FAKULTÄT
FÜR !NFORMATIK
Faculty of Informatics

A Virtual Dressing Room based on Depth Data

DIPLOMARBEIT

zur Erlangung des akademischen Grades

Diplom-Ingenieur

im Rahmen des Studiums

Medieninformatik

eingereicht von

Philipp Presle
Matrikelnummer 0526431

an der
Fakultät für Informatik der Technischen Universität Wien

Betreuung: Priv.-Doz. Mag. Dr. Hannes Kaufmann
Mitwirkung: -

Wien, 07.06.2012

(Unterschrift Verfasser) (Unterschrift Betreuung)

FAKULTÄT FÜR !NFORMATIK
Faculty of Informatics

A Virtual Dressing Room based on Depth Data

MASTER'S THESIS

submitted in partial fulfillment of the requirements for the degree of

Diplom-Ingenieur

in

Media Informatics

by

Philipp Presle
Registration Number 0526431

to the Faculty of Informatics
at the Vienna University of Technology

Advisor: Priv.-Doz. Mag. Dr. Hannes Kaufmann
Assistance: -

Vienna, 07.06.2012

(Signature of Author) (Signature of Advisor)

Erklärung zur Verfassung der Arbeit

Philipp Presle
Gschwendt 2B/1, 3400 Klosterneuburg

Hiermit erkläre ich, dass ich diese Arbeit selbständig verfasst habe, dass ich die verwendeten Quellen und Hilfsmittel vollständig angegeben habe und dass ich die Stellen der Arbeit - einschließlich Tabellen, Karten und Abbildungen -, die anderen Werken oder dem Internet im Wortlaut oder dem Sinn nach entnommen sind, auf jeden Fall unter Angabe der Quelle als Entlehnung kenntlich gemacht habe.

(Ort, Datum) (Unterschrift Verfasser)

Abstract

Many of the currently existing Virtual Dressing Rooms are based on diverse approaches. Those are mostly virtual avatars or fiducial markers, and they are mainly enhancing the experience of online shops. The main drawbacks are the inaccurate capturing process and the missing possibility of a virtual mirror, which limits the presentation to a virtual avatar. By utilizing depth cameras like the Microsoft Kinect it becomes possible to track the movements of a body, extract body measurements and furthermore create a virtual mirror with the corresponding video stream. The video image can be merged with a piece of clothing frame by frame.

The project is implemented in Unity, a programming environment for 3D applications. OpenNI and the NITE middleware are used for various fundamental functions and for the tracking process in combination with the Microsoft Kinect.

Taking a closer look at the results, several 3D cloth models were created and textured. The pieces of garment are based on a female 3D model. The clothes are adapted to the body of the user in front of the Kinect during runtime. In addition, cloth physics are taking care of a realistic representation. Furthermore trying on clothes in front of different backgrounds and surroundings (e.g. at night) shall be possible. Also a lot of value is placed on the interaction aspects of the Virtual Dressing Room.

Kurzfassung

Viele der momentan existierenden virtuellen Umkleidekabinen basieren auf unterschiedlichen Lösungsansätzen wie beispielsweise virtuellen Avataren oder aber auch fiducial Markern. Vor allem in Webshops kommen solche Lösungen zum Einsatz. Diese haben den Nachteil, dass sie öfters ungenau sind und keinerlei Möglichkeit eines virtuellen Spiegels bieten, also lediglich einen Avatar darstellen. Durch die Verwendung von Tiefenkameras wie der Microsoft Kinect wird es nun möglich, die Körperbewegungen und Körpermaße einer Person zu ermitteln und weiters mit Hilfe des zugehörigen Videostreams eine Art virtuellen Spiegel zu erstellen. Frame für Frame soll so das Videobild mit einem Kleidungsstück 'verschmolzen' werden.

Implementiert wird dieses Projekt in Unity, einer Entwicklungsumgebung für 3D Anwendungen. Um diverse grundlegende Funktionen und auch das Tracking mittels der Microsoft Kinect durchzuführen, wird OpenNI und die NITE Middleware verwendet.

Für das Endresultat wurden 3D-Kleidungsmodels erstellt und texturiert. Die Kleidungsstücke basieren auf einem weiblichen 3D-Modell. Diese werden während der Laufzeit auf den jeweiligen Benutzer vor der Kinect angepasst. Zusätzlich sollen Cloth Physics für eine realistischere Darstellung sorgen. Auch ein Ankleidevorgang vor verschiedenen Umgebungen und Hintergründen (beispielsweise bei Nacht) soll möglich sein. Weiters wurde großer Wert auf eine unkomplizierte und schnelle Interaktion gelegt.

Contents

CHAPTER 1

Introduction

1.1 Overview & Motivation

Trying on clothes in stores today is one of the most time-consuming tasks. Usually long waiting periods have to be taken into account, for example when standing in front of full fitting rooms. Furthermore, additional time is lost when taking clothes on and off. Reducing this time and helping people to put on a large collection of garment in reduced time was a relevant motivation for this thesis. Using modern technology - hardware as well as software - the try-on experience can be drastically improved.

Even in web shops people are very sceptic buying clothes because a try-on of clothes is not possible. The techniques discussed in this paper can enhance the shopping experience. It even offers customers a more precise representation than 2D images of the cloth they are willing to buy and therefore this may also reduce the amount of goods the buyers return.

In this thesis I will introduce a Virtual Dressing Room, which offers a solution for the mentioned aspects. The application is based on a mirror, represented by a display that outputs the image of the camera. If a person is standing in front of this virtual mirror, the person will be able to select desired clothes. The selected garment is then virtually superimposed with the image recorded by the camera. In general, this technique can be categorized under augmented reality (AR), where a real-time view of the reality is extended and furthermore overlaid with additional information. This paper mainly focuses on the applications in cloth stores although a home setup is possible as well.

1.2 Goals & Challenges

The aim of the thesis is to create a Virtual Dressing Room that realistically reflects the appearance and the behavior of garment. It should further adapt to specific bodies of different persons depending on their body measurements. This will be one of the main challenges since the pieces

of cloth should correctly fit to as many persons as possible independent of their individual dimensions.

Technically speaking, the fitting room will be based on the Microsoft Kinect, an innovative technology which provides a new way of interaction between humans and the computer. From the depth image of the Kinect the skeleton is extracted and the position and orientation of the cloth are adapted in regard to the joint positions and body measurements. 3D models of the cloth will be overlaid with the color image from the RGB camera to obtain the function of a virtual mirror. To achieve a realistic simulation of the cloth, a physical simulation is performed on the garment. Providing a useful connection between the Microsoft Kinect and all the other parts of the application will be another ambition.

CHAPTER 2

Related Work

The following four subsections are focusing on related work. First of all, the generation of avatars is discussed. These avatars are used as a three dimensional representation of the human body. The second subsection is dealing with the generation and the simulation of the garment. The simulation part depicts different methods that will enhance the try-on experience such as physical simulations of the garment or realistic cloth textures. The next section is dealing with real time tracking technologies. Most of the existing real time virtual fitting rooms are based upon marker based approaches, optical tracking methods or depth cameras like the Microsoft Kinect. The last section is describing example systems that are comprehending the presented techniques.

2.1 Avatar generation

This section will deal with the generation of avatars that are representing the human body in many virtual fitting room applications. The measurements can be entered manually, which is usually done when using web based applications. Beyond that they can also be extracted by performing various measurement techniques, for example laser scanning or using a multi-camera capturing setup.

An example of avatar generation is presented in [6, 7]. The system is based on a web application that adjusts the mannequin according to the measurements of the human body, resizes the cloth and in addition allows a realistic simulation due to physics. In order to generate an avatar, the eight length measurements have to be entered by the user. Based on that the particular segments of the avatar are adapted by deforming the skeleton as well as using interpolations (cf. Figure 2.1). In general, the idea is based on a server which offers the data (data like 3D models, motion data, etc.) and a client that is calculating operations like animations (e.g. walking, rotation of body), size adjustments of the clothes and of the mannequin depending on the body measurements.

Body measurement	Definition
Stature	Vertical distance between the crown of the head and the ground.
Crotch length	The vertical distance between the crotch level at center of body and the ground.
Arm length	The distance from the armscye shoulder line intersection (acromion) over the elbow to the far end of the prominent wrist bone (ulna) in line with small finger
Neck girth	The girth of the neck-base
Chest/Bust girth	Maximum circumference of the trunk measured at bust /chest height
Underbust girth	Horizontal girth of the body immediately below the breasts
Waist girth	Horizontal girth at waist height
Hip girth	Horizontal girth of the trunk measured at hip height

Figure 2.1: Table taken from [7], depicting the 8 basic measurements to form an avatar.

In the paper [6,7] they are especially noting the high return rates of web stores since there are a lot of problems (e.g. clothing does not fit) that can occur if people are buying clothes online. Therefore they deal with these problems using this mannequin.

A different approach deals with the creation of a three-dimensional avatar using laser scanning [8]. That means the person who wants to try on clothes has to be captured first of all. The scanning can be accomplished during a visit at the cloth store, for example. A mesh is then generated out of the scanned point cloud and the textures, that are captured during the process as well, are applied on it. This serves as the basis for the virtual avatar.

In another more enhanced web-based application, a more natural look of the avatar is achieved by applying a face detection algorithm on a photo of the user, texturing the model its face with it and furthermore adjusting the skin color according to the face [22]. Also a direct relationship to the CAD data is provided, which would allow a generation and fabrication of real clothes in respect to the body measurements directly out of the application.

Another application described in the paper 'Free Viewpoint Virtual Try-On With Commodity Depth Cameras' [13] is using optical tracking in order to scan a person and use its body mesh as an avatar later. Using a multi-camera setup with 10 cameras (resolution: 640x480), a person (and therefore its body measurements) gets captured from all sides. To distinguish this person

from the rest of the image, a colored background is used. They used this method because a fast process of scanning is assured compared to other methods. In the following step a mesh representing a human body is adjusted so that the pose of the mesh is adapted to the pose of the scanned person, which is required for further processing. Afterwards the OpenNI [26] pose estimation algorithms are used by converting the image into a depth map. By using deformations the mesh is matched based on the images to resemble the proportions of the captured person. The described technique is illustrated in Figure 2.2. In addition, if a person was not scanned yet, it is also possible to import an avatar with self-determined measurements. The avatar can be generated in an application that is designed for the creation of human bodies, for instance.

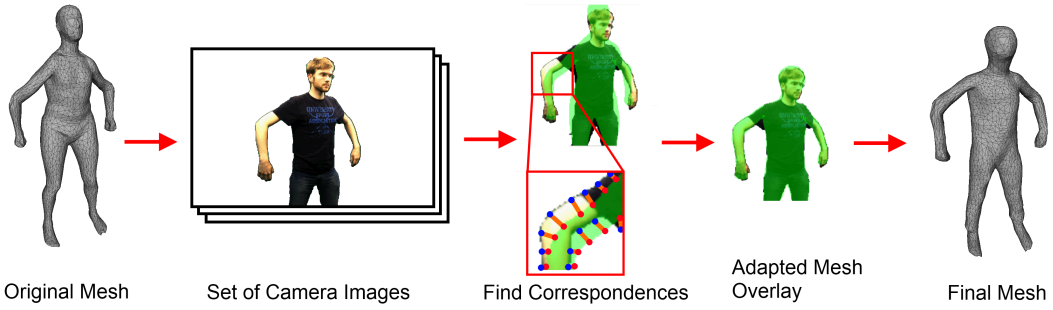

Figure 2.2: Illustration taken from [13]. The figure shows the transformations that are applied on the original mesh based on the multiple camera images.

2.2 Virtual cloth generation and simulation

In general, garment can be either modeled manually by creating a 3D mesh that represents a piece of garment or by scanning an existing cloth (e.g. using laser scanning techniques). Besides some fitting rooms are also using two-dimensional planes that have a texture applied, although mostly the results do not appear very realistic. My thesis is focusing on garment represented by three-dimensional meshes.

A very practical approach is introduced in the paper [13] regarding the generation of garment. As already mentioned, it is based on a multi-camera setup. This multi-camera setup is capturing a person that is wearing the actual cloth that wants to be scanned. Of course, the garment has to be separated from the body which is done manually or with chroma keying of the camera images according to the paper. When trying on a captured cloth at home, the size of it has to be deformed in order to match the measurements of the person in front of the Kinect (and not the one that was capturing the cloth). Furthermore physics are applied on the clothes by using the PhysX engine of Nvidia by inputting the convex hull of the body parts for collisions. Physics are mainly used to simulate long clothes like skirts.

A method of cloth generation that is quite similar to the technique I have used in this thesis

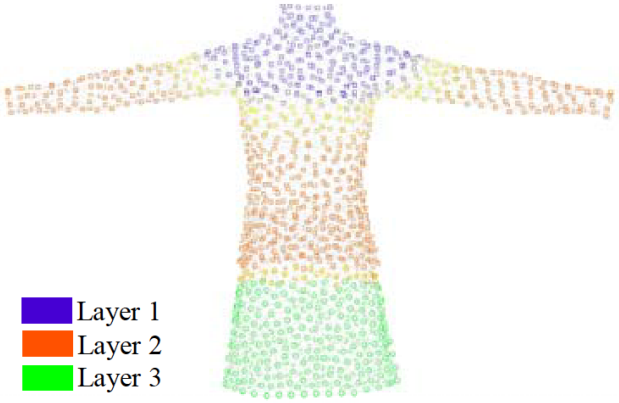

Figure 2.3: Different layers (represented as vertices) of garment [7]. The cloth is simulated differently in respect to the particular position of the garment on the body.

(see Chapter 4.4.1) is depicted in [6, 7]. The outline of a cloth is drawn, then placed on the body, and in a final step the garment is automatically adapted according to the shape of the model. Considering cloth simulation, the garment is treated differently in relation to its position on the body with intention to avoid heavy calculations (cf. Figure 2.3) [6, 7]. Layer 1 moves with the body whereas layer 2 deals with the loose clothes. Layer 3 is used for skirts, as the movement is not necessarily depending on the legs.

