# Table of Contents

List of Abbreviations ........................................................................................................ iv
List of Figures .................................................................................................................. v
Abstract ............................................................................................................................ vi
1 Introduction .................................................................................................................... 1
  1.1 Target Audience ........................................................................................................ 1
2 State-of-the-Art ............................................................................................................... 2
  2.1 Motion Capture .......................................................................................................... 2
   2.1.1 Marker-less Motion Capture ............................................................................. 2
  2.2 Gesture Recognition Software .................................................................................. 4
   2.2.1 lisu ..................................................................................................................... 4
   2.2.2 FUBI - Full Body Interaction Framework ....................................................... 5
  2.3 Game Engines .......................................................................................................... 6
   2.3.1 Unity3D ............................................................................................................ 6
   2.3.2 Unreal Development Kit ..................................................................................... 7
  2.4 Virtual Agents .......................................................................................................... 7
   2.4.1 Representation ..................................................................................................... 7
   2.4.2 Interaction .......................................................................................................... 8
  2.5 Screens ...................................................................................................................... 9
   2.5.1 Flat Panel Displays .......................................................................................... 9
   2.5.2 3D display technology ...................................................................................... 9
   2.5.3 Projectors .......................................................................................................... 11
   2.5.4 Conclusion ......................................................................................................... 11
3 Problem Definition ....................................................................................................... 12
4 Design Method .............................................................................................................. 13
  4.1 Creating Scenarios ................................................................................................. 13
  4.2 Video Sketching ....................................................................................................... 13
5 System Design .............................................................................................................. 16
  5.1 Requirements .......................................................................................................... 16
  5.2 Environment ............................................................................................................ 16
  5.3 Mobility .................................................................................................................... 18
6 Implementation ............................................................................................................. 19
  6.1 UDK Environment Design ....................................................................................... 19
  6.2 FUBI ......................................................................................................................... 23
List of Abbreviations

API – Application Programming Interface

DLL – Dynamic Link Library

FUBI – Full Body Interaction (Framework)

IK – Inverse Kinematics

IR – Infrared

IMS – Imagination Studios

LED – Light Emitting Diode

MoCap – Motion Capture

VMT – Virtual Multi Tool

UDK – Unreal Development Kit

VA – Virtual Agent
List of Figures

Figure 1: SwissRanger™ SR4000 ...................................................... 2
Figure 2: Microsoft Kinect Sensor .................................................. 3
Figure 3: Measuring respiratory rate with the Kinect (Burba, et al. 2012) .................... 3
Figure 4: Natural interaction with culturally adaptive virtual characters (Kistler, Endrass, et al. 2012) 4
Figure 5: Towards a Virtual Environment for Capturing Behavior in Cultural Crowds (Lala, Thovuttikul and Nishida 2011) ................................................................. 4
Figure 6: Full Body Gestures enhancing a Game Book for Interactive Story Telling (Kistler, Sollfrank, et al. 2011) ........................................................................................................ 5
Figure 7: Show Some Respect! (Johnsen, et al. 2010) ................................................... 8
Figure 8: Virtual Multi-Tools for Hand and Tool-Based Interaction with Life-Size Virtual Human Agents (Kotranza, et al. 2009) ........................................................... 8
Figure 9: Parallax barrier (Urey, et al. 2011) ................................................................ 10
Figure 10: Lenticular array (Urey, et al. 2011) .............................................................. 10
Figure 11: Video sketch scene 1 ................................................................................. 14
Figure 12: Video sketch scene 2 ................................................................................. 14
Figure 13: Video sketch scene 3 ................................................................................... 14
Figure 14: Video sketch scene 4 ................................................................................... 14
Figure 15: Video sketch scene 5 ................................................................................... 14
Figure 16: Video sketch scene 6 ................................................................................... 14
Figure 17: Video sketch scene 7 ................................................................................... 15
Figure 18: IMS MoCap studio entrance angle ............................................................... 17
Figure 19: IMS MoCap studio inside angle .................................................................... 17
Figure 20: UDK scene 1 ......................................................................................... 20
Figure 21: UDK scene 2 ......................................................................................... 20
Figure 22: UDK scene 3 ......................................................................................... 21
Figure 23: UDK scene 4 ......................................................................................... 22
Figure 24: UDK scene 5 ......................................................................................... 22
Figure 25: Master’s student that was the MoCap actor ................................................. 25
Abstract

This thesis was carried out at Imagination Studios in Uppsala. IMS is a motion capture studio that also does animation. Motion capture is the capturing of (generally) human motions to make 3D animations look more realistic. In motion capture, the actors have to imagine the scene. The goal of this thesis is to help the motion capture actor by creating a tool that lets the actor interact with a virtual agent that represents his acting partner. Scenarios and a video sketch were created to describe how the interaction can work. The Microsoft Kinect is used to capture the motions of the actor and recognize gestures. These gestures are then responded to by a virtual agent that is displayed in a 3D environment created in the Unreal Development Kit. Programming was done in C++ and UnrealScript to make this solution work. Motions were recorded and applied to the virtual agent to create realistic animations that are played in response to the actor's gestures. The final product is an interactive application that can be used to immerse a person in an acting scenario.
1 Introduction

This report describes a thesis in computer science that was carried out at Imagination Studios (IMS). IMS is a motion capture studio that performs motion capture and animation for its clients. Motion capture is the method of capturing and recording an actor’s motions. These recorded motions are then used for creating realistic animations. Imagination Studios’ clients are mainly game developers that need animations and cut scenes. Cut scenes are animation sequences that the player generally does not interact with. It is used to tell a story and immerse the player. Some of the notable games that IMS have worked on are Battlefield 3, Alan Wake and Bulletstorm. IMS’ research and development department is looking at ways to help the motion capture actors with their performance. This thesis was created by IMS’ R&D department as part of a bigger project to improve motion capture by helping the actor.

