Eds.) *Computer Vision in Human-Machine Interaction* , pp. 191-215.

Efford, N. (2000). *Digital Image Processing - A practical introduction using Java.* Harlow, Essex, England: Pearson Education Limited.

Eisenstein, J., & Davis, R. (2004, October). Visual and Linguistic Information in Gesture Classification. *Proceedings of International Conference on Multimodal Interfaces (ICMI'04)* , pp. 113-120.

Elmezain, M., Al-Hamadi, A., Appenrodt, J., & Michaelis, B. (2008). A Hidden Markov Model-Based Continuous Gesture Recognition System for Hand Motion Trajectory. *Proceedings of Pattern Recognition, 2008. ICPR 2008.* , pp. 1-4.

Evoluce. (2010, November 19). *Plug Kinect in a PC an now interact with Windows 7 applications through gestures*. Retrieved February 15, 2011, from Evoluce: http://www.evoluce.com/en/company/pressreleases.php?we_objectID=28

Freeman, W. T., & Roth, M. (1994, June). Orientation Histograms for Hand Gesture Recognition. *Proceedings of International Workshop on Automatic Face and Gesture Recognition* , pp. 296-301.

Freeman, W. T., & Weissman, C. D. (1995). Television Control by Hand Gestures. *Proceedings of IEEE International Workshop on Automatic Face Recognition* , pp. 179-183.

Harvard Graphics. (2011). *Our Company*. Retrieved February 18, 2011, from Harvard Graphics: http://www.harvardgraphics.com/about.asp

Hewitt, J. (2008, October 29). *A Brief History of Microsoft Powerpoint*. Retrieved February 18, 2011, from Bright Hub: http://www.brighthub.com/office/collaboration/articles/13189.aspx

Husz, Z. (2004). Vision Based Human-Computer Interaction with Hand Gestures. *Proceedings of Automatic Control and Computer Science* (49), pp. 227-232.

Ivanov, Y., Bobick, A., & Liu, J. (2000, June). Fast Lighting Independent Background Subtraction. *International Journal of Computer Vision* , pp. 199-207.

Jones, M. J., & Rehg, J. M. (1999). Statistical color models with application to skin detection. *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* , pp. 274-280.

Kuchera, B. (2011, February 11). *PlayStation Move creator wants to open the device to PC users*. Retrieved March 12, 2011, from ars technica: http://arstechnica.com/gaming/news/2011/02/interview-sonys-dr-marks-wants-to-open-the-move-for-pc-users.ars?utm_source=rss&utm_medium=rss&utm_campaign=rss

Lenman, S., Bretzner, L., & Thuresson, B. (2002). *Computer Vision Based Hand Gesture Interfaces for Human-Computer Interaction.* Stockholm, Sweden: Centre for User Oriented IT Design - Royal Institute of Technology.

Malik, S. (2003). *Real-Time Hand Tracking and Finger Tracking for Interaction.* Toronto: University of Toronto.

McElroy, J. (2010). *PrimeSense 3D-sensing tech licensed for Project Natal [update]*. Retrieved May 13, 2011, from Joystiq: http://www.joystiq.com/2010/03/31/primesense-3d-sensing-technology-licensed-for-project-natal/

NIST/SEMATECH. (2011). *Upper Critical Values of the Student's-t Distribution*. Retrieved May 10, 2011, from e-Handbook of Statistical Methods: http://www.itl.nist.gov/div898/handbook/eda/section3/eda3672.htm

OpenCV. (2011). *Welcome*. Retrieved February 15, 2011, from OpenCV: http://opencv.willowgarage.com/wiki/Welcome

OpenFrameworks. (2011). *About*. Retrieved March 12, 2011, from OpenFrameworks: http://www.openframeworks.cc/about

OpenKinect. (2011). *Main Page*. Retrieved February 15, 2011, from OpenKinect: http://openkinect.org/wiki/Main_Page

OpenNI. (2011). *About*. Retrieved May 13, 2011, from OpenNI: http://www.openni.org/about

Pankinkis, T. (2010, June 29). *Kinect technical specifications revealed*. Retrieved March 2, 2011, from CVG: http://www.computerandvideogames.com/253794/news/kinect-technical-specifications-revealed/

Perzanowski, D., Schultz, A. C., Adams, W., Marsh, E., & Bugajska, M. (2001, February). Building a Multimodal Human-Robot Interface. *Proceedings of Intelligent Systems, IEEE* , pp. 16-21.

Pottar, I., Sethi, Y., Ozyildiz, E., & Sharma, R. (1998). Toward Natural Gesture/Speech HCI: A Case Study of Weather Narration. *Proceedings of PUI* , pp. 1-6.