Many different techniques to enhance the try-on experience and furthermore to provide a realistic visualization of clothes are presented in [8]. Improvements can be achieved by using correct illumination and adequate textures on the garment, for example. A finished result should give the user a real feeling of the material properties based on the virtual cloth. This is also one of the fundamentals for a lot of virtual clothing rooms. Some of the proposed methods are discussed here. As already mentioned in the previous section, the technique uses laser scanning, therefore the meshes of the clothes can also be generated using this technique. Furthermore different tests are executed on the garment materials (bending, shearing), to achieve a realistic behavior and moreover physically correct wrinkles.

In the paper [8] clothes are also created using two-dimensional CAD geometry models that are typically utilized in the fashion industry. Additional information how to sew the garment is provided in the CAD files as well. This can be used by the application to position the particular parts of the cloth model around the avatar based on certain points of the human body and in the next step sew them virtually according to the bounding surfaces. Furthermore a system for cloth animations and physics is utilized.

To furthermore achieve a realistic look of the garment, BTF (bidirectional texture functions) are introduced. BTFs consist of images of textures that are taken from different viewing angles. A setup for an automatic measurement of garment consists of a light and a moveable camera on a rail system, that is made up to capture garment from different viewing angles. This allows an

accurate capturing of the surface of a piece of garment. Furthermore many particular properties (anisotropic reflection properties, subsurface light transport, interreflections, self-shadowing, etc. [8]) of a material can be recorded. Figure 2.4 gives a closer look on the differences between BTF and normal texture mapping. During the render process the calculated texture is assigned on the triangles according to the direction of the light and the viewing angle. Besides high dynamic range images were introduced to achieve a realistic illumination.

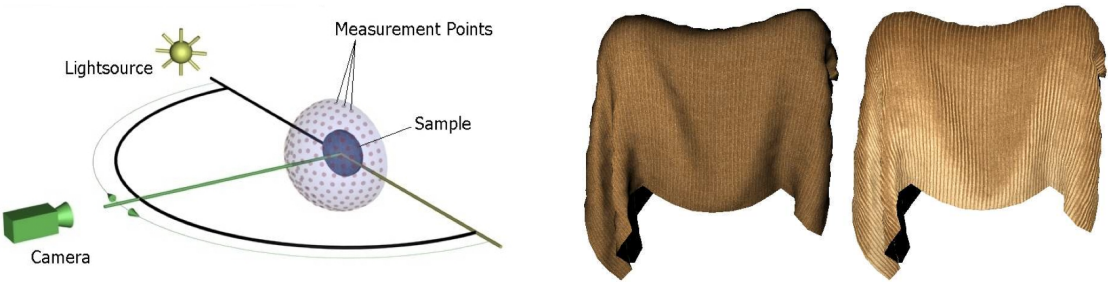

Figure 2.4: This figure on the left illustrates the capturing setup when using BTF. The camera is moved on a rail to caption the sample from all directions. The image on the right side shows the significant difference between a rendered simple texture (left) and a BTF (right). Both images are taken from [25].

Overall the described paper [8] delivers some interesting aspects concerning cloth generation that should be considered during the creation process of a virtual fitting room.

Concerning realistic cloth simulation, many applications - even the graphics engine of Unity which is used for the Virtual Dressing Room presented in my thesis (see Section 3.1.2 for more details) - are relying on PhysX by Nvidia [27]. It is a physics engine that is hardware (using the graphics processing unit / physics processing unit) accelerated and designed for the purpose of real time applications. Besides cloth physics it also supports rigid bodies, soft bodies and fluids, for instance.

A garment as seen from the PhysX engine is a mesh that consists of cloth particles. Some particles have a fixed position (e.g. the side of a flag pinned to a flagpole). Other particles are intended to move and behave physically correct. Therefore constraints are generated between the particles to achieve cloth-like effects like stretching. Using PhysX it becomes even possible to create cloth that is tearable or to put pressure inside of a cloth [27], which is also available in Unity. Furthermore the engine also features a collision detection that is executed only on the vertices. For a realistic simulation also self collisions are featured [5].

2.3 Real time tracking technologies

Concerning real time tracking, different methods exist. The most important technique for my thesis is the one dealing with *depth cameras*. Therefore this method will be explained in more detail. In general, there are three types of technologies that are typically used:

- Markers
- Optical tracking
- Depth cameras

Their exact functionality and the technical fundamentals of the mentioned methods will be discussed in the appropriate subsections.

2.3.1 Markers

This subsection is dealing with fiducial markers. In the case of virtual fitting rooms those markers are placed all over the body. Each marker has a specific pattern printed onto it in order to relate the marker to a specific position (e.g. the left hand or the left shoulder) and to recognize the specific marker later during processing. The markers are captured with a calibrated standard video camera. The resulting image is then analyzed in real time and the markers are found by a software using image processing techniques, resulting in the exact three dimensional positions and orientations of the markers.

One paper that introduces markers is the ARDressCode [19] application, for instance. The setup of the virtual mirror is mainly based on a camera that is capturing the scene. In the next step the output of the camera is analyzed (using DART, Designer's Augmented Reality Toolkit, with ARToolkit, a common tracking library for augmented reality), the positions of the markers are determined and according to that the 3D model of the garment is placed over the image of the body. The final step consists of the projection of the composite image.

According to the paper dealing with the ARDressCode application markers were attached on the middle torso and on both arms. Each marker has the particular part where the marker has to be positioned on the body imprinted, in order to simplify the process (cf. Figure 2.5: printed markers on the left illustration).

To recognize the markers even if the user is not captured by the front camera the system can be extended with 2 additional cameras. This is also shown in Figure 2.5 (illustration in the middle). To move the clothes in respect to the marker's position a bone model was introduced. The 3D model is then moving according to the bone model.

One negative aspect of this augmented dressing room is the fact that markers have to be placed before the tracking and furthermore the try on of the clothes can start. Also, the spots of the fiducial markers have to be chosen precisely in order to achieve an adequate positioning of the cloth.

Another marker-based approach is presented in the paper 'Follow-the-Trial-Fitter: Real-time Dressing Without Undressing' [1]. The whole setup primarily consists of a computer, a camera and a display that is mirroring the environment and superimposing the cloth on the person. An interesting aspect quoted in the paper is a half transparent mirror with an LCD screen behind the mirror, although it is not used because of problems with colors that are not passing through accordingly. Capturing of a person is accomplished by using small colored markers on the user. The markers are positioned on specific joints. Moreover, the markers

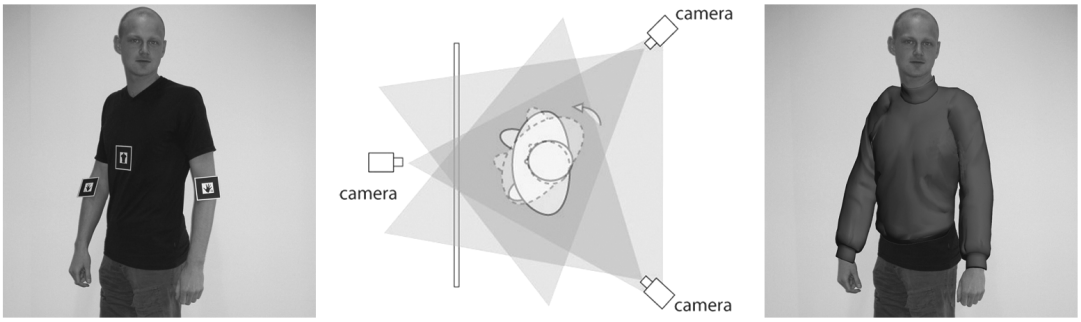

Figure 2.5: Illustration taken from [19]. The left image shows the position of the fiducial markers; the figure in the middle depicts the 3 camera positions to capture the body of the person from all sides; the right illustration shows an overlaid cloth based on the orientation of the markers.

have different colors according to the actual placement on the body. A simple image-processing algorithm is detecting the markers and according to that the 2D garment is transformed using the triangle vertex information of the cloth. The paper states that the results were quite satisfying for the front view, although a negative point of the application is that a person cannot be captured from the side. So it was not possible for users to take a look at garment from the side view, for example.

In order to achieve a more precise and accurate result concerning the position and orientation of the markers, it makes sense to take a look at optical tracking.

2.3.2 Optical tracking

Optical tracking techniques are calculating the position and orientation of a point in space by using passive reflectors (e.g. spheres) that are placed all over the human body. They are emitted with light (usually infrared light) and this light is then captured from a camera. Using multiple cameras, the three dimensional data can be obtained by implicating their captured view and the orientation and location in relation to each other camera using triangulation. Before tracking starts, a calibration is necessary as well.

A technique that is based on real-time optical motion capturing is stated in [4]. The developed application uses an avatar which is moving in a virtual scene according to captured positions. An optical tracking system is used that is based on passive reflective markers that are placed on the required points all over the body. An emitter is sending out the infrared light which is sent back by the reflectors. This light is then captured by multiple cameras and output to a computer, which is analyzing the data. By using triangulation on the captured positions the particular 3D locations of the markers can be extracted. Before the capturing of a person can start, additional calibration steps have to be taken into account. Also, reflectors have to be positioned

on the appropriate locations of the body based on the placement rules. In a last step, the movement is assigned in real-time to an avatar which is then able to move in a virtual environment. Figure 2.5 is showing a person with attached markers captured by the application. Overall, the idea is quite interesting although it is not intended for practical use since a lot of calibration work has to be done at first. In addition, no body measurements are taken into account.

Figure 2.6: Illustration taken from [4]. The image depicts the passive reflective markers that are placed all over the body in order to obtain the 3D joint positions. The camera-units, capturing the scene and emitting infrared light that is reflected by the markers, are also illustrated.

Marker-based as well as optical tracking solutions in general are a good and valuable approach. The real-time component is an important factor, although the manual positioning of markers can be quite time-consuming. Placing the markers on the appropriate positions may appear as a source of error as well. To avoid the placement of markers or reflectors it's necessary to take a look at markerless systems, for example systems based on depth cameras.

2.3.3 Depth cameras

In general, depth cameras are measuring the distance to all points of a scene and are not limited to markers, for instance. Two types of different concepts for depth cameras will be explained here: *Time-of-Flight* and *Structured light*.

On the one hand there are Time-of-Flight (ToF) cameras, capturing the traveling time of light waves. More precisely this means that the scene is emitted with light and a detector camera is measuring the time the light is traveling into the 3D scene and back. Since the exact speed of the light spread is known (approx. $c = 3 * 10^8$ m/s) it is possible to calculate the corresponding distance [32]. Therefore the time has to be measured efficiently. Those cameras provide a direct measurement of the ToF, since no complex algorithms have to be executed compared to structured light, for instance. This allows a quick scan of the scene and therefore offers fast frame rates. Compared to the Kinect, which is a mass product, the costs of a ToF system are very high.

On the other hand there are techniques that are based upon structured light. As already mentioned, the Kinect rests upon this method and therefore I will explain the technique by - first of all - describing the basic principles of this device.

The Kinect is a gadget basically aiming on motion detection. It was introduced by Microsoft in late 2010, at first for the Xbox 360, a gaming system by Microsoft. Some days thereafter a first driver that establishes a connection between the Kinect and the PCs was introduced. Nowadays the most used software development kits are the Microsoft Kinect SDK [24] or a combination of the OpenNI [26] framework, the NITE [28] middleware and SensorKinect [2]. In my thesis I will focus on the latter one. A short comparison between both of the earlier mentioned SDKs will be provided in Section 3.1.2. Generally speaking, the device offers new possibilities in the field of human-computer interaction (HCI). A new field of interactions based on the skeleton and joint positions of users is introduced. For instance gesture recognition, which is replacing peripherals such as mice, gamepads or control elements with full body tracking.

The fundamental component of the Kinect device is the infrared unit, consisting of an infrared laser and a depth sensor. The laser is projecting an infrared pattern that is captured by the depth sensor (CMOS sensor). It offers a resolution of 640x480 running at 30 frames per second, providing 2048 levels of depth [17]. The sensor can recognize objects between a range of 80 centimeters to 4 meters, although an optimal range of 1.2 - 3.5 meters for interference-free tracking is recommended by Microsoft. If a user is standing too close to the device a complete recognition of the body is impossible. A distance of 2.1 meters is the minimum distance to capture a full body, therefore it is also the recommended distance when using the Virtual Dressing Room. That allows a full tracking from head to toe in order to correctly superimpose the shirts and trousers. An RGB camera will caption the belonging color stream simultaneous to the depth stream. Face detection is one of the application areas, for instance. Furthermore the Kinect offers an array of 4 microphones that can assign a certain voice to the particular user who is speaking. This is a feature that will not be picked up in this application, since it is only supported by the Microsoft Kinect SDK. Additionally, a motor that allows a rotation of the Kinect's components, is installed. It ranges from -27 to 27 degrees. An accelerometer is measuring the tilt and therefore allows a stable tracking of the ground/floor plane.

Roughly clustered the course of the whole tracking process can be divided into three major parts. First of all, the environment - more precisely the infrared pattern - gets captured by the

depth sensor of the Kinect and all persons in front of the device are extracted out of the image. After finishing this step, an algorithm determines particular body parts. Finally, based on all the previous information, a skeleton and its corresponding joint positions are calculated.

The technique the Kinect is adopting for depth measurement is called structured light. In general, a lot of different projecting patterns exist, beginning with a single dot up to special lines, grids, colored points or binary stripes. The basic setup for such a capturing system consists of a camera and a projector. Furthermore a calibration process (which is consisting of a translation and rotation between the camera and projector) is needed. The Kinect is based upon this system. The device is able to reconstruct the depth of a scene by capturing the pattern of the infrared laser using the CMOS sensor. In particular the structured light technique is using the disparity of the projected pattern in comparison to the recorded pattern. By including the offset between the infrared laser and the sensor, the depth can then be calculated using triangulation. A special PrimeSense chip for this purpose is embedded in the Kinect [3].