1.1 Target Audience

The product is designed to be used by motion capture studios to aid and improve the performance of motion capture actors. It is important to note that this project is not intended to be a final solution. It’s a start to a series of projects to improve motion capture actors’ performance. Parts of this product will be used in other projects at IMS.
2 State-of-the-Art

2.1 Motion Capture

Motion capture is the process of capturing motions of one or more actors. The standard practice is to use active or passive markers as reference points on the body. Passive markers are reflective rubber balls placed on the person or object that is to be captured. A series of cameras shine infrared light to triangulate the position of the markers. A downside of using passive markers is that you need to eliminate any other reflective surface in the shoot area. Passive markers are not very expensive and are easy to attach to the object or person being captured. Active markers contain an infrared LED which emits light that the camera can register. Each active marker has a unique ID, which makes the data more clear. Active markers can be used in natural light and is not restricted to a dark studio (Maletsky, et al. 2007).

The main use of motion capture is in game development, but it is used in several different areas. Video and TV, film, scientific research and education are some areas that employ motion capture (MetaMotion n.d.). In games, there are two general areas that motion capture can be used for. Real-time playback is when the player in some way chooses when the motion is played. For example when the player moves his character forward, a walking motion is played and displayed on the 3D character. The other area is in-game cinematics. These cinematics are often pre-rendered animations which are used to tell the story of the game and immerse the player. The cinematics are usually not interactive and are therefore plays out like a part of a movie.

2.1.1 Marker-less Motion Capture

There are several kinds of marker-less MoCap: inertial MoCap, magnetic MoCap and optical MoCap. As the environment for this project warranted a device-free system, optical MoCap is focused on in this report. MoCap that does not use any kind of markers is sought after in the industry because it would make the process of motion capture require less setup. A lot of research is being done in this area and new technologies arise frequently. The technologies that exist for optical marker-less MoCap today are not able to reach the same accuracy as marker-based systems, but they are coming close.

2.1.1.1 SwissRanger™ SR4000

The SwissRanger SR4000 (MESA Imaging n.d.) is a 3D time-of-flight camera. It performs optical marker-less MoCap at 50 frames per second. There are two general versions of the camera, one that is optimal at a range of 5 meters and another for 10 meters range. The price of the camera is 28800 SEK (October 11, 2012), which makes it a very expensive piece of hardware for a thesis project. The camera is successfully used in the IMoRa project to communicate with the humanoid virtual agent Vince (Sadeghipour, et al. 2011). When Vince does not recognize the gesture that the user is performing, he will learn it and ask for a label to associate with the gesture. The gesture recognition middleware isu (SoftKinetic n.d.) is used in combination with the SR4000 camera to recognize the gestures performed by the user.
2.1.1.2 Kinect

A common marker-less system is the Microsoft Kinect. The Kinect was released November 4, 2010 in America. The Kinect was released as an accessory to Xbox 360. Microsoft has shipped 18 million units as of January, 2012 (Takahashi 2012). The price of the Kinect is around 1000 SEK (October 10, 2012). The Kinect utilizes a RGB camera, and a depth-sensor with an IR light source (Pheatt, et al. 2012). This allows the Kinect to output three dimensional position data in real time. The Kinect also has four microphones forming a microphone array that is used to recognize voice commands. The precision of the Kinect does not come close to motion capture with markers but it is a cheap alternative that can be a good choice when the data does not have to be extremely accurate. There are several research projects that use the Kinect which proves it is useful beyond game development.

The Kinect was chosen for this project for several reasons. There is a wide variety of software libraries to choose from that use the Kinect. There is a huge amount of information online about the Kinect, how it works and how to use it. The price of 1000 SEK is affordable and IMS already had a few Kinects that could be used.

2.1.1.3 Kinect Applications

Burba et al. use the Kinect in order to measure subtle non-verbal behavior (Burba, et al. 2012). The Kinect is used to measure respiratory rate of a person sitting down. They also use the Kinect to measure “leg fidgeting”, which is the motion of rapidly jumping either leg up and down. This motion is usually associated with nervousness and can therefore be measured to help determine the emotional state of the person using the system. The paper mentions problems with tracking when trying to measure the respiratory rate of a person standing up. This project proves that the Kinect can be used to measure subtle movements, but precision is limited when the person being tracked is moving.

The project “Natural interaction with culturally adaptive virtual characters” (Kistler, Endrass, et al. 2012) use the Kinect to research cultural differences. They focus on the proximity at which people are comfortable when talking to other virtual agents. The Kinect allows the user to have full body control of a virtual agent in third person view. Testing showed that the users liked the intuitive control method that the Kinect allowed. The authors of this paper are also responsible for creating...
the FUBI (Full Body Interaction) framework (Kistler, Augsburg University 2012). The FUBI framework is written in C++ and uses OpenNI (OpenNI n.d.) and the middleware NITE (PrimeSense n.d.). FUBI allows gestures to be defined in XML or C++ code which can then be recognized by the Kinect.

There are several projects that explore cultural differences with the help of the tracking offered by the Kinect. Another project which also focuses on cultural differences concerning proximity is called “Towards a Virtual Environment for Capturing Behavior in Cultural Crowds” (Lala, et al. 2011). This project has a large focus on the cultural behavior of the virtual agents towards each other and the user. Each virtual agent has social parameters which instruct it how to react when interacting with other agents. These parameters can be how big the personal space is for the virtual agent. The virtual agent will try its best to keep its personal space clear. Eight screens were set up around the user to create an immersive environment. The game engine jMonkeyEngine (jMonkeyEngine n.d.) was used to create the 3D environment.