PrimeSense. (2010). *Product Technology*. Retrieved May 13, 2011, from PrimeSense Ltd.: http://www.primesense.com/?p=487

Scantleberry, C. (2003). *Tech: Eye Toy Review*. Retrieved March 2, 2011, from The Next Level: http://www.the-nextlevel.com/features/eye_toy/eye-toy.shtml

Schlömer, T., Poppinga, B., Henze, N., & Boll, S. (2008, February). Gesture recognition with a Wii controller. *Proceedings of the 2nd international conference on Tangible and embedded interaction* , pp. 11-14.

Sharp, H., Rogers, Y., & Preece, J. (2007). *Interaction Design: Beyond Human-Computer Interaction* (Second edition ed.). Chichester, West Sussex, England: John Wiley & Sons Ltd.

Spielberg, S. (Director). (2002). *Minority Report* [Motion Picture].

Sukthankar, R., Stockton, R. G., & Mullin, M. D. (2001). Smarter Presentations: Exploiting Homography in Camera-Projector Systems. *Proceedings of International Conference on Computer Vision* , pp. 247-253.

Unity3D. (2011). Retrieved May 2, 2011, from UNITY: Game Development Tool: http://unity3d.com/unity/

Viola, P., & Jones, M. (2001, December). Rapid Object Detection using a Boosted Cascade of Simple Features. *Proceedings of the 2001 IEEE Computer Society Conference on Computer Vision and Pattern Recognition* , pp. I-511 - I-518.

Wexelblat, A. (1995, September). An Approach to Natural Gesture in Virtual Environments. *ACM Transactions on Computer-Human Interaction, Vol. 2, No. 3* , pp. 179-200.

Wicii. (2010). *About*. Retrieved February 15, 2011, from Wii Controller Interface Suit: http://wiici.dyadica.net/about

Wilson, A., & Bobick, A. (1999, September). Parametric hidden Markov models for gesture recognition. *Proceedings from Pattern Analysis and Machine Intelligence, IEEE Transactions* (9), pp. 884-900.

12.1 Illustrations

All illustrations and tables not listed below are created by the author of the report.

Illustration 7.1: Retreived May 10, 2011 from http://www.kinecteducation.com/ . Direct image URL: http://www.kinecteducation.com/blog/wp-content/uploads/2011/05/Screen-shot-2011-05-19-at-5.21.32-PM.png

Illustration 7.2: Screenshot from Unity (Unity3D, 2011).

Illustration 8.3: Screenshot from Unity (Unity3D, 2011).

13. Appendix

13.1 Survey about use of digital presentation tools

In correlation with my master thesis on gesture based interaction, I would like you to answer this short survey about your usage of digital presenta tion tools. The survey should only take a few minutes to complete, so I hope you will take the time to fill it out.

Thanks in advance Heino Jørgensen, Medialogy, AAU København.

What is your age?
<Drop-down list, containing age ranges: 0-17, 18-25, 26-35, 36-45, 46-55, 56-65, 66+>

What is your occupation?
<Text field for writing answer>

How often do you use presentation tools (e.g. Microsoft Powerpoint, Prezi, Keynote)?
<Multiple choice between: Weekly, Monthly, Every 2-4 months, Every 5-8 months, Rarer, Never>

What tool do you use?
<Check list with choices: Microsoft PowerPoint, Apple Keynote, Prezi, Zoho, Other…>

For what purpose do you use presentations?
<Check list with choices: Teaching, Presentation of research, Presentation of products, Exams, Other…>

How do you control your presentations?
<Multiple choice between: Using keyboard/mouse, Using a remote control, Using an automated slideshow (timed change of slides), Other…>

What functionalities do you use in your presentations?
<Check list with choices: Change slides (back and forth), Jump between slides, Highlight text, Highlight graphs/objects, Play videos, Other…>

Describe briefly what gesture you would use to change slide forward.
<Text field for writing answer>

Describe briefly what gesture you would use to change slide backward.
<Text field for writing answer>

Describe briefly how you would highlight something in a slideshow presentation, using gestures.
<Text field for writing answer>

Are there any other slideshow related functions that you feel could be controlled by gestures?
<Text field for writing answer>

13.2 Final test questionnaire

Questionnaire about understanding

AMERICAN BOOK SALES DECLINING

Give a brief description of what you were just presented to.
<Text field for writing answer>

Were there any pictures in the presentation? If yes, what did they show?
<Text field for writing answer>

What was the unit of the y-axis of the graph on slide 2?
<Multiple choice between: Copies sold in billions, Copies sold in millions, Copies sold in thousands, Copies sold in hundreds>

What did the blue bars in the graph show?
<Multiple choice between: Amount of paper books sold, Amount of digital books sold, Amount of camper vans sold>