A patent issued by PrimeSense is dealing with 'Depth mapping using projected patterns' [10]. A look from above on the device is illustrated in Figure 2.7. It cleary shows the casing of the device with the two infrared units. The upper one is the light source. In case of the Kinect, this is the infrared laser. In front of the laser there's a transparency, which is forming the light in order to create a pattern. Different methods for the creation of a pattern are mentioned in the patent. The lower unit is the sensor that is capturing the pattern. As already mentioned, the Kinect has a CMOS sensor. An lens in front of it is focusing the image respectively the pattern in order to hit the sensor. The advantage over other solutions is that the two infrared units are in a fixed distance to each other, which is simplifying the depth calculations.

2.4 Example systems

This section will take a look at some applications that have been realized and can therefore be considered exemplary.

A convenient system is depicted in a paper from the Fraunhofer Heinrich Hertz Institute [14]. The setup is based on a single camera and a display functioning as a virtual mirror outputting the camera stream. The camera is capturing the upper body of a person in front of the setup. Additional light sources are taking care of proper lighting. If a person is then moving into the camera's field of view and is further recognized, image processing techniques are applied. Furthermore this means that a specific region on the person's cloth can be retextured and the color of the shirt can be changed as well. In their work they are using a touchscreen to select the desired color and texture.

To perform this task, the shirt the person is wearing in front of the mirror has special requirements like green color with any desired rectangular texture in the middle. The person can move in front of the camera as long as the texture is visible to the camera. According to these facts the cloth is detected and further algorithms to separate the shirt from the rest of the image are applied. Also, the pattern texture is recognized inside of the separated green region and a mesh is generated out of it. In additional steps the deformation of the mesh is estimated. Furthermore,

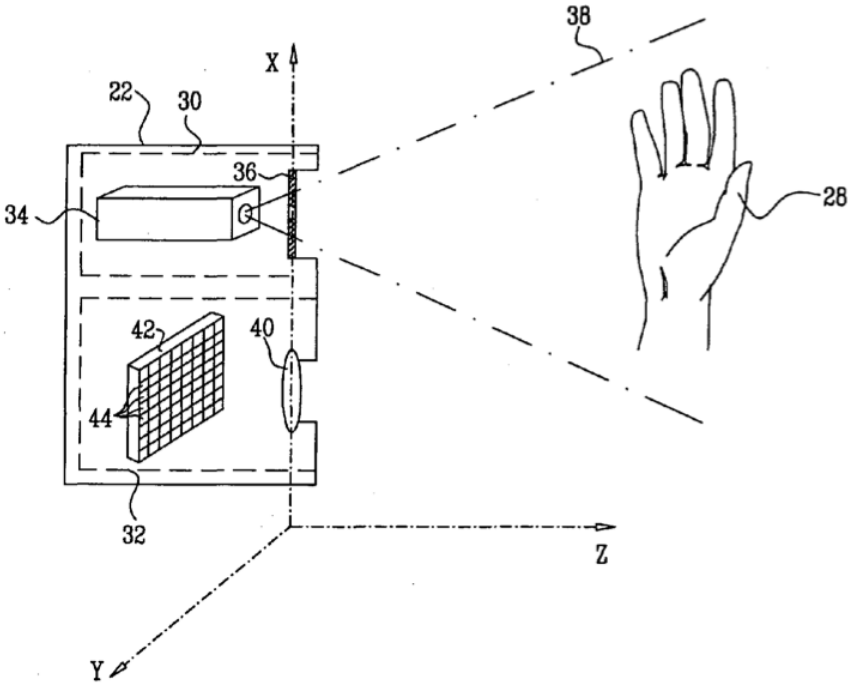

Figure 2.7: Illustration taken from the patent [10]. The whole device (22) is split into two parts. The upper unit (30) on the figure is dealing with the creation of the pattern. A hand (28) in front of the device is illuminated with the pattern. The lower unit (32) is the sensor, capturing the hand and its pattern.

to achieve realistic illumination the regions of shadows that are existing in the original image were reused on the output image.

On the one hand the described technique shows a lot of advantages. The system allows a stable real-time recognition and deformation of the cloth. The low hardware requirements are mentioned in the paper as well. On the other hand the fact that it is limited to the upper body and a special t-shirt is needed for tracking are the major downsides in respect to an easy and comfortable usage.

The paper mentioned in one of the last subsections is dealing with the ARDressCode [19] application. It introduces an augmented dressing room that is based on fiducial markers. It was designed to be mainly used in clothing stores, although a home setup is also mentioned. It offers a special approach on garment selection: The clothes in the store are equipped with RFID tags. By placing the selected garment in a special wardrobe, the system identifies the cloth by the tag. In addition, it is possible to transmit the individual body measurements via a bluetooth connection of a phone to the system. Unfortunately no detailed description of this process is given in the paper.

An application mentioned in [21] even introduces a particle system to try on cloth during different weather conditions like rain, fog or snow that leads to an even more realistic simulation.

Another system is presented in the paper 'Free Viewpoint Virtual Try-On With Commodity Depth Cameras' [13], which is part of the Narkissos Project [31] that is developed by the Institute of Computer Graphics and Vision at the Graz University of Technology. It is based on 2 major steps. First of all, the user is captured with multiple cameras and an avatar is created. This step can be realized in a cloth store, for example. The second step is carried out at home. The captured person can download his avatar as well as the garment and try on clothes while the avatar is controlled in real-time by using a depth camera like the Kinect. The process is illustrated in Figure 2.8.

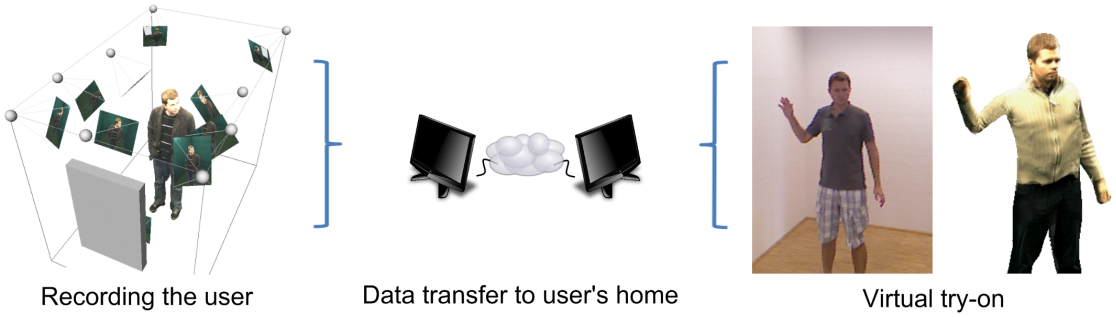

Figure 2.8: Figure illustrating the major steps [13]. A user is scanned and an avatar is created using the outputted images. At home, the user can try on different clothes on his avatar using a depth camera like the Kinect.

Moreover a try on at home is not necessary and can also be executed using the multiple camera setup mentioned. For this purpose a virtual try-on can be achieved by applying IBVH (Image Based Visual Hull) rendering, which is generating a depth map from the silhouette images and inputting the depth data into OpenNI. Everything is executed in real-time [12]. In a further step OpenNI will be used to calculate the skeleton and generate the related joint positions. This technique is an option if the whole system is intended to be adopted only in cloth stores.

A similar approach based on the Kinect is demonstrated in another application [16]. This technique is - in contrast to the method introduced in my thesis that relies on 3D cloth meshes - based on a 2D model which is scaled based on the distance between the user and the Kinect sensor and then overlaid with the video image. Skin color segmentation in the YCbCr color space is applied using thresholds, which allows the user to interact with his hands even in front of the superimposed t-shirt. Overall this technique describes an useful approach for t-shirts, although the paper does not depict the treatment of longer shirts or pullovers.

Besides, the first projects that are focusing on commercial applications are appearing. Usually those systems are based on the Kinect, since it's an easy and proper tool to create dressing

rooms that are based on augmented reality. Such virtual fitting rooms are available from FaceCake (Swivel) [11] or ARDoor [34], for example.

Some applications have also focused on other articles of clothing. A virtual mirror was created to take a look on different sport shoes, for instance. It is based on motion tracking capturing the feet and replacing the actual shoe with a virtual 3D model [9]. Another example is a system that is dealing with a virtual try-on of eyeglasses, superimposing glasses on an image of the face [20] after certain features were detected. The focus of our work is not on shoes or accessories.

Considered as a whole, a lot of approaches on creating virtual fitting rooms have been made. Some of the papers provide a lot of information concerning improvement of the realistic look of clothes, which is an important factor for an accurate visualization of cloth. Other papers are focusing on a correct recognition of the body and a useful setup for capturing. Nowadays as depth cameras combined with a color camera and based on that in combination with human pose recognition are becoming more popular - as the Kinect for instance - solutions building upon these technologies are increasing. Even in the commercial area many companies are focusing on the Kinect in order to build robust virtual fitting rooms without the need of a long calibration or a long scanning procedure of the user.

CHAPTER 3

Fundamentals

The following sections will provide a basic outline of the fundamentals. In the first section an overview of the skeleton tracking algorithm introduced with the Microsoft Kinect will be given. In addition, I will outline the frameworks that are available for controlling the Kinect device which will be important later on. Furthermore all kinds of data produced by the frameworks and their particular functionality will be discussed. The remaining parts will be covering the software solutions - Unity and Cinema 4D - that were used for the creation of the Virtual Dressing Room.

3.1 Microsoft Kinect

The exact specifications and the technical details concerning the Microsoft Kinect have been discussed in Chapter 2.3.3. The following subsections will take a look at the tracking procedure and will discuss the frameworks that are needed in order to successfully use the Kinect.

3.1.1 Skeleton tracking

A substantial part in the tracking process is the retrieval of the particular body joint positions. This is achieved by an algorithm introduced in 'Real-Time Human Pose Recognition in Parts from Single Depth Images' [30]. The recognition system runs at 5 ms per frame. Also, the system computes each frame independently from every other frame, furthermore implicating no joint information (e.g. transformations) between frames. The algorithm allows a full rotation of the body and a robust distinction between the left and right side of a body.

Figure 3.1 gives an overview of the recognition process. By inputting the depth data of the Kinect device, special regions of the body are recognized and based on this the joint positions are defined. The joint positions are needed in order to move a skeleton, for instance, or in case of the Virtual Dressing Room, a piece of garment in respect to the motion of a user. The next paragraph will take a closer look on the process.

As already mentioned, the algorithm is built upon the depth data. This data is analyzed and a human body is separated in different regions that are representing several body parts. The

human body can be captured accurately and robustly even processing different sizes of a person's body correctly. For each pixel the corresponding body part is calculated. Overall the system is trained with data from a comprehensive motion capturing database, which is consisting of humans with different body sizes and measurements, furthermore using a randomized decision forest. The parts are then assigned using depth comparisons. In a further step, the joint positions are evaluated based on the body parts using mean shift, also retrieving the 3D positions for all joints.

Compared to other methods, the skeleton tracking algorithm introduced for the Kinect shows stable and efficient results in real-time. This is the most important part for the Kinect, since it has to provide a stable and fast tracking system on various fast-paced games on the Xbox 360, for instance.

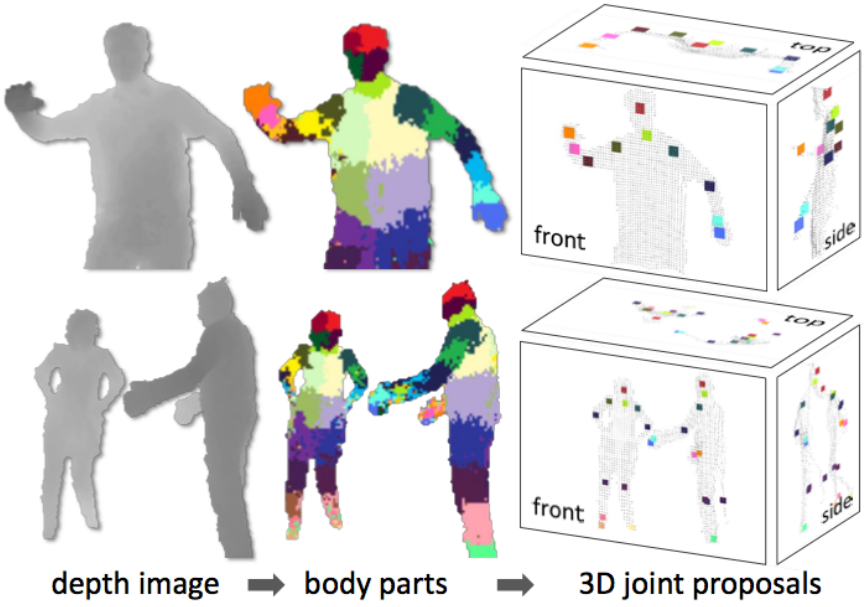

Figure 3.1: Illustration taken from [30]. It shows the different steps that are executed on the depth image data in order to retrieve the final 3D joint positions of a human body.

The next section will give an overview of the software modules that are required in order to obtain the joint positions.

3.1.2 Software modules

In general, there are two major solutions that are available to use the Kinect as a tracking device. On the one hand there is the Microsoft Kinect SDK [24], on the other hand there is the system composed of OpenNI [26], NITE [28] and SensorKinect [2]. Both solutions have their advantages and disadvantages.

The major advantage of the Microsoft's SDK is the support of tilt motors and Kinect's microphone arrays, which further enables speech recognition features. Furthermore no calibration pose is needed. The limitation to Microsoft Windows and no support for Unity3D, which will be used for this Virtual Dressing Room, are negative aspects.

For my application I will be using a combination between OpenNI, NITE and SensorKinect. Those were available several months before the Microsoft Kinect SDK was released. This has the advantage that it is platform-independent, offers support for Unity3D, and has lot of features and predefined functions (e.g. gesture recognition) already built in. On the downside, a calibration pose has to be executed by the user in order to start tracking and the installation takes a bit longer since 3 different software packages have to be installed in order to work correctly.