2.2 Gesture Recognition Software

When it was clear that the Kinect was going to be used, software to make it work had to be decided upon. There are two routes that can be taken when choosing how to use the Kinect for gesture recognition. Either only choose a foundation like OpenNI or Microsoft Kinect SDK (Microsoft n.d.), or choose a framework that expands on the foundation, making it easier to define gestures. For this thesis project, a framework that expanded on a foundation was chosen. The reason for this is that it eliminates a lot of unnecessary extra work.

2.2.1 iisu
lisu is a gesture recognition middleware that is developed by the company SoftKinetic. It most notably provides skeleton tracking and a gesture library. lisu supports C/C++ and C#. It has integrated plug-ins for flash and Unity3d. lisu is free and fully featured for three months if it is used non-commercially. After the three months, some of the features are limited or removed. Below is a list of the differences and limitations of the free version compared to the professional version.

- The free version can only actively track one person. The pro version can track up to four people simultaneously.
- The lisu interaction designer, which is “a tool for technical designers to easily prototype gesture interactions” is only available for 3 months in the free version.
- The lisu toolbox, which “Provides access to live data and performance analytics during development.” is only available for 3 months in the free version.

The commercial license for lisu can be bought for a one-time fee of $1500 USD. lisu has been used in several successful projects. Some of the most notable are Disney’s “The Sorcerer’s Apprentice” (Inwindow Outdoor n.d.), which is a 3D interactive outdoor advertisement and GURU Training Systems Swinguru (Swinguru n.d.) which analyzes a golfer’s swing movement.

lisu could be a good choice for this project. It has been used in several successful projects. It can be used with the Kinect but it is meant to be used with one of SoftKinetics 3D cameras. The fact that it isn’t specifically meant to be used with the Kinect and that the free version is limited warranted looking for other options.

2.2.2 FUBI - Full Body Interaction Framework

The FUBI framework is able to recognize full body gestures using the data provided by a depth sensor that can communicate with OpenNI. The framework is developed in C++ and offers a C++ API, as well as the ability to define gestures in XML. There is a C# wrapper that allows use of the FUBI API in C# as well. FUBI requires OpenNI and NITE to work. If the Kinect is the depth sensor being used with FUBI, the avin2 Kinect driver (github 2012) has to be installed. Fubi is available for free under the terms of the Eclipse Public License. There are a few projects that use FUBI. The previously mentioned “Natural interaction with culturally adaptive virtual characters” (Kistler, Endrass, et al. 2012) is one of those projects. The project “Full Body Gestures enhancing a Game Book for Interactive Story Telling” (Kistler, Sollfrank, et al. 2011) uses FUBI. It requires the users to do a number of so-called Quick Time Events (QTEs) which are different gestures and body movements. The FUBI framework distinguishes between four gesture categories:

1. **Static postures**: Configuration of several joints, no movement (e.g. figure 1: “arms crossed”).
2. **Gestures with linear movement**: Linear movement of several joints with specific direction and speed (e.g. figure 2: "right hand moves right").

3. **Combination of postures and linear movement**: Combination of sets of 1 and 2 with specific time constraints (e.g. figure 3: "waving right hand").

4. **Complex gestures**: Detailed observation of one (or more) joints over a certain amount of time and recognition of specific patterns/paths (e.g. symbolic gestures like hand writing shapes).

FUBI was chosen for this project as it seemed like a good, free alternative. Quickly after trying it out, it was apparent that defining gestures in XML was a quick and easy way to prototype and test gestures. These gestures could later be implemented in C++ to allow for more control over the code. FUBI works with either OpenNI or the Microsoft Kinect SDK. OpenNI was chosen because of being an open source API and OpenNI had been used by a previous thesis student at IMS. This decision was also based on the comparison of the two APIs in his thesis.

### 2.3 Game Engines

To display the virtual agent in a realistic manner, a virtual environment had to be created. The tool to create and display a virtual environment had to be able to display the virtual agent with animations as well. All of these requirements are met by game engines. Level design is an important part of game development and is supported by many game engines. Displaying 3D models and playing animations is an essential part of a game engine as well. Suitable game engines were investigated and ultimately one was chosen for this project. Unity3D and UDK are the two game engines that were researched for this project because they are two of the prominent game engines on the market. They both offer free licenses and have been used for successful game products. The comparison will be presented in a bullet point format accompanied by text.

#### 2.3.1 Unity3D

- **Platform support**: Web player, Adobe Flash, iOS, Android, PC, Mac, Nintendo Wii, Playstation 3 and Xbox 360
- **Programming language support**: JavaScript, C# or the Python dialect Boo.
- **Built-in NVIDIA PhysX Engine** (hardware accelerated physics)

Unity3D is a popular game engine, used by games such as Battlestar Galactica Online and Tiger Woods PGA Tour Online. There is a free version available that is limited in some important areas. The free version of Unity does not support IK (Inverse Kinematics) rigs. The motion data that IMS records are mapped to a 3D character with an IK rig bound to it. To display animations on the virtual agent, support for IK rigs is needed. The free version does not support 3D textures, realtime shadows, native code plugins and other relevant features. The native code plugin is needed if we are to use some kind of C++ library as FUBI for the gesture recognition. These limitations make the free version less interesting, and $1500 USD is expensive for a thesis project when there are free alternatives available.
2.3.2 Unreal Development Kit

- Platform support: Adobe Flash, iOS, PC, Mac, Playstation 3, Xbox 360, Nintendo Wii U, Playstation Vita
- Programming language support: Unrealscript (similar to C++ and Java)
- Build-in NVIDIA PhysX Engine (hardware accelerated physics)

UDK was a tool suggested by the thesis supervisor. It has been used in a previous thesis project at IMS. UDK is a game engine developed by Epic Games that has become increasingly popular. It has been used in several successful games. There is an impressive amount of successful games made with UDK. Some examples of games that are developed with UDK are the Gears of War Series, Mass Effect 3, Batman: Arkham City and some games that IMS has done MoCap for, like Bulletstorm and XCOM: Enemy Unknown. UDK is free and fully featured for noncommercial use. For commercial use, a $99 USD fee is required. When the company earns UDK related revenue over $50000 USD, a royalty of 25% has to be paid to Epic. This licensing model is very well suited for a thesis project. The noncommercial version of UDK has no limitations, IK rigs are supported, which is vital to this project. UDK is able to interface with native code through a Windows dll. This means that it can interface with other programs needed for gesture recognition, such as FUBI.