Name (at least) one reason, given in the presentation, as to why there is decline in the sale of paper books?
<Text field for writing answer>

BACTERIA GROWTH IN DAIRY PRODUCTS

Give a brief description of what you were just presented to.
<Text field for writing answer>

Where there any pictures in the presentation? If yes, what did they show?
<Text field for writing answer>

What was the timescale of the graph on slide 2?
<Multiple choice between: 6 months, 6 weeks, 6 days, 6 hours>

What did the red bars in the graph show?
<Multiple choice between: Bacteria in milk, bacteria in cheese, bacteria in meat>

Name (at least) one reason, given in the presentation, as to why cheese develop bacteria slower than milk?
<Text field for writing answer>

Questionnaire about opinion

Of the two presentations you have just seen, which did you prefer?
<Multiple choice between: The first one, the second one, neither of them>

On a scale ranging from 1-5, how interesting would you rate the information given in the first presentation? (5 being the very interesting)
<Scale ranging from 1-5>

On a scale ranging from 1-5, how interesting would you rate the information given in the second presentation? (5 being the very interesting)
<Scale ranging from 1-5>

On a scale ranging from 1-5, how difficult did you find it to understand the content of the first presentation? (5 being very easy)
<Scale ranging from 1-5>

On a scale ranging from 1-5, how difficult did you find it to understand the content of the second presentation? (5 being very easy)
<Scale ranging from 1-5>

On a scale ranging from 1-5, how did you perceive the presenters movement in the first presentation? (5 being very distracting and 1 being not distracting at all)
<Scale ranging from 1-5>

On a scale ranging from 1-5, how did you perceive the presenters movement in the second presentation? (5 being very distracting and 1 being not distracting at all)
<Scale ranging from 1-5>

One a scale from one 1-5, how would you rate the change of slides in the first presentation? (5 being very effective)
<Scale ranging from 1-5>

One a scale from one 1-5, how would you rate the change of slides in the second presentation? (5 being very effective)
<Scale ranging from 1-5>

One a scale from one 1-5, how would you rate the highlighting of objects in the first presentation? (5 being very effective)
<Scale ranging from 1-5>

One a scale from one 1-5, how would you rate the highlighting of objects in the second presentation? (5 being very effective)
<Scale ranging from 1-5>

One a scale from one 1-5, how would you rate the highlighting of text in the first presentation? (5 being very effective)
<Scale ranging from 1-5>

One a scale from one 1-5, how would you rate the highlighting of text in the second presentation? (5 being very effective)
<Scale ranging from 1-5>

Comments:
<Text field for writing feedback>

By signing this form you agree that the information given in this questionnaire can be used in the project work of my master thesis. No information will be given to third parties and your replies are being used only for the purpose of my project work.

Thank you for your cooperation. Heino Jørgensen, MED10

13.3 Qualitative test data

TP1:

- I figure out/ most people should figure out what's the objective of the test. When I received 1st questionnaire after the first presentation.
  This test is full of uncertainty. I tend to remember more info at the 2nd presentation, because I believe I might answer these questions.

TP3:

- I think you need a way to remove the blue dot. It was jumping to much and its size made me focus on that instead of the text. The gesture thing is cool but I can't remember anything from second presentation.

TP6:

- The "shaking" of the dot is really disturbing. Also that the cursor has to be moved to the corner seemed to be a little unnatural.
- The questionnaire: To rate the interest of the presentation might be really depending on the interest of the person.

TP7:

- The blue cursor dot used in the first presentation was too big and opaque.

TP8:

- Apart from the movements when changing slides, I found the visuals on top of the slides also distracting. (The visuals which appeared when changing slides).
- The highlighting was definitely better in the first one.

TP9:

- I found the blue dot's shaking a little annoying.
- Other than that the first presentation better.

TP10:

- I guess you know it already but the dot has to be way more precise and also a lot more non flickering. It is very confusing with a dot moving all the time.

TP11:

- A shame the dots movements were distracting the eye from the text and not making the highlights as effective as they could.

TP12:

- There might be a problem in the test with getting a questionnaire in the middle. In the second test you are much more aware of what to look for.

TP13:

- We are slightly biased because of the first presentation – you know you are supposed to remember stuff!

TP14:

- Liked the second presentation, but were a little confusing to look at the blue dot shaking all the time.

TP16:

- The blue circle was extremely annoying.
- The highlighting of the bars was ok.
- The highlighting of the text was annoying (all comments for presentation #1).

TP17:

- Second presentation was not given as close to natural as you normally would. Is was like it was attempted to over emphasize how the current use of slideshows can be cumbersome.

TP19:

- I found the blue dot and your pointing finger very disturbing during presentation.