This abstract will give an overview of the three different software solutions needed for tracking.

- OpenNI
- NITE
- SensorKinect

First of all, there is *OpenNI* [26], an open source framework that is offered by PrimeSense. This framework forms the basis for the connection to the hardware parts and also to the software, more precisely offering application programming interfaces for human natural device interaction. In addition we need drivers for the Kinect. Those are called *SensorKinect* [2] and are offering access to the RGB camera and the depth sensor, for instance. Also, the *NITE* middleware [28] is needed. This module offers features like algorithms for stable skeleton tracking, a recognition of movements in real-time and furthermore methods for gesture recognition. Figure 3.2 is illustrating the connection between the different components.

OpenNI is producing several kinds of data that will play an important role in a later chapter that is dealing with the implementation of the Virtual Dressing Room:

- Production nodes
- Production chains
- Capabilities

Production nodes are, as the name suggests, producing data. Usually this is a low-level data which can be enhanced by a middleware component. This component is then converting the data into a more useful representation of the information. OpenNI offers two types of production nodes: *sensor related production nodes* and *middleware related production nodes*. In the case of the Virtual Dressing Room we will be using the depth generator and the image generator as sensor related production nodes. The depth generator is producing a depth map of the scene in front of the Kinect, the image generator is generating an RGB image recorded by the camera of

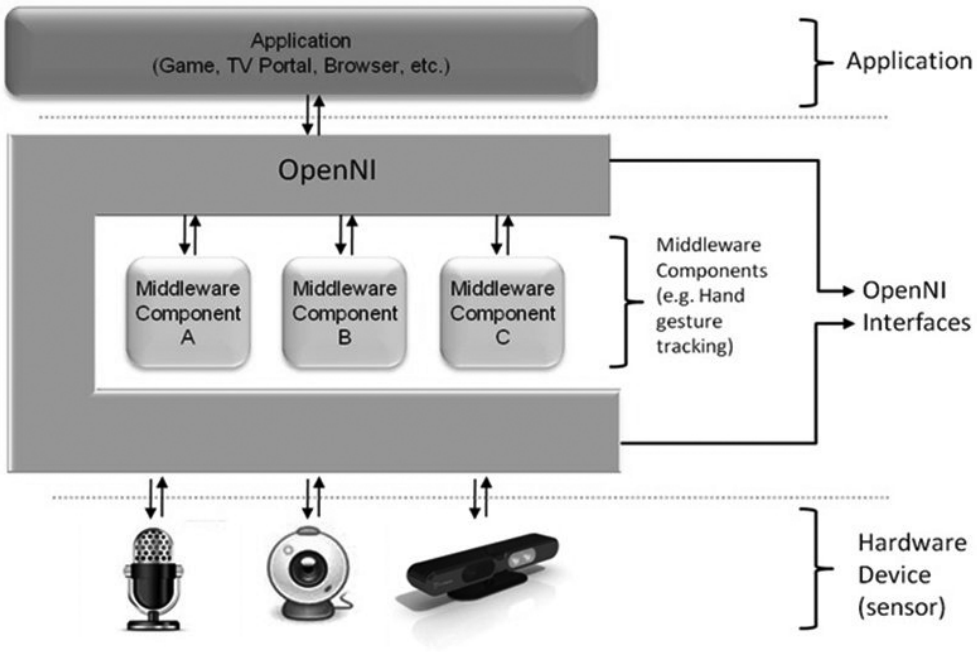

Figure 3.2: An overview taken from [26] is demonstrating the different layers and their communication with OpenNI. In this case, the Virtual Dressing Room is forming the application. It is built upon OpenNI that is maintaining the connection to the hardware and is using NITE as a middleware component.

the device. The user generator is part of the middleware-related production nodes. It is used to detect users and separate users from the rest of the scene in an recorded image.

A *production chain* is generating data based on consecutive nodes. For example a user generator is producing data based on the depth map from the depth generator, which is getting the data from the device. Furthermore this allows to use different middlewares on different sensors, offering various production chains.

Capabilities are an important element of OpenNI. They offer different resources for the production nodes in respect to various devices and middleware components. Capabilities are simply additional functions for production nodes. Some of the capabilities used in the Virtual Dressing Room are the following:

Skeleton capability The capability to process skeleton data and extract different joint positions of a user. This is one of the most basic capabilities and it is based upon the user generator.

Pose detection capability This is an essential function in order to detect specific poses. It is used for the calibration pose that has to be executed before the tracking can start. It is also based on the user generator.

Alternative viewpoint capability Usually the depth and the image sensors are outputting maps that are different to each other because of the offset of the two sensors. This capability allows to automatically match both images, so that they are appearing to record the scene from the same viewpoint.

Mirroring capability The capability to mirror the output streams. This can be applied on the image and the depth stream. It is needed for the Virtual Dressing Room since the display should act as a mirror, for instance representing a left foot in front of the mirror as a right foot in the mirror.

After the capabilities are defined, the production nodes can generate data by calling the appropriate commands provided by the API.

Furthermore taking a look at the NITE middleware, Figure 3.3 is illustrating the basic skeleton that is provided by this component. It's also representing the calibration pose, which is needed in order to allow the framework to execute a calibration procedure and furthermore start tracking. The calibration process usually takes some seconds. The illustration clearly shows the distinction between the left and right side. All joint positions on the left side are also available on the right side.

A more precise look on the programming side considering the processing of the data using the APIs - beginning with the depth image up to skeleton tracking - will be taken in the implementation section of this thesis.

3.2 Unity

Unity [33] will make up another important part in the development of the fitting room. Unity is a graphics engine, which is mainly used by developers for game creation. It is offered for Windows and Mac OS (as well as some mobile platforms), which also allows the Virtual Dressing Room in combination with the Kinect frameworks to be executed on both operating systems. It is offered for free, though there is a premium version offering further features (e.g. shadowing, etc.).

Unity also provides an editor, which can be used to generate scenes. Furthermore the developer can create objects (e.g. `GameObjects`), set different properties (physics, etc.), assign materials and sounds. To adjust the behavior of objects and in addition to control the program flow, scripting is used. I will be using `C#` as the main programming language, though Unity also supports `JavaScript` and `Boo`, which are all based on Mono (platform-independent).

Moreover the Unity framework was chosen because it supports lots of functions that are needed for a virtual fitting room. First of all, it features full support of the Kinect, by using OpenNI, the NITE framework and SensorKinect. To implement all API functions in Unity, it is necessary to embed the `OpenNI.net.dll` and `XnVNite.net.dll` in the project.

The second important part is the `InteractiveCloth` component of Unity. Several settings of cloth behavior can be specified in order to adapt the physical behavior to the cloth. Interactive cloth is based on colliders, that attaches the garment to certain objects, for the purpose of a realistic simulation of the cloth.

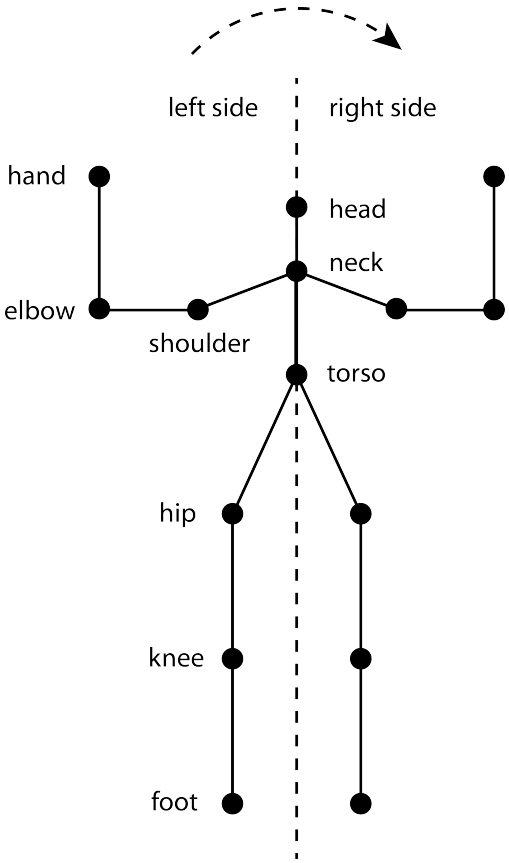

Figure 3.3: The provided skeleton and the different joint positions indicated by the black dots. The joints on the left side are also available on the right side of the skeleton. Furthermore the skeleton is illustrating the calibration pose.

Also provided is the import function for various storage formats of 3D objects (`.obj`, `.FBX`, etc.). Directly reading common formats of 3D applications, like the `.c4d` format of Cinema 4D, is also possible. The import function for objects will be used to import the three-dimensional models that will be representing pieces of cloth. Those garments are created in Cinema 4D.

In the following an explanation of essential elements of Unity for this thesis is given.

Scene A `Scene` is consisting of various objects. It is usually used for separating different levels of games to improve the overall performance. The Virtual Dressing Room is based on one `Scene`, since the loading and unloading procedure of the cloth objects is performed via scripting.

GameObjects These objects are the basic elements of Unity. Usually they represent an imported 3D model or a predefined mesh like a cube or sphere, for instance. Creating a `GameObject` produces an empty object with a `Transform` attached to it that allows the `GameObject` to be translated, rotated and scaled in space. Furthermore `GameObjects`

can be grouped in a hierarchy. Using this hierarchy actions applied on the parent are also applied on the child objects. Translating the parent object would therefore also translate the child GameObject.

Components A Component is usually attached to a GameObject. It defines certain properties and behaviors of a GameObject. Examples of this might be physic settings (e.g. InteractiveCloth, ClothRenderer, BoxCollider), different mesh operations (e.g. MeshFilter), particle emitters (e.g. MeshParticleEmitter) or audio sources.

Properties Every Component has a lot of different Properties that, if changed, affect the behavior or the appearance of the GameObject the Component is attached to, for instance.

Scripts Scripts are representing the programming files and are written in Javascript, C# or Boo. Scripts in Unity must inherit from MonoBehaviour, if behavioral functions as Start() or Update() want to be used. The Start() method is called before the Update() function and is intended for initialization of variables and a basic setup of the scene. The Update() method is called every frame and is the most important function when programming actions and behaviors. Besides there is also a FixedUpdate() method, executed at a fixed rate, and an Awake() function, also used for initializations as well as an OnGUI() method for GUI events. Scripts are usually attached to GameObjects.

Prefabs A Prefab is a predefined GameObject that can be instantiated and used in a scene multiple times. Changes made to the parameters of a Prefab are applied to all of the instantiated objects.

InteractiveCloth This is a main Component of this thesis. It simulates a realistic behavior of cloth if attached to a GameObject. Parameters as Bending Stiffness, Stretching Stiffness, Damping or Pressure are affecting the physics, for instance. The Mesh property represents, as the name suggests, the Mesh that will be used for cloth simulation.

The InteractiveCloth component also adds a ClothRenderer component to the GameObject, which is required for the rendering process and determines the appearance of the object. In order to avoid render artifacts, the standard MeshRenderer should be disabled if the ClothRenderer is in use.

Colliders These are used in combination with the InteractiveCloth. Colliders are needed to attach a cloth to an object. The vertices of a cloth mesh that are inside of a collider are in a fixed position and the remaining part of the cloth is moving and stretching according to the position of the colliders. Various Colliders can be attached to one InteractiveCloth component using its AttachedColliders property. Different types of colliders are available, such as BoxColliders, SphereColliders or MeshColliders. Figure 4.7 is illustrating the BoxColliders that are positioned all over the human body in order to adapt the cloth to the person in front of the Kinect.

3.3 Cinema 4D

To generate the garment models that will be needed in order to try-on clothes in the Virtual Dressing Room, a 3D modeling application is needed. In my thesis I will be using Cinema 4D for this task, which is developed by MAXON [23]. This application offers tools for modeling, animation, texturing, rendering and other features. Rendering of scenes is not that important since it will be carried out by Unity. Therefore I will mainly focus on the modeling tools and to some extent on texturing, which will be used for the creation of realistic garment for the Virtual Dressing Room.

Cinema 4D is offering a special cloth generation tool that can adapt the cloth to the body of a three-dimensional human body model. This further means that no complex and time-consuming modeling tasks for every piece of cloth have to be executed. This can be accomplished by simply setting the outlines of a garment in 2D, converting it to a three-dimensional object and in a last step defining the cutting and the seam and shrink the cloth. I will be using this tool to create garment that fits accordingly to the body of a human model in order to follow up here with the interactive cloth of Unity. The finished model will be imported in Unity, textures will be assigned and further processing steps will take place.

CHAPTER 4

Design

In this chapter I will take a look at the design aspects of the Virtual Dressing Room. An introduction of the provided features will be given. Subsequently, a basic setup for the application will be presented. Thereafter the program flow will be outlined, an overview of garment modeling and adaption will be given, the design of the user interface and at last the debug functions will be explained.

4.1 Features

As already stated in the introduction, trying on clothes in stores can be a time-consuming task. The goal of this thesis is to introduce a virtual fitting room that allows to take a look at several pieces of garment in a short period of time. Aside from that, a realistic appearance of the modeled cloth is important in order to achieve an identical virtual look of the garment in comparison to the real cloth.

The appearance of garment is depending on several factors: First of all, a realistic 3D model of the cloth is essential. This includes an exact mesh modeled down to the last detail also resembling the correct proportions of a human body. Second, a realistic look of the texture is essential. Third, to achieve a realistic behavior of the garment, physic simulations will be involved in the application. If trying on skirts for instance this will add relevant and realistic characteristics to the cloth letting it flutter as the user moves in front of the Kinect. Another feature that is contributing to realism of the Virtual Dressing Room is the adaption of the cloth to various sizes of the body. Depending on the measurements taken during the initialization step, the cloth colliders and therefore the particular parts of the cloth itself will adapt in respect to the body measurements of the user.