2.4 Virtual Agents

The majority of the research done early in the thesis process was related to virtual agents (VA). The acting partner in the projects virtual environment was to be represented by a VA. Extensive research concerning what others have done was conducted, how the agent is represented, how the VA is displayed and in what way the user can interact with the VA.

2.4.1 Representation

One big benefit of interacting with a virtual agent instead of a human being is that the VA can be represented by anything. Most projects researched in this thesis are using a humanoid representation for the virtual agent. This includes standard humans, as seen in (Kistler, Endrass, et al. 2012) as well as humanoid robots, such as Vince that was used as a VA in the IMoRa project (Sadeghipour, et al. 2011). The method of displaying the VA to the user varies as well. Most of the projects researched uses a setup with displays such as standard television screen. The paper “Show Some Respect! The Impact of Technological Factors on the Treatment of Virtual Humans in Conversational Training Systems” (Johnsen, et al. 2010) tests how a humanoid VA is received depending on the type of screen that is used to display the VA. Tests are done with a typical 22” LCD monitor and a 42” plasma TV. The plasma TV displays the VA in life-size scale relative to the user. The test scenario was that the VA, called VIC (Virtual Interactive Character) was a 35 year old patient that had pain in his abdominal region. The user that interacts with VIC is a doctor that has to perform a patient interview. The tests were done in a WoZ (Wizard of Oz) manner. A WoZ operator controls all the actions of VIC, he is not controlled by an artificial intelligence. The tests show that when the vertically placed 42” plasma TV is used to display VIC, he is given more respect and the participants in the test were more engaged, empathic, pleasant and natural. When the 22” LCD monitor was used, the participants appeared disconnected from the social interaction. The test results in this paper were a motivation to aim for a close to life-size VA in the final product.
2.4.2 Interaction

The optimal solution for this project would be using the motion capture data to determine gestures. MoCap data is very precise; therefore the system would be very robust when detecting gestures. It was made clear early on in the thesis process that a device-free solution was required. Motion capture actors are wearing suits with many markers on them; this means that devices on the user had to be kept to a minimum. Since there isn’t a live feed of the motion data available at IMS, some other solution to recognize gestures was required. Use of the Kinect sensor was recommended for the thesis, but any solution that kept devices on the user to a minimum could be tested with enough motivation. Research into what interaction methods were used for other VA interaction projects was conducted.

The method of interaction between the user and the VA is a vital part of immersing the user in a virtual environment. The interaction method should be natural and effective. If the interaction method is too error-prone, users are easily dissatisfied (Kotranza, et al. 2009) which in turn can easily break immersion. In the paper “Virtual Multi-Tools for Hand and Tool-Based Interaction with Life-Size Virtual Human Agents” (Kotranza, et al. 2009) a Nintendo Wii remote is used as a “virtual multi-tool” to interact with a virtual patient. The remote is used to perform an eye examination on a patient. The VMT is used to simulate a hand and an ophthalmoscope that is used to examine the inside of the patient’s eye. The buttons on the remote are used to switch between tools, turning the ophthalmoscope light on or off, and changing the number of fingers held up for the patient to count. It is mentioned that this method was very effective at minimizing the errors that is usually associated with interaction methods such as speech recognition. The VMT method is not very suitable for this thesis since it involves using a device. MoCap actors often have to handle props (e.g. guns) that are tracked which would interfere.
with this method. This specific method with the Nintendo Wii remote would be limited to recognizing gestures that can be decided from the position of one hand (the remote).

Most of the projects that was researched used marker-less optical sensors like the Kinect. A device-free interaction method is sought after by many people. There are options to the Kinect, like the DepthSense 311 or the previously mentioned SwissRanger SR4000. These are either not significantly better than the Kinect or too expensive to justify purchase for a thesis project. IMS already had the Kinect hardware ready to be used, and since there is an immense amount of information and libraries available for the Kinect, it was chosen as the interaction method for this thesis.

2.5 Screens
It was clear that a life-size VA would be best at immersing the actor in the virtual environment. Different solutions for displaying the virtual environment and the VA were explored.

2.5.1 Flat Panel Displays
There are two major types of flat panel displays: liquid crystal displays and plasma displays. LCDs are made up of liquid crystals between glass plates. An electrical charge is applied to the crystals to create an image. Plasma displays are made up of tiny gas plasma cells that are charged by precise electrical voltages to create pictures.

2.5.1.1 LCD Plasma comparison

**LCD**
- Power efficient.
- Higher native resolution.
- Less bulky than plasma.

**Plasma**
- Better viewing angle than LCD.
- A lot cheaper than LCD when size is 50” and bigger.
- Generally more natural colors and deeper blacks.

The differences of LCD and plasma displays are fading when it comes to picture quality. They are both reaching similar quality. One of the major differences is cost. Displays over 50” are generally cheaper if it is a plasma display, rather than a LCD display.

2.5.2 3D display technology

2.5.2.1 Stereoscopic Direct-View Technologies
The market for stereoscopic displays capable of displaying 3D images has been increasing lately. These displays need the viewer to wear some kind of glasses. There are three types of stereoscopic direct-view technologies that are most common. These are color anaglyph, active shutter and passive polarization.

Color anaglyph technique achieves a stereoscopic 3D effect by using filters of different colors. Red and cyan are common colors used. 3D images that can be viewed with anaglyph glasses contain two
differently filtered colored images. Each image reaches one eye. The brain fuses these images into a three dimensional image.

Passive polarization technology displays two superimposed images on the same screen that have gone through polarizing filters. When used in projection, a silver screen is used to preserve the polarization. The glasses contain opposite polarizing filters and are passive, which means they have no need for a power supply. Each filter only passes light that is similarly polarized and blocks opposite polarized light. A benefit of this technique is that the glasses are inexpensive. When polarization displays are used in 2D mode, 50% of the light is lost to polarization components (Urey, et al. 2011).

The active shutter glasses needs electronics and power to work properly. The glasses contain a liquid crystal layer which becomes dark when voltage is applied. They work by blocking the view of the eyes separately in coordination with the display refresh rate. The glasses are quite expensive since they need electronics to work. It can be cumbersome that they require a power supply as well. Shutter-glass-based systems have a resolution advantage over polarized-based systems since polarization-based flat panel displays typically lose half of the spatial resolution to produce a stereo pair image (Urey, et al. 2011).

2.5.2.2 Autostereoscopic Technologies

Autostereoscopic 3D or “glasses-free” 3D is a technology that is evolving rapidly. The benefit of this technique is that it requires no headgear or glasses for the viewer.

One way to achieve this effect is by using what is called a parallax barrier. The parallax barrier allows the eyes of a user to see two separate images. The downside of this technology is that for the effect to work, the user has to be in a sweet spot. This sweet spot is at a certain range from the display and only works within a certain angle from the screen. The optimum viewing distance is proportional to the distance between the display and the parallax barrier (Urey, et al. 2011). Loss of brightness and spatial resolution are two problems with parallax barriers. Half of the pixels are used for each viewing zone, which causes the loss of spatial resolution.

Lenticular systems combine cylindrical lenses to flat panel displays to direct the diffused light from a pixel such that it can only be seen in a certain viewing angle in front of the display (Urey, et al. 2011). The lenticular-based display creates repeated viewing zones for the left and right eyes. Alignment of the lenticular array is difficult for higher resolution displays. Misalignment of the array can cause distortion in the displayed images.
2.5.3 Projectors

Projectors have the benefit of being more flexible when it comes to screen size. They require more setup than a display does however. The projector needs a clear path for the light to reach the screen. The standard projection setup is to have a projector in front of the screen with the light bouncing off the screen surface, to then be picked up by viewer’s eyes. This method is problematic if the screen is close to the ground. If a person is standing close to the screen, the light from the projector will be blocked. This can be solved by having a rear-projected screen.

When using rear-projection, the projector is placed behind the screen. This requires a screen surface that is able to catch the light successfully from behind with most of the light being visible from the other side of the screen. A downside to rear projection is that it requires a lot of space behind the screen. The space requirement can be reduced by using a mirror to reflect the projectors light to the back of the screen surface.

2.5.4 Conclusion

For this thesis, the actor needs to be able to move freely in front of the display. This means that the actor must be able to get close to the display if needed. Standard front projectors are a problem if the display will be close to the ground, which it should be since it will display life-size virtual agents that the actor can interact with. Rear projection poses the problem of space requirement. The motion capture environment requires a solution that leaves the actors much space to move around in. For these reasons, I believe that flat panel displays are the best solution, they do not require much space and they can be close to the ground. The Kinect sensor needs a power cable connected and will be close to the screen, therefore the screen will have a power connection as well. 3D is becoming more common and it can definitely help in immersing the viewer. The problem of having to wear glasses or not wearing glasses, but having a limited view zone is quite significant. Passive polarization seems to be a fitting technology, since the glasses are not bulky and polarized contact lenses are available. The 3D effect can be turned off on these displays, if for example face capture without glasses are needed in the motion capture scene. The polarized 3D displays offer the advantage of the immersive 3D effect while allowing the device-free environment of a LCD or plasma display if needed.
3 Problem Definition

The employees at IMS work in an old church. Motion capture requires a big empty room, which a church building provides. The motion capture room’s dimensions are 7x12x15 meters. There are a total of 38 motion capture cameras in the studio that can record the motions of over 10 actors simultaneously. The studio can also capture sound and the actors’ facial expressions, which a lot of animations require. One method of helping the actors is by immersing them in a virtual environment with a virtual agent as an acting partner.

Acting in a motion capture environment and acting in a movie or theatre environment is commonly very different. The biggest difference is that a motion capture actor is very limited in the amount of visual and physical aid to immerse him in the environment. A movie actor often has sets, props and clothing to help with imagining the scene. Motion capture actors have to wear specific clothes with reflective points on them that allow the cameras to capture their motions. Movie or theatre actors can see the environment they are acting in. In a motion capture environment there is no scenery.

Imagination Studios is looking for ways to help the actor improve his performance. One way the actor’s performance can be improved is to supply something that can immerse him in the scene that is being recorded. There are two broad approaches that can be taken when immersing someone by virtual means. The solution can focus on a virtual environment or the interaction.

This thesis combines a virtual environment with a virtual agent that can be interacted with. The focus is on the interaction with the virtual agent. The benefit of having a virtual agent to interact with instead of another person is that the virtual agent can be displayed as anything. It allows for more diverse acting partners.
4 Design Method

Early in the thesis process, three main parts were decided upon. The first part was to do research. Find out what has been done related to virtual agents, interaction methods and displays. This part also included deciding what kind of tools to use. The second part was creating scenarios and a video sketch, which will be explained in detail in this chapter. The third and last part was the implementation. This part included creating the virtual environment, implementing gesture recognition and testing.