4.2 System description

The basic setup of the Virtual Dressing Room consists of the following parts:

- Microsoft Kinect
  - recording of the depth data
  - capturing the rgb video stream
- Display / Screen (large)
  - outputting the recorded video stream (mirrored!)
  - the output is superimposed with the selected garment
  - displaying the user interface for cloth selection
- Computer
  - executing algorithms for skeleton tracking
  - controlling the movement of cloth colliders
  - combining of video stream and skeleton data (same viewpoint)
  - computation for cloth physic simulations
  - etc.

Figure 4.1 illustrates the setup. The Kinect camera is capturing the scene, including the user that is moving in front of the camera. After the user has performed the calibration pose for a few seconds, the tracking will start. The display is outputting the mirrored image superimposed with pieces of cloth. The clothes are selected using an onscreen menu. Also, the full body tracking distance is illustrated in order to capture a non-interpolated skeleton. If a person is closer than the minimum distance then the non-visible skeleton points have to be interpolated. For the purpose of a stable skeleton tracking without interference a simple plane or a wall as a background behind the captured person is recommended.

4.3 Program flow

This section will take a closer look on the program flow of the application from the start of the application up to the processing steps of the garment.

First of all, after the start of the program, the standard Unity dialog appears. This dialog allows a setup of the screen resolution, an option for full screen as well as selections for the desired graphics quality. Furthermore, a second tab is presenting various settings to the user in order to set/reset different key strokes for debugging purposes. This includes the display of the debug output (e.g. frames per second, currently active users, etc.), the rendering of the colliders that are in general responsible for the movement of the interactive cloth and an option to draw the skeleton of the user. A key to quit the application and shut down all the allocated resources

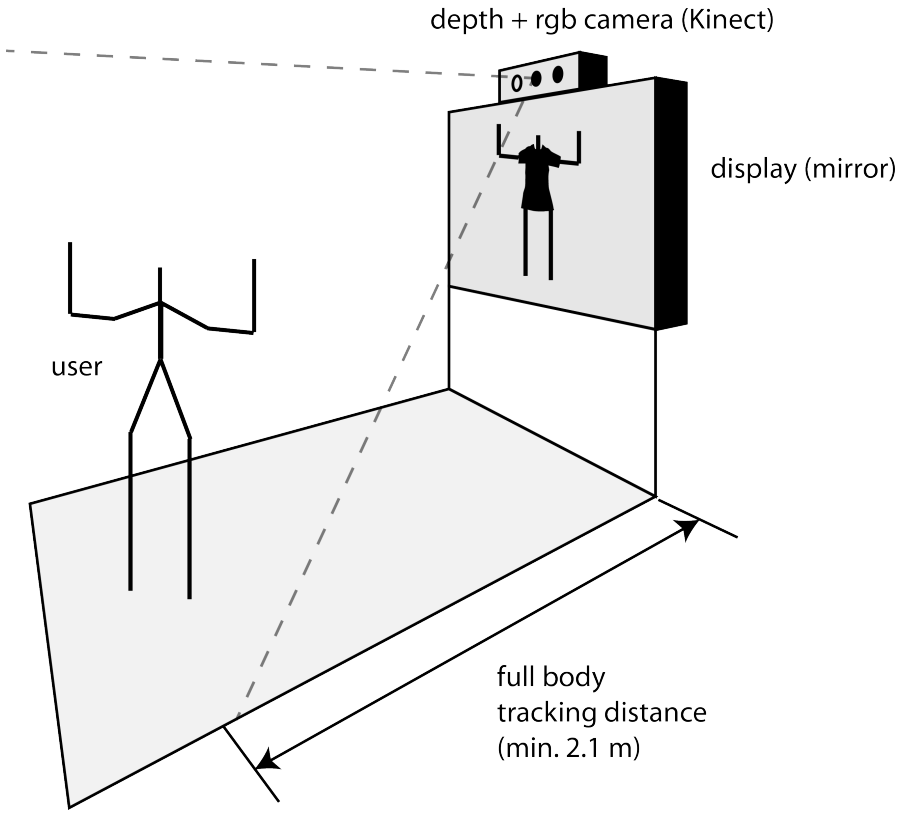

Figure 4.1: An intended basic setup of the fitting room. It consists of the Kinect device, a display as well as a computer (not illustrated here). The person in the front of the Kinect is interacting with the Virtual Dressing Room, the display showing the superimposed t-shirt selected by the user.

is provided as well. These settings shouldn't be presented to the user, since they are just basic debug settings and are intended for administration purposes only.

Shortly after the program is started the Virtual Dressing Room is ready to accept users. The users can then try on garment and can interact with the virtual try-on by utilizing the Kinect in combination with a screen, a so-called virtual mirror. These steps are described now.

The next part treats the general tracking process. After the program is fully started and the mirrored camera stream is appearing on the screen, it is ready to register users. If a user then enters the Kinect's field of view, the person is prompted to move the body into calibration pose. After the pose has been executed and measuring is finished, the registration is complete. It is now possible to access the skeleton parameters with the help of OpenNI. If problems during the calibration pose are occurring, the process starts from the beginning.

Subsequently, the user is now able to interact with the Virtual Dressing Room and chose between different garments. In general, the user can try-on different trousers and shirts at the same time. The user has to select the desired garment that he wants to change by placing his

hand over the icon. Details on the user interface will be explained in a later section.

If the user has chosen a piece of cloth, the cloth gets attached to the colliders. The first positions of the colliders are based upon the initial joint positions of the human body model since the garment mesh depends on this model. The colliders are then moved to the according joint positions of the user in real-time.

Furthermore the particular interactive cloth settings and additional informations are parsed from an `XML` file. This file is storing details on the cloth positions, price information and physic settings for every piece of cloth that is available for try-on in the Virtual Dressing Room.

Besides it is also possible for the user to enable background separation. This means, that the user can try-on the garment in different surroundings. If the person in front of the mirror likes to take a look at a cloth to wear during the night, the background of the scene can be changed accordingly.

At the end of the whole try-on process - after a user has left the viewpoint of the Kinect device - the system resets. Furthermore this means that the application is rolling back to the first step and is waiting for a new user.

Figure 4.2 is representing a rough outline of the described application cycle. After a calibration step is carried out for a new user, the data provided by the Kinect is read-out and processed. In a next basic step, the particular steps to deal with garment are executed. These consist of cloth selection, superimposing of cloth, moving the cloth colliders and applying physics, for instance.

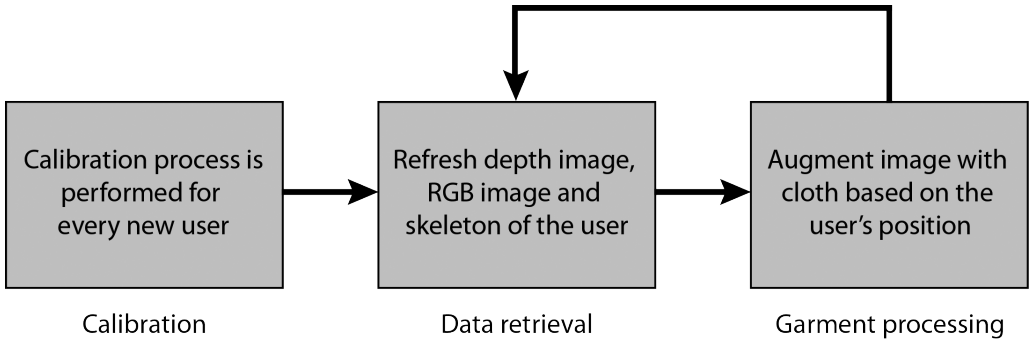

Figure 4.2: An overview of the workflow for a single user is depicted in this illustration. Starting with the calibration step, up to the data retrieval process until finally reaching the algorithms dealing with garment processing (i.e. moving cloth, applying physics, etc.). The last two steps are executed until the user leaves the tracking area. The system enters the calibration process again if a new user is detected.

4.4 Garment

The following subsections will deal with the processing of the garment. First of all the modeling and creation of the garment based on a female human model is annotated. In section 4.4.2 the adaption of the garment in respect to specific body measurements is explained.

4.4.1 Cloth generation

Modeling and texturing of clothes is an essential part, since it will determine the realistic look of the garment. The whole modeling process is executed in Cinema 4D. In this thesis I am using a human body model, which forms the basis for the other steps that are dealing with the garment part. Every garment that is created during the modeling process is adapted to this human body model. It is a standard model of Cinema 4D, although other tools, for instance MakeHuman [15], could be used for body creation as well. Figure 4.3 is illustrating the human body. In this thesis a female human body will be used. A male body could be used as well, although this would involve a re-adaptation of the cloth colliders.

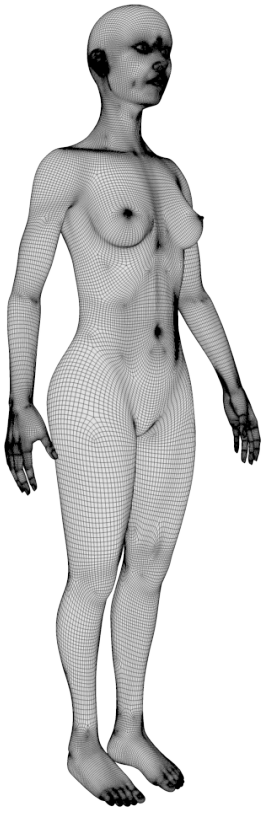

Figure 4.3: The basic female human body model in polygon view mode. All garments are based on the dimensions of this object.

The next step is the creation of the clothes and garment, which are based on this elemental three-dimensional model of the human body. A special method that automatically fits the cloth according to the body will be used for the creation of the garment.

Figure 4.4 is depicting the process of cloth generation. Beginning with polygon creation up to the final three-dimensional cloth model that can be imported in Unity in order to execute the

next steps.

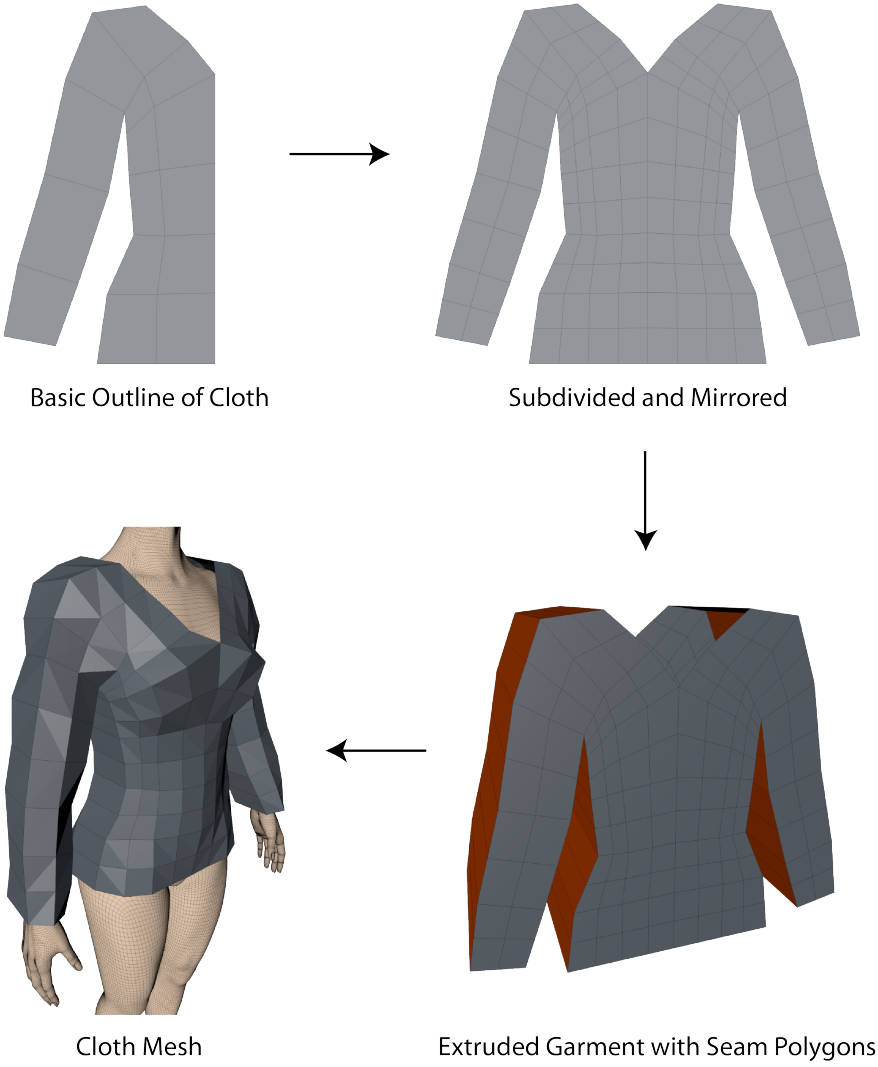

Figure 4.4: An overview of the process of garment modeling. It is starting off by defining a basic shape of polygons based on the outline of the body. After the polygons are mirrored and subdivided, the cloth is altered into a three dimensional form by cloning the polygons and defining the area in between. After setting the seam polygons a final cloth mesh is produced by shrinking the garment step by step.

First of all, a general outline of the cloth is determined by setting the polygons that are forming the garment by defining the vertices. Furthermore, taking a look at garment we see that mostly all clothes are equal if vertically flipped around the Y-axis (vertical). Therefore one side has to be modeled and the other side can be obtained by mirroring the created outline. By

duplicating the outline and connecting these two parts, on the areas where the garment doesn't show gaps (e.g. hands, neck and feet areas), a rough three-dimensional model is made. In the next step the cloth generation techniques of Cinema 4D are applied. After selecting the seam polygons of the cloth and setting the number of iterations to be executed, the garment is automatically contracted to fit accurately to the female body.

The generation of the mesh is now finished and textures can be assigned. Afterwards, the cloth is saved (either as a general format like `.fbx` or more specific as `.c4d`) and imported in Unity. The produced cloth provides the basis for a detailed positioning of the cloth colliders of Unity on the garment in respect to the female human model.