4.1 Creating Scenarios

The initial scenarios were created after the research part of the thesis process. The scenarios created for this thesis are a description of backgrounds and settings which have accompanying interaction scenes where the virtual agent and actor interact with each other. Their purpose was to explore how the product could be used. An important part of creating the scenarios was to not focus on the limitations of the solutions. The scenarios can be adjusted to work within the limitations at a later stage. Early in the process it is beneficial to keep an open mind. With this method, diverse scenarios were created. Short descriptions of the first four scenarios:

- Desert battlefield. The actor is a captain in the army and is interacting with a soldier under his command.
- Hostage situation in a bank. The actor is a security guard trying to resolve the situation with the bank robber.
- Police interrogation. The actor is a suspect in a murder investigation; he is being interrogated by a detective.
- Alien encounter. The actor is a four legged alien that is trying to attack a futuristic soldier in a jungle.

After discussions with the thesis supervisor it was decided that the desert battlefield and hostage situation scenarios would be the focus for testing. The interaction scenes were combined to use some of the same gestures and animations.

4.2 Video Sketching

Video sketching is a communication tool to explain ideas in an easily digestible format. A video is made that shows a prototype and tells a story that explains how the prototype is supposed to work. Video sketching is a fast prototyping method. The material needed for the video sketch can be collected in a few minutes. A video sketch can be a series of pictures in a slide show or a video file. The pictures or scenes should be taken from the same angle to make it as clear as possible. The process of creating a video sketch is as follows:

- Make a script (idea, interaction, product)
- Make a storyboard (scenes, actors, location)
- Shoot the pictures/scenes
- Add text or voice

For this thesis, it was decided that a slide show format would be best for explaining the product. To effectively explain the product, a visual representation of the virtual agent had to be added. This
would be hard to add in a video format. Video sketches are not meant to take a lot of time to make. The pictures for the video sketch were taken in the motion capture studio in front of the projector screen. The projector screen was chosen as it was already in place. A screen that was closer to the ground would have been better but it was sufficient to explain the idea. An adaption of the desert battlefield scenario was used to create a video sketch where the actor is interacting with a soldier. Figures 11-17 display a sequence of pictures from the video sketch that shows a potential scenario the system can be used in.

![Figure 11: Video sketch scene 1](image1)
![Figure 12: Video sketch scene 2](image2)
![Figure 13: Video sketch scene 3](image3)
![Figure 14: Video sketch scene 4](image4)
![Figure 15: Video sketch scene 5](image5)
![Figure 16: Video sketch scene 6](image6)
The 3D model in the video sketch was provided by IMS and the model was positioned in Autodesk Maya. The text was added and the pictures were put together in Windows Movie Maker. The video sketch was showed to the company supervisor and he gave feedback that he understood the thesis idea quite well based on the video sketch. The video sketch was added to the internal company Wikipedia page for employees at IMS to check out and see what the thesis was about.
5 System Design

There were several things to keep in mind when designing the system. The requirements and suggestions from the thesis supervisor were used as a base. The system had to be appropriate for a motion capture environment. This entails that the system needed to be mobile and easy to set up.

5.1 Requirements

These were the initial requirements of the technologies that were to be used in the thesis:

- Kinect
- UDK
- MotionBuilder
- C++

The initial requirements could be changed if there was a reason for it. They were guidelines of what should be expected to work with during the thesis. All of the technologies listed were used in some way, along with other technologies and tools that were found to be beneficial to the thesis during the research.

A strict requirement was that the gesture recognition had to be device-free. This meant that no accessories on the actor should be used in order to get more accurate gesture data. The reason for this is that the motion capture actor has to be able to move freely.

The final product will not necessarily be used alone. It is designed to be a part of several projects that IMS’ R&D department is working on. It is possible that it will be integrated in another project, be it the whole product or parts of it.

5.2 Environment

The motion capture studio at IMS is a big room with the dimensions 7x12x15 meters. There are 38 motion capture cameras in the room. There is a control room located where you enter the studio. In that control room, the technicians that handle recording and related tasks are situated. Next to the control room is a small lounge area. There is a projector screen in the MoCap studio that can be used to review the recorded scene among other things.
Figure 18: IMS MoCap studio entrance angle

Figure 19: IMS MoCap studio inside angle
5.3 Mobility
In a motion capture room, props like cages, chairs and mattresses have to be moved in and out a lot. Use of space in the environment changes depending on what is going to be recorded. Scenes are recorded in various locations in the room. The system developed for this thesis needs to be mobile. It should be possible to move it in and out of the shoot area with ease. The ability to move it around in the shoot area is beneficial as well. Brainstorming was conducted to figure out how to make the system mobile. A trolley on which the screen could be placed with room underneath for the Kinect sensor and a small laptop or a netbook with a wireless connection would be optimal. The software that is connected to the Kinect and the software that runs UDK communicate over the network. This means that the system can be divided to two computers. The computer running UDK can be in the control room. The computer connected to the Kinect can be on the trolley. A connection would be required from the UDK pc to the screen. Wireless HDMI connections like the IOGear Wireless HD Digital Kit (IOGear n.d.) can solve this problem. This can be solved by a long HDMI cable, but it would be problematic with a lot of cables lying around in the MoCap room. A power cable for the Kinect and the screen is required but until reliable wireless power is available, there is no way around that. The solution would be easier if the UDK and Kinect software is run on the same machine, which is possible. The benefit of having the UDK pc in the control room is that events (e.g. explosions, pulling a gun) can be triggered from there.
6 Implementation

Implementation was started two weeks after the thesis started. The first two weeks were focused on research and design. The first part of implementation was to get familiar with UDK and create an environment for the virtual agent. Designing of the UDK level was done in the Unreal Editor that is downloaded with the UDK. The second priority was to get familiar with FUBI and the Kinect. FUBI is downloaded as a Visual Studio 2010 solution; for that reason VS2010 was used as the primary development tool. Development for UDK was conducted with nFringe (Pixel Mine n.d.) which is a language service for VS2010. The last important part of implementation was to make UDK and FUBI work together.

6.1 UDK Environment Design

An environment for the virtual agent had to be created. It was decided that the desert battlefield scenario would be the base for the first level. The first step was learning how to design levels in UDK. There are many tutorials available from Epic Games (Epic Games n.d.). In a meeting with the thesis supervisor it was decided that the desert level should have four different scenes. The virtual agent would stand in the middle and the camera could be moved around in 90 degree steps. The virtual agent would be rotated to face the camera. The scenes created had to be different while still being feasible in a desert environment. The technician in the MoCap studio should be able to change the scene by pressing a button. The scenes should be different from each other to make them cover a wider variety of scenarios that the MoCap actor has to imagine. At a later stage in the thesis process it was decided that a sand dune would be fitting as a fifth scene. Various meshes and textures that are included in UDK were used in the level.
For the first scene, a building fitting for a desert environment was created. The structure was formed using methods learned from the tutorials. Suitable meshes were added to create an appropriate atmosphere. Some of the barrels and boxes seen in the scene are free meshes from a creator called Nobiax (Nobiax n.d.).

Initially, the second scene only had some plants from the Nobiax mesh package in the background. A UH-60 Blackhawk Helicopter that was downloaded from a site with free 3d models (The Free 3D Models n.d.) was added to the scene later in the thesis process. There were some issues with importing the mesh from Maya to UDK since it was very big. This was solved by separating the
helicopter mesh into two parts and importing them individually. The meshes were grouped together in UDK to form the helicopter. Animations were created for the main and rear rotor. They rotate to imitate a hovering helicopter.

![UDK scene 3](image)

**Figure 22: UDK scene 3**

The third scene is simple. A BTR-80 armored personnel carrier was added to the background. The BTR-80 was downloaded from the same free site as the helicopter mesh.
The fourth scene is an entrance to an ancient temple. The palm trees are from the Nobiax mesh package, everything else is default UDK assets.

Figure 23: UDK scene 4

The last scene that was added later is the sand dune. The sand dune was created with the terrain tool that is available in UDK. This tool is used to form terrain. It can be used to create anything from mountains to lakes.

Figure 24: UDK scene 5
6.2 FUBI
After the virtual environment was ready, the next step was working with FUBI and the Kinect. Development with FUBI was done in Visual Studio 2010. FUBI offers a C# wrapper if that is preferred, but C++ was chosen since it seemed easier to integrate with UDK. FUBI allows gestures to be defined in XML format or in C++ code. Defining gestures in XML is a fast way of prototyping. In FUBI, there are three basic recognizers that can be defined: JointOrientationRecognizer, JointRelationRecognizer and LinearMovementRecognizer. These are defined and given names. A PostureCombinationRecognizer can be defined, which is composed of any of the three basic recognizers. In this thesis, combinations of JointRelationRecognizers and LinearMovementRecognizers were used for most gestures.

An example of a gesture that was created for this project is the “Throwing” gesture. Two JointRelationRecognizers are defined, one for checking that the right hand is below the torso and the other for checking that the right hand is over the right shoulder. Two LinearMovementRecognizer are defined, one for checking if the right hand is moving fast forward and the other for checking if the right hand is moving up. Finally, a PostureCombinationRecognizer with two states is defined. The first state checks that the right hand is below torso and that the right hand is moving up. The second state checks that the right hand is over the right shoulder and that the right hand is moving fast forward. If these conditions are fulfilled in the correct order, a “Throwing” gesture is recognized and handled by the program.

The gestures that were created for the final product are the following:

- Salute
- Handing out (paper)
- Hands up
- Hands down
- Throwing
- Aiming pistol
- Lowering pistol
- Shooting pistol
- Aiming rifle
- Lowering rifle
- Shooting rifle

The gestures that proved the most troublesome were “Handing out”, “Shooting pistol” and “Lowering pistol”. Much testing and tweaking was done with these until it worked as intended. When many of the movement checks were eliminated from the gesture definitions it worked a lot better. Instead of checking movement between joint positions, only the joint positions were checked and it was much more robust in testing.

6.3 UDK
It was necessary to learn how to animate skeletal meshes through UnrealScript. Some animations for a default 3d model from Autodesk MotionBuilder were supplied from the thesis supervisor. With these, development and testing was done to play animations on a skeletal mesh from unrealscript.
This was an essential part, since an animation would be played when a gesture was recognized with the Kinect.

Camera control was another thing that had to be implemented in UnrealScript. On a keypress (1-5), the camera should move to the corresponding scene in the level. Keybindings was done through UDK configuration files. A key is bound to a specific function in the Controller class. These functions then call on the correct function in the PlayerCamera class. The camera class essentially consists of a switch in the UpdateViewTarget function that checks which scenario is active and changes the camera location and rotation to the correct values. The camera is only moved if a change has been requested.

6.4 FUBI -> UDK

When FUBI and UDK were working correctly, a connection between FUBI and UDK was needed. FUBI needs to be able to notify the UDK application when a gesture is recognized. For performance reasons, it would be optimal to have FUBI running inside the UDK application. This is not possible without access to the Unreal Engine source code. It is possible to interface UDK with native (C++) code through the use of dll binding (Epic Games n.d.). UDK can bind to a windows dll and call functions in the dll. It is theoretically possible to port FUBI to a dll format that can be used by UDK. For this thesis, it was decided that it would take too long and might not be completed in the short time span of a bachelor’s thesis. Instead, the decision was made to separate UDK and FUBI and to create a communication link over the network. This solution might not be as effective as embedding FUBI as a dll, but it has the added bonus that the applications can be separated to different machines.