4.4.2 Garment adaption

After a garment model is imported in Unity, the corresponding texture has to be assigned and specific parameters like the scale factor of the cloth are set. When a piece of garment is loaded during runtime, the cloth has to be adapted to the specific sizes of various human bodies. In general, two approaches have been reviewed. Before implementing the concept of cloth colliders, cylinders were used to expand the cloth depending on their individual size.

The first idea was to use cylinders. Those cylinders would be adjusted according to the size of the particular body parts of a user. The cloth, equipped with interactive cloth physics, would then contract or expand corresponding to the dimensions of the adequate parts. After testing this implementation the idea was discarded. It brought up a lot of problems since sometimes cloth didn't perform reasonably and the cloth dropped off the body several times. Therefore clothes attached to cloth colliders as stated in the last paragraph were introduced. The idea is illustrated in Figure 4.5 with visible red cylinders, also showing the measured size of the body parts. Overall, this first approach did not work out well since bloating the garment caused problems and a strange behavior of the cloth and slowed down the application during runtime.

The second idea was to use cloth colliders that are stretching the cloth mesh in a specific direction. The vertices of the mesh are static inside of a cloth collider, the remaining part of the garment is behaving according to the physics. These colliders are placed all over the body, especially on the joint positions (e.g. the neck, the torso and the hips). Depending on the position every collider has different scale factors that are including various quantities of vertices. The cloth colliders are then placed according to the dimensions of the person that is using the Virtual Dressing Room. They are stretching the cloth in various directions. This results in a cloth that perfectly adapts to individual bodies. Therefore this concept was implemented in the application.

Since OpenNI and the NITE middleware are just providing a limited number of joint positions, the application has to measure additional distances. In the case of the Virtual Dressing Room we have to determine the size of the waist. The calculation is based upon the image that is separating the user from the background. Using this technique it becomes possible to calculate the arm width and other dimensions though this is not used in this thesis.

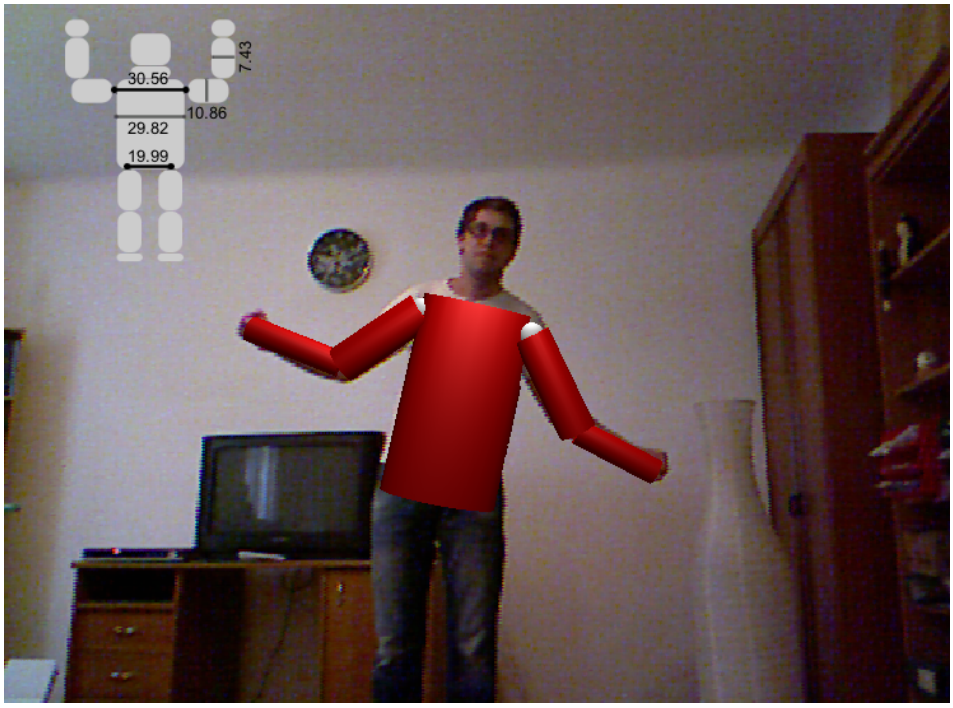

Figure 4.5: A scene showing a human body superimposed with the cylinders that should bloat up the garment. The size of the cylinders is based on body measurements taken during the calibration process. The dimensions are illustrated in the top left area only for debug purposes. The measurements between the shoulders and the hips are taken directly from the joint positions retrieved from the Kinect.

4.5 User interface

The user interface design was an essential part of the project since the interaction with the application is quite different to a 'normal' application that can be controlled with a computer mouse. Instead, other approaches to interact with the application have to be considered. The reason for this is not only the fact that a third dimension is added compared to the computer mouse, but moreover the Kinect allowing new ways of interaction that can involve all joints of a human body. This thesis takes a closer look on some different techniques that can be applied, for instance:

- Gesture recognition
- 2D graphical user interface
- 3D interaction elements

4.5.1 Gesture recognition

The first technique that I was implementing was a gesture recognition algorithm. The idea was based on a swipe gesture, furthermore distinguishing between a left and a right swipe. A simple swipe gesture would therefore change cloth, for example. The algorithm was based on a formula that is calculating the angle between the line defined by the 2 consecutive points *(a, b)* and the horizontal axis:

$$\alpha = atan(\frac{b_{t+1} - b_t}{a_{t+1} - a_t})$$

By comparing a set of calculations of the angle to a specific pattern (e.g. if most of the calculated angles of consecutive hand motions lie in between -20 to 20 degrees a swipe to the right side has been executed by the user), it becomes possible to identify gestures. Furthermore it may be necessary to limit the execution to a certain time in order to exclude wrong gestures.

Though this method is quite useful for other purposes, it generated lots of errors during interaction for a user of the try-on application. The cause for that is the characteristic behavior performed by the users during the try-on of clothes. If a user was simply holding up his hands and taking a look at clothes from the right side and then turned his body and therefore the hands a bit to the left so that the user can take a look from the other side, the algorithm would have recognized a swipe gesture to the left. Since dealing with different parameters and adjusting the algorithm to ignore certain positions did not have the desired effect, other kinds of interaction had to be considered.

4.5.2 2D graphical user interface

Taking a look at comparable virtual fitting rooms utilizing the Kinect, I discovered that some of them were using 2D GUI elements. In some similar projects, the mouse was replaced with the coordinates of the hand using projection to convert the three-dimensional points into two-dimensional coordinates.

On the one hand this is a huge advantage since a lot of choices can be presented to the user at one time. On the other hand, customers that are not technically experienced wanting to simply try on clothes may be overwhelmed by the user interface. Another downside is the fact that the arm of the user has to be pushed far to the side, since the 2D elements are positioned on the outer frame.

4.5.3 3D interaction elements

Therefore the consideration was emphasized on 3D elements that are integrated into the Virtual Dressing Room in an augmented reality like manner. The basic idea of a concept was an interaction element that is always next to the customer. Moreover these elements should allow a person to quickly switch trousers, shirts and even furthermore change the actual background of the Kinect camera stream to try on clothes in different environments. More precisely this means that the operational elements are overlaid on the mirrored screen and are moving in respect to the position of the current user of the Kinect. To hide all the mentioned elements, a 3D button was

introduced, in order for the user to be able to take a closer look at the garment without disturbing elements around him.

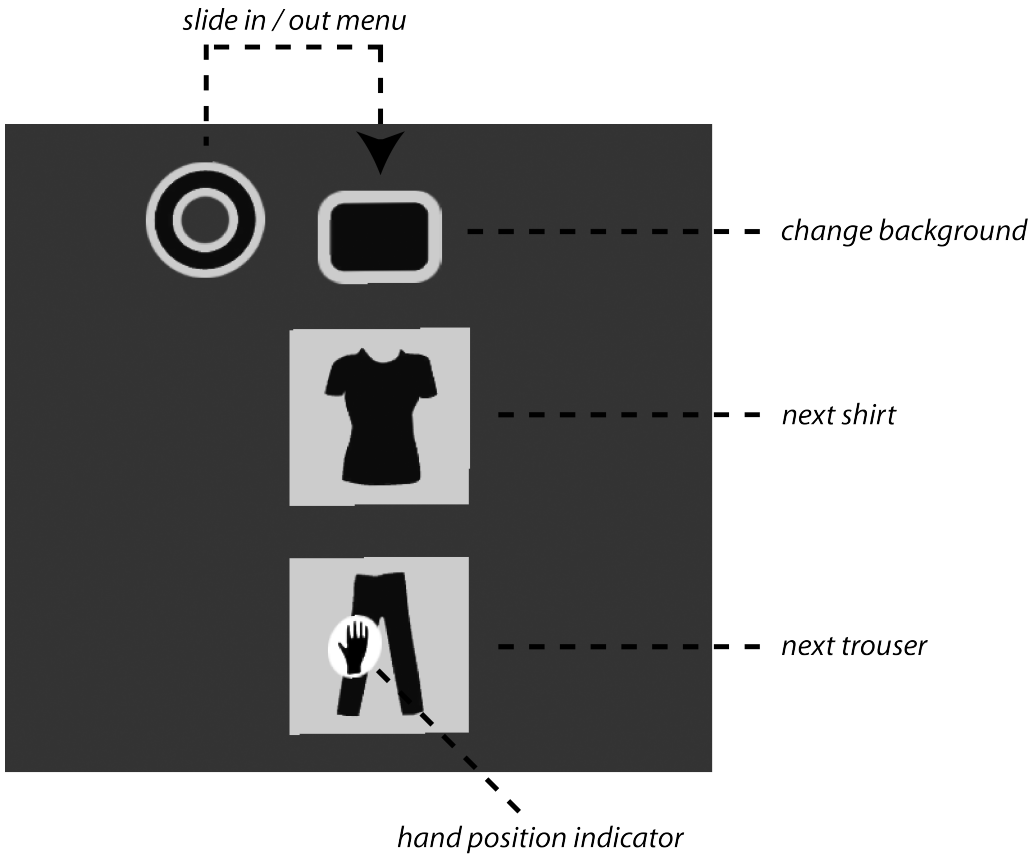

Figure 4.6: The user interface elements of the Virtual Dressing Room are depicted in this figure. The button on the upper left side of the illustration is turning the menu on the right side on or off. The other elements are functions for changing of background image, and the selection of the next clothes. An hand position indicator is showing the current coordinates of the hand on the screen similar to a mouse.

Figure 4.6 is illustrating the overall look of the three-dimensional user interface. If an user is sliding over the UI elements, an indicator is illustrating the current hand position similar to a cursor of a computer mouse on the screen. By moving the hand over the menu activation button on the upper left side of the image, the remaining control elements of the menu on the right side of the activation button appear. This menu allows an interaction with the Virtual Dressing Room, i.e. changing background, switching to the next shirt or to the next trouser. The desired action is executed by leaving your hand above the button for some seconds. This is indicated by a color fade of the button. This allows an easy way of interaction by just holding your hand above the element and waiting for the application to switch from one cloth to the next one.

The advantage of this method is the quick access to all of the essential functions that are needed in order to interact with the Virtual Dressing Room. Another benefit is the fact that all operational elements are positioned next to the body of the person and a user only has to move his hand to perform an action. Also no instructions have to be given to the user since the interaction with the elements is quite self-explanatory. Besides, the slide in/out menu button allows a customer to take a closer look at a shirt without being interrupted by the remaining buttons around him.

4.6 Debug functions

The keys described in Table 4.1 are defined for debugging functions. They can be changed during the start up dialog of the Virtual Dressing Room. Figure 4.7 is providing an overview of the mentioned functions.

Key	Alternative Key	Description
1	F1	Show Debug Information (e.g. framerate, tracked users, etc.)
2	F2	Display the Cloth Colliders
3	F3	Draw the Skeleton of the User
escape	q	Quit the Application

Table 4.1: Standard settings and descriptions for the debugging keys.

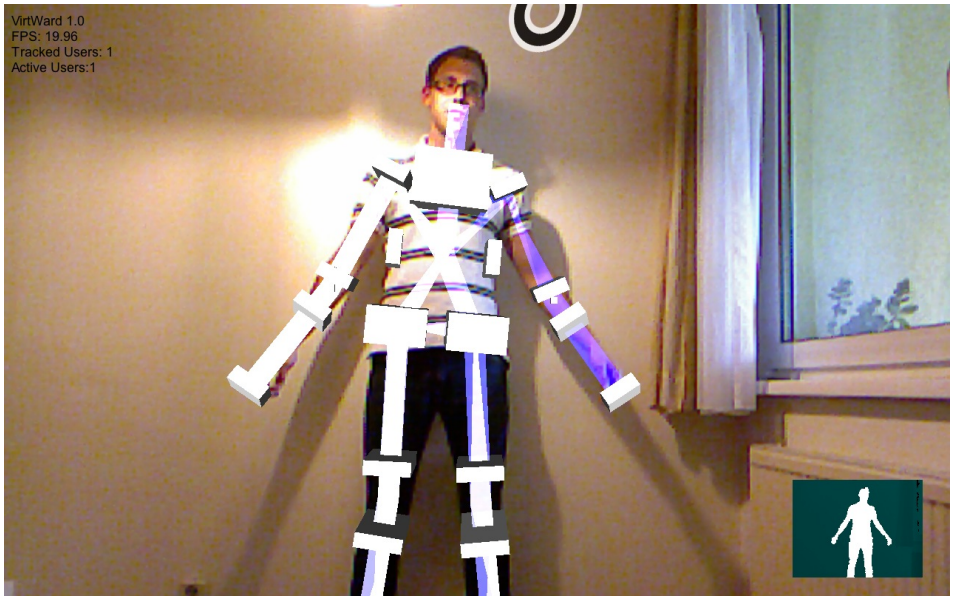

Figure 4.7: The debug output of the Virtual Dressing Room. The line (changing color from white to blue) is connecting all joints of the skeleton, the cloth colliders are positioned on the body. Further debug information is displayed on the top left. The depth image on the bottom right is illustrating the currently tracked users in white color.

CHAPTER 5

Implementation

The following sections will provide an overview of the implementation of the Virtual Dressing Room. After the class structure has been specified, an insight on the configuration of the scene in Unity is given and the `XML` settings of OpenNI are specified. Besides the implementation of the outputting streams is illustrated and the tracking procedures are explained. Furthermore all aspects dealing with garment like adaption and positioning as well as the associated cloth physics and the structure of the `XML` file containing the garment settings are presented.

5.1 Class structure

The class hierarchy of the Virtual Dressing Room is illustrated in Figure 5.1. An overview of the class structure illustrating the class names and their functions and furthermore the connections to each other is presented in this subsection.

KinectController This is the main class of the Virtual Dressing Room. It is maintaining the connection to the Kinect (i.e. updating the depth/image stream, controlling the joint positions, etc.) on the one hand and is executing functions of classes that are related to the dressing room (i.e. cloth operations, collider positioning, user interaction, etc.) on the other hand. The class derives from `MonoBehaviour`.

ClothCollider The `ClothCollider` class is responsible for creating and updating the colliders that are positioned all over the body. It is also setting the parameters for the cloth physics and is responsible for loading and unloading the pieces of garment.

XMLParser A static class that is loading the `XML` file which is holding all the parameters (e.g. name, price info, cloth physic settings, etc.) for each individual cloth.

ClothSetting The loaded parameters from the `XML` file are stored as a `ClothSetting` object for further use.

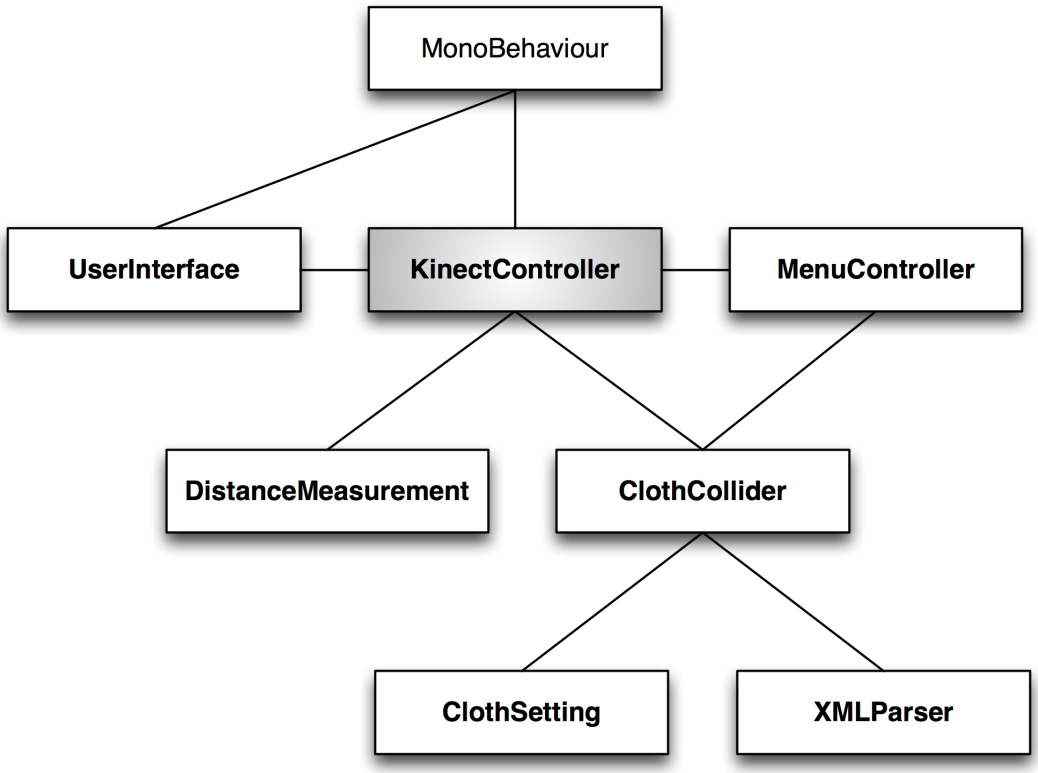

Figure 5.1: The class hierarchy of the application. The `KinectController` class is the main class, which is deriving from the Unity base class `MonoBehaviour`. All other classes have their specific functions as described in this section.

MenuController The `MenuController` class is the main class dealing with the user interaction. It is controlling the three dimensional elements of the user interface and is placing them next to the position of the user in front of the Kinect.

DistanceMeasurement This static class is responsible for measuring the distances that are not covered by the joint positions (i.e. the waist size).

UserInterface The `UserInterface` class is controlling all elements of the 2D GUI. These are mostly used for the purpose of debug output. The `UserInterface` class derives from `MonoBehaviour`.

The main class of the Virtual Dressing Room is the `KinectController` class. It is responsible for maintaining the connection to the Kinect, the processing of the streams, reading out the skeleton information and moreover acting as an initial point for further operations and object initializations. The `KinectController` class derives from the base Unity class

MonoBehaviour in order to execute essential functions such as Start(), Update() or FixedUpdate().

The KinectController class is creating an instance of the ClothCollider class. The ClothCollider class is focusing on the cloth colliders that are needed for the purpose of garment interaction. It is also responsible for loading and unloading of the clothes. Furthermore it is using the static class XMLParser to load the corresponding parameters for the pieces of garment, which are parsed from an XML file. The parameters are stored as a ClothSetting object.

The static DistanceMeasurement class is calculating the distance of body joints that are not delivered by the standard joint position set which is in the case of this application the waist size.

To cover the interaction with users and therefore the three-dimensional menu elements, the MenuController class is introduced. The KinectController class is creating an instance of it and is refreshing the position and orientation based on the location of the user in front of the Kinect. When an action is triggered by the user (e.g. switching to the next shirt, changing background) the corresponding method is executed in the KinectController or ClothCollider instance.

The UserInterface class is responsible for the two-dimensional elements of the user interface implicating the debug output as well as the active user count or the frame rate. It is deriving from MonoBehaviour since it has an own Update() function that is calculating the frame rate as well as a Start() method for initialization purposes and an OnGUI() function for rendering GUI events.

5.2 Unity scene configuration

This section deals with the composition and layout of the scene in Unity. The scene is consisting of the following elements:

Cloth_Female The three dimensional body of the human cloth model that was used for modeling the pieces of garment. The positions of the ClothColliders are also based upon this GameObject.

Cloth_Objects This is a GameObject that is containing all garment models. Every object that is a child of this GameObject can be loaded and used as a cloth during runtime.

Plane The plane that is positioned in the background. It is either showing the video stream recorded by the Kinect camera or displaying a different setting that is chosen by the user. When the background wants to be changed the texture is replaced with a static image and the UserPlane is then used for extracting and displaying the user in front of the Kinect.

UserPlane A textured plane that is only used when the background of the scene is exchanged with an image. It is then displaying the user separated from the background and is positioned in front of the Plane.

DepthPlane A plane assigned with a material and a texture that is displaying the depth map received from the Kinect and additionally the user in white color. It gets updated every frame.

Directional light A directional light source that is illuminating the scene.

Point light A point light source lighting up the surrounding area.

Infotext A 3D text mesh used to display the info that is stored in the `XML` file.

KinectController The main controller for the Kinect. It is maintaining all connections to various `GameObject` entities and to the transforms of the skeleton joints.

Main Camera A perspective camera that is capturing the scene that will be displayed to the user.

Menu This is the parent object, containing the elements for the three dimensional menu responsible for the user interaction.

Torso This is the parent `GameObject` used in the `KinectController`. All other joints (ElbowL, ElbowR, FootL, FootR, etc.) which are `Transform` elements are children of the torso.

5.3 Controlling the Kinect

The parameters for OpenNI can be specified with an `XML` file that is loaded during startup. The file is used for configuring the streams, for instance. These values can determine logging, mirroring, the resolution of the depth and RGB camera, the FPS (frames per second) and some other parameters. At the beginning the license and the appropriate keys are specified. Then the value of the `LogLevel` is set to 3, which means it only outputs errors (i.e. no warnings or info). Additionally mask or dump elements can be specified. Since we need to mirror the output stream in order to achieve a correct mirroring of the scene this value is set to true for the image and the depth stream. Furthermore the image output is set to the highest possible resolution for maximum quality. The exact configuration settings that are applied for the Virtual Dressing Room are the following:

```
<OpenNI>
 <Licenses>
  <License vendor="PrimeSense" key=
  "0KOIk2JeIBYClPWVnMoRKn5cdY4="/>
 </Licenses>
 <Log writeToConsole="true" writeToFile="false">
  <LogLevel value="3"/>
  <Masks>
   <Mask name="ALL" on="false"/>
  </Masks>
```

```
  <Dumps>
  </Dumps>
 </Log>
 <ProductionNodes>
  <Node type="Image" name="Image1">
   <Configuration>
    <MapOutputMode xRes="640" yRes="480" FPS="30"/>
    <Mirror on="true"/>
   </Configuration>
  </Node>
  <Node type="Depth" name="Depth1">
   <Configuration>
    <Mirror on="true"/>
   </Configuration>
  </Node>
  <Node type="User" />
 </ProductionNodes>
</OpenNI>
```

The `KinectController` class is handling the connection between the computer and the Kinect device. OpenNI and NITE are initialized by creating a new `Context` using the `XML` file to set up the parameters. After a `DepthGenerator` and an `ImageGenerator` are created to process the depth and video image, the application is dealing with the skeleton/user detection. This includes the `UserGenerator`, which is used for event subscribing (`NewUser`, `LostUser`), a `PoseDetectionCapability`, which is detecting the calibration pose, and a `SkeletonCapability` responsible for ending the calibration process and starting the skeleton tracking process. To tell the production nodes to begin generating the data, the function `StartGenerating()` is called on the `UserGenerator`.

The method that is refreshing the application is Unity's `Update()` function, also responsible for updating the context using the `WaitOneUpdateAll(depth)` method. In this case that means that the application is waiting for the node (here: depth, the `DepthGenerator`) to generate new data and is then updating every node.

5.4 Reading and outputting streams

The scene is captured by a video camera as well as a depth sensor, but the two streams are not exactly aligned due to the spacing between both units. Therefore we have to match both streams and align them to each other. This can be done using calculations implicating the offset, or by using a function provided by OpenNI to set the alternative viewpoint from the depth to the image generator: `SetViewPoint()`.

To render the depth as well as the image stream, materials are placed on the planes and the material's textures are refreshed using the `Update()` function. The resolution of the textures is based on the values of the image/depth output. Since resolutions that are based on a power of two

are faster processed by the GPU, the function `GetNextPowerOfTwo()`, which is calculating this value as suggested by a forum post [18], is implemented. This will have a positive impact on the performance.

Furthermore also a bug during the debayering-process, which has a strange effect on the image output offsetting the pixels of some rows, is fixed using the corrected script file [29]. The problem of debayering can be seen on the edges of the objects in Figure 4.5.

After all necessary variables and textures are initialized using the `CreateTextures()` method, the `UpdateDepthMap()` and the `UpdateImageMap()` functions can be called during runtime. These are filling an array of `Color` with the appropriate values and are then setting the pixels of a `Texture2D` that is applied on a `Material` which is assigned on a plane. To further separate a user from the background the `UpdateUserMap()` method is called, which is refreshing a texture map on the `UserPlane`. To extract the user from the rest of the image, the user map is taken and combined with the image map, setting all values where no user is found (user map value equals 0) to an alpha value of 0. When using this method, the background of the Virtual Dressing Room is replaced with a static image (cf. Figure 6.5).

5.5 Tracking

If a user is correctly tracked, which is checked using the `IsTracking()` function that is returning a boolean value for a `userid` that is representing the user, he/she can now interact with the Virtual Dressing Room and continue with the try-on procedure. The joint positions are then refreshed during the `Update()` method and are assigned to the created `Transform` elements depending on the `SkeletonJoint` position (e.g. `Torso`, `RightHand`, `LeftHand`, etc.) according to the `userid` of the tracked person.

A problem caused by OpenNI is the timeout counter that is set to 10 seconds. More precisely this means, that after a person has left the scene the next person that enters to interact with the virtual fitting room has to wait this time in order for the system to delete the old user and register the new one. Therefore the Virtual Dressing Room application maintains a separate list of users to bypass this timeout. By accessing the center of mass of a user, using the `GetCoM()` function with the `userid` as parameter, it becomes possible to detect users that have left the area in front of the Kinect. That means by knowing that a center of mass (CoM) of a user with parameters `CoM: X, Y, Z == 0` is equivalent to a user that has left the scene, the user can then be manually removed from the corresponding list.

5.6 User interfaces

This subsection focuses on the three-dimensional interaction elements for the user in front of the Kinect and the two-dimensional user interface in order to display the calibration pose texture and the debug output.

5.6.1 3D interaction elements

The main class which is responsible for user interaction is the `MenuController` class. The `MenuController` is generated in the `Start()` method of the `KinectController`, and refreshed in its `Update()` function.

The `Refresh()` function of the `MenuController` takes the desired menu position based on the location of the user and the position of the hand as an argument. If the position of the hand is inside the bounds of the three dimensional planes, only comparing the `X` and `Y` values, the desired action is executed. For the menu activation button this is immediately executed, rotating in to show off the menu until the normal vectors are facing towards the user. Taking a closer look at the other buttons also a timer is used during which the positions and the colors (with `Vector3.Lerp()` and `Color.Lerp()`) are changed in order to better visualize the pressing of a 3D button.

Besides, the application is also showing a hand icon to display the current hand position.

If a button is pressed and therefore the current piece of garment wants to be changed, either the function `NextShirt()` or `NextTrouser()` of the `ClothCollider` class is executed in order to load the next piece of garment. A closer explanation of these methods is given in the garment section.