A dll was created for UDK to interface with. The dll is the client side of the connection. It uses the WinSock2 API to connect to the server. Both the server and client side code is heavily based on examples found on msdn (Microsoft 2012). The dll starts a windows thread that checks the socket for new data. When new data is received, it is added to a queue that UDK can retrieve the gesture from. Gestures are sent as strings, but the foundation for sending struct types is there if needed. UDK requires strings to be sent as a struct that contains information about the current and maximum length of the string. In the UnrealScript class that handles the virtual agent, a timer runs and checks if new gestures have been received. If gestures have been received, all gestures are retrieved and handled in UDK. The initial thought was to have a thread running in UDK, but multi-threading is not supported so a timer would have to be sufficient.

FUBI was extended with a WinSock2 server. FUBI listens for connections when it is started. When the UDK application is started, it connects to FUBI. FUBI also has a thread running that will send gestures to the UDK dll whenever they are detected. If FUBI hasn’t recognized any gestures for 5 seconds, an “Inactive” message will be sent to keep the connection alive.
7 Testing and Evaluation

7.1 Acquiring Motion Data

During most of the development, a 3D model supplied with MotionBuilder and animations created by the thesis supervisor was used for testing in the UDK application. For later testing, real animations on a soldier model was needed. IMS had a 3D soldier model that could be used for the project. When it was clear which gestures and animations was needed for the scenarios, motion data had to be collected. The Master’s student at IMS also needed some motions for his project. Motion data for both projects was captured 8th of October 2012. The master’s student was the motion capture actor for these motions and the thesis author was the director. The director’s job was to make sure that the captured motions were good enough. A list of the captured motions for this project:

- Handing out paper.
- Give out paper.
- Raising a handgun.
- Shooting motion with handgun.
- Getting shot and falling down to the ground.
- Hands up.
- Hands down.
- Throwing away gun to the ground from aiming position.
- Throwing away gun to the ground from holster.
- Walking away to the side.
- Reacting to a grenade: crouching down and protecting head with arms.
- Idle stance.

7.2 Tests

The majority of testing involved gesture recognition. Developing gestures entails a lot of testing and tweaking. A problem when one person is continuously developing and testing gestures is that it is easy to keep repeating a gesture in the same manner. The developer knows the parameters and how the gesture should work, therefore it is necessary to bring in someone else to test as well. The thesis supervisor helped a lot with testing the gestures on several occasions. Having other people help with testing the gestures affected the development and the gesture recognition is a lot more robust because of it.
7.3 Evaluation

The system has yet to be tested with a real motion capture actor in a motion capture studio. The reason for this is the cost involved in hiring actors and technicians. The environment and scenario would have to be designed specifically for the motion capture scenes that were to be recorded. The testing of the system has led to a lot of improvements and finally a robust system with good gesture recognition. The gestures developed for this system will be used in other projects at IMS.
8 Conclusion

8.1 Final Product
The final product is two applications that together form an interactive application. The FUBI application that is connected to the Kinect sensor detects gestures performed by the user and sends them over the network to the other application. The application developed with UDK receives the gestures and displays an appropriate reaction from the virtual agent. The virtual agent is displayed in a virtual environment that was created in the Unreal Editor. The virtual environment is a desert that consists of five scenes that can be changed with a key press.

The application is designed to immerse a motion capture actor in a virtual environment. Testing made it apparent that it was an interesting experience to interact with the virtual agent. Employees at Imagination Studios were very interested when observing tests of the product.

The state of the art in this report shows that there exist similar human – virtual agent interaction applications. During the research, no human – virtual agent interaction project specifically made for assisting motion capture actors were found. In this aspect, this thesis is innovative.

This project will be used in subsequent projects at IMS. The gestures that were defined for this project, is already being used by the master’s student at IMS. The virtual environment, or parts of it, might be used as well.

8.2 Issues
The main issue that had to be solved in this thesis was how to make Kinect, FUBI and UDK work together. Converting FUBI to work in dynamic link libraries and connect it to UDK was explored but it was decided that it might take too much time. A solution with network communication between the applications was decided to be the best option. A big motivation for this was that previous experience with network programming made it clear that it would definitely work. This solution allows the choice to either have both applications running on the same machine or to have them separated on two different machines.

Testing and fine tuning the gestures that was defined in FUBI was another issue that proved difficult and time consuming. Much of this testing was performed near the end of the thesis process. Testing gesture recognition entails a lot of testing, tweaking and repeating. The position of the sensor is different depending on the environment it is being tested in and this can change the effectiveness of the gesture recognition. The end result is a product of many testing sessions with different people.

8.3 Own Thoughts
I am satisfied with the result. It is an interesting application to use. Even the people that are not meant to be the end users find it interesting and fun to interact with the virtual agent. I had to learn how to use these applications for this thesis: Motionbuilder, Unreal Editor and Maya. I had to learn UnrealScript, which was a new scripting language for me. I was not familiar with animation and motion capture before this project, I learned a lot about both. It was a very nice experience to work with Imagination Studios. The people who work at IMS were very friendly and helpful.
8.4 Future Work
There are some improvements that could be made to the product. The program code has room for performance improvements. The threads and the network communication parts can be made more efficient. Given more time to work on the project, these parts could have been improved. The final product is not very portable in the sense that it is not very easy to replace the virtual agent with new animations if there is a need to make the new virtual agent react differently. This could be solved by making a system where animations and reactions for the virtual agent are read from XML files with a specified format.
Bibliography