5.6.2 2D user interface

To deal with the 2D GUI elements of the application, such as the displaying of the debug information (e.g. frames per second, active users) or the texture of the calibration pose, the `UserInterface` class is maintained. This class mainly consists of an `Update()` function, responsible for the calculation of the framerate, and Unity's `OnGUI()` method, which is outputting the information.

The debug information considering the drawing of the skeleton - more precisely superimposing the skeleton over the user's body - is realized in the `KinectController` class. It is utilizing Unity's `LineRenderer` component for this purpose.

5.7 Garment

Now we will take a closer look on the steps dealing with the adaption of the garment to specific bodies, taking body measurements, the loading procedure including the parameters for the cloth physics, which are the most essential aspects of a virtual try-on.

5.7.1 Garment adaption

Garment adaption is essential in order to accurately fit a cloth to a specific human body. The part of the application that is, amongst other things, handling the positioning of the colliders, is the `ClothCollider` class. It is initialized in the `Start()` method of the `KinectController` class. Besides taking care of the orientation of the cloth colliders, the class is also responsible for their initialization as well as the loading and unloading procedures for all pieces of garment.

When creating a new `ClothCollider` object all colliders are created. These colliders are `GameObject` primitives, in this case cubes (in Unity: `PrimitiveType.Cube`). For every necessary spot on the left and right side of the body these colliders are created and are getting updated during runtime. Furthermore a transparent material is assigned to all of the colliders and they are moved to their initialization position by using the function: `CollidersToInitPosition()`.

The method `CollidersToInitPosition()` is one of the main functions considering garment adaption. First of all, the colliders are positioned based on the joint positions of the three dimensional body illustrated in Figure 4.3. Taking a closer look at the arm, standard joint positions for the left/right hand and the left/right shoulder are defined. The left/right elbow has to be treated differently. By defining only 1 collider, the long-sleeved shirt would bend too strong causing issues like folding the polygons together on the forearm and the upper arm. Therefore 2 colliders are defined, one placed near the elbow on the forearm, the other one placed near the elbow on the upper arm. The following formula is used for calculating the new positions:

$$p_{new} = x + t * dist(x, y) * (y - x).norm$$

The formula is calculating the new joint position values based on the two positions x and y, which are representing the hand and the elbow, for instance. The value t is describing the distance as percentage between the two positions where $0 \leq t \leq 1$. It is also applied on the left and the right knee.

The same operation can also be executed easier using Unity's `Vector3.Lerp()` function, which is taking a start point and an end point as input as well as a value that is used for interpolation between the two points. The function is then returning the interpolated point (as a `Vector3` structure).

Depending on the position of the cloth colliders also different scaling values of the cubes serving as colliders have to be taken into account. Therefore a scaling is realized by using the `Transform.localScale` property, assigning a `Vector3` object as a scaling factor.

At last, the rotations of the cloth colliders have to be adjusted as well. This is done by using the `Transform.LookAt()` method of Unity, which is taking the world position (the LookAt position) and an upwards vector as parameters.

During runtime, the `FixedUpdate()` method of the `KinectController` class is setting the positions and rotations of the cloth colliders. To achieve this, the `FixedUpdate()` method has to be used in order to stay in sync with the physics engine of Unity, since this function is executed differently depending on the current frame rate in comparison to the normal `Update()` function. In the `FixedUpdate()` function, the different orientations of the joints are passed to the `ClothCollider` object, which is positioning the colliders based on the data. Finally, `UpdateColliders()` is run, updating the rotation values of the cloth colliders.

5.7.2 Body measurements

The middleware is providing many joint positions that can be used in order to compute body measurements. The space between two arbitrary joints can be calculated using the standard

euclidean distance formula which outputs the result as the distance in meters if the joint positions are based on real world coordinates.

However, using only joint positions as already mentioned to get the measurements of a person has limitations since certain dimensions as the waist size (or not used for this application: the arm or foot width) can't be determined. Therefore these values have to be calculated. This is achieved by executing the `Calculate()` function of the `DistanceMeasurement` class, which is evaluating the distance based on the map that is segmenting the user in front of the camera from the rest of the scene. Therefore the parameters for the `Calculate()` method are an array of colors as well as the resolution of the map representing the user, the current size of the `UserPlane` and a start point as well as an end point and another start and end point for the reason of comparison. The distance is measured perpendicular in the middle of the line that is formed by the start and the end point. By simply using a while loop that is counting the pixels until the alpha value changes to zero, which represents the end of the user map, and comparing the pixel values to the real value it becomes possible to measure an approximate distance, though the results can differ a bit in comparison to the real values.

Taking a look at the waist, the start point would be the position between the shoulders and the end point would be the position between the hips. In the middle of those two points, the pixels of the user map are counted perpendicular to the line between the start and the end point. This results in the pixel distance of the waist. To get the real distance of the waist, we take a known value, for example the distance between the left and the right hip. We can now calculate the according pixel distance between the left and the right hip. Since we are knowing the pixel size of the waist and the hip, and the real distance of the hip, we can easily calculate the real distance of the waist as well.

The whole procedure is executed shortly after the calibration process that is realized by the `KinectController` class. Subsequently the waist size is calculated and the appropriate colliders are positioned based on the result. The correct positioning of the colliders at the waist can be seen in Figure 4.7.

5.7.3 Loading procedure

If a user is selecting the designated cloth in the user interface, the Virtual Dressing Room is loading the appropriate model from the `Cloth_Objects` `GameObject`, which is the parent of all pieces of cloth, in Unity. All garment models that are intended to be loaded during runtime have to be stored as a child in the `Cloth_Objects` hierarchy. The application is able to find every piece of cloth according to the name in the corresponding `XML` file and can then load the appropriate parameters.

The next steps are executed when switching clothes: First of all, the `InteractiveCloth` as well as the `ClothRenderer` component are assigned to the `GameObject` that is representing the garment model. Secondly, the settings defined in the `XML` file, especially the type and the options, are taken into account in order to set the right physic settings and also assign the cloth colliders that are holding the cloth on the correct positions. Finally, the cloth colliders are moved to the skeleton of the user. In particular the cloth colliders are positioned in respect to the joint positions of the skeleton or depending on measurements in case of the waist size. This is stretching the cloth and furthermore adapting it to the body of the particular person in front of

the Kinect due to the cloth colliders. Starting from now the cloth colliders can move the cloth arbitrarily in front of the body of the user.

The following steps will give a detailed explanation of the procedure.

To load the next shirt, the method `NextShirt()` of the `ClothCollider` class is called. First of all this unloads the current shirt by using the `UnloadCloth()` function. Secondly, it searches for the name of the next shirt in the `clothSettingsList`, and finally loads this shirt using the `LoadCloth()` method. The same applies to trousers, which are calling the `NextTrouser()` function.

The `LoadCloth()` method is responsible for a major task: the loading of the specified `GameObject`, which is handed over as a parameter. Since the `GameObject` is a piece of garment that needs to be positioned according to the cloth collider positions of the three dimensional body model, the initial parameters for the orientation of these colliders have to be determined. Furthermore they have to be attached to the cloth before the following steps can proceed. The `InteractiveCloth` component of Unity forms the basis for this. After this component is added to the specified piece of garment, the application is selecting the mesh of the garment model and moreover assigning it to the `InteractiveCloth` element in order to define this mesh to be used for cloth simulation.

Next, the `CollidersToInitPosition()` function is called. The function is responsible for setting the position and the orientation of the cloth colliders, based on the human body model. After the colliders are positioned they are added to the array of attached colliders of the `InteractiveCloth` element. The vertices of the mesh are now static inside of these colliders. The remaining vertices are behaving according to the `InteractiveCloth` settings.

In the following step, the `Type` and `Option` settings of the `XML` file are checked to attach the cloth to the belonging colliders, depending if it's a shirt or trouser and a long or short version. Thereafter the `ClothRenderer` component of Unity is added to the garment that is used in combination with the `InteractiveCloth` element in order to render the physic effects, hence the standard renderer of the cloth is disabled. In a last step the physic settings of the `InteractiveCloth` are set to the defined values of the `XML` file.

An `UnloadCloth()` function is responsible for unloading the garment when switching to the next piece of cloth. It is taking a `GameObject` as parameter and is destroying all components attached to a piece of cloth.

5.7.4 Cloth parameters and XML structure

Cloth physics are representing a main aspect of realism of the Virtual Dressing Room. The interactive physics show the most effect on skirts since these are the ones that usually have the maximum distance to the body. Moreover, some parts are not effected by cloth colliders.

Using Unity, `InteractiveCloth` allows to set specific parameters that are specifying the behavior of the garment. Different attributes that are most important for cloth physics are bending stiffness, stretching stiffness, damping and gravity, for instance. For the virtual dressing room an `XML` parser is used in order to define the settings for every piece of garment individually. This offers a huge advantage since it allows an easy access and modification of the particular parameters. If a new garment model is added to the hierarchy in Unity, it needs no more steps than adding additional elements to this file. This makes it really easy to model further clothes

based on the female body model, drag them into the hierarchy and add the corresponding `Name` of the `GameObject` and all the necessary parameters of the two components: the `General` and the `InteractiveCloth` part.

A sample structure of this `XML` file (for one piece of cloth) is listed below.

```
<?xml version="1.0"?>
<!-- Settings for Clothes -->
<Clothes>
   <Cloth>
      <General>
         <Name>Trouser_1</Name>
         <Type>Trouser</Type>
         <Options>short</Options>
         <Info>99.- EUR</Info>
      </General>
      <InteractiveCloth>
         <Thickness>0.1</Thickness>
         <BendStiffness>0.1</BendStiffness>
         <StretchStiffness>0.1</StretchStiffness>
         <Damping>0.001</Damping>
         <Pressure>1.0</Pressure>
      </InteractiveCloth>
   </Cloth>
</Clothes>
```

As depicted in the listing, `General` settings that are related to the cloth as well as parameters that are serving as input for the `InteractiveCloth`, have to be specified. In the following, the input values for the `XML` file are stated in detail.

The `General` part is providing settings that are related to the cloth and are not associated with the `InteractiveCloth` component:

Name The name as a string of the `GameObject` in Unity that is representing the cloth object. The system finds the corresponding cloth according to this string.

Type The type of cloth, currently there are two types of cloth supported: *Trouser*, *Shirt*. It has to be specified in order to attach the appropriate cloth colliders to the `GameObject`.

Options Additional options that are also effecting the cloth colliders. The two settings describing the length of a shirt or trouser are *long* and *short*. A t-shirt would have the *short* attribute, for instance.

Info This is usually representing the price info, although other infos can be added as well. The string will be displayed after a cloth is selected. This value can also be left empty.

The `InteractiveCloth` settings are related to the `InteractiveCloth` component assigned to the garment. This provides settings to adjust the physical parameters of the cloth in order to react to movement in a realistic way.

Thickness This attribute defines the thickness of the hull of the cloth. The thickness value is affecting the other parameters as well.

BendStiffness The bend stiffness setting of the cloth. The value lies between 0.0 (disabled) and 1.0.

StretchStiffness The stiffness that is affecting the stretch value. The stretch stiffness property has a value between 0.001 and 1.0.

Damping The damping of the cloth physics. Damping property starts from 0.0 up to 1.0, though 0.0 is disabling damping.

Pressure This setting is defining the pressure that is built up inside of the cloth. A value of 1.0 means that the same pressure is applied on the inside that exists on the outside, 0.0 is disabling this parameter. Usually a value of 1.0 seems to fit best. A significantly larger value sometimes causes few polygons of the model to flicker, although values up to 10000 can be entered.

Parallel to the cloth the stated parameters are loaded and assigned to the interactive cloth component during runtime after a piece of garment has been selected.

In order to readout the elements of the `XML` file, the application is accessing the static `XMLParser` class. It provides the function `ReadXMLClothData(path)`, taking the path (a string) as an argument. If the `XML` file is correctly formatted, all parameters of one cloth are stored as as a `ClothSetting` object. The `ClothSetting` class is consisting of strings and floats, that is storing the values (i.e. Name, Type, Thickness, etc. - settings of the garment) in an object format. The parser is reading the XML data once by using a `while` loop and is then returning a `List` of `ClothSettings`. This procedure is executed when the `ClothCollider` class is loaded.

CHAPTER 6

Tests and Results

In this section I will take a look at the tests and the results of the Virtual Dressing Room. In the first section I am going to provide a short presentation of the achieved results, illustrated with some screenshots. The second section will take a look at the feedback received from the users concerning their experience with the Virtual Dressing Room by evaluating questionnaires.

6.1 Results

During the testing stage special attention was paid to all of the implemented functions and how they behave on different participants.

First of all the focus is set on the individual adaption of clothes since the dressing room should be usable for persons of varying measurements independent of their waist length, their arm length or their leg length. As already stated, the cloth colliders are positioned on the particular spots to simulate this behavior.

Furthermore also an overall look on the mesh representing the cloth and on the belonging texture is taken. The clothes are especially designed for women since the garment meshes are based on a female person. Due to the fact that the clothes are adapted to particular bodies they also fit for men. Besides, a glance is taken on the physics and their particular reactions on the body. Also the correct separation of the participants in front of different environments is considered.

After the persons have used the Virtual Dressing Room a questionnaire was handed over to them. The given answers are then discussed in the next chapter.

Overall the application is running at a stable frame rate of around 22-25 frames per second on a MacBook on Mac OS X 10.6.8 with a 2.4 GHz Intel Core 2 Duo, 4 GB of DDR3 RAM and an NVIDIA GeForce 9400M graphics card. The resolution of the application is set to *1280 x 720* on a flat screen TV (16:9) or *1280 x 800* on the MacBook screen both in fullscreen mode, the graphics quality is set to *good* (i.e. primarily disabled antialiasing). These settings are defined in

the configuration dialog before startup and are delivering proper results with an acceptable frame rate. Compared to the same machine running Windows 7, the identical application is running just around 10 frames per second. The exact reason for this behavior is unclear. Nonetheless this is below a meaningful frame rate and therefore all the tests have been executed on Mac OS X. Figure 6.1 is illustrating the standard startup dialog.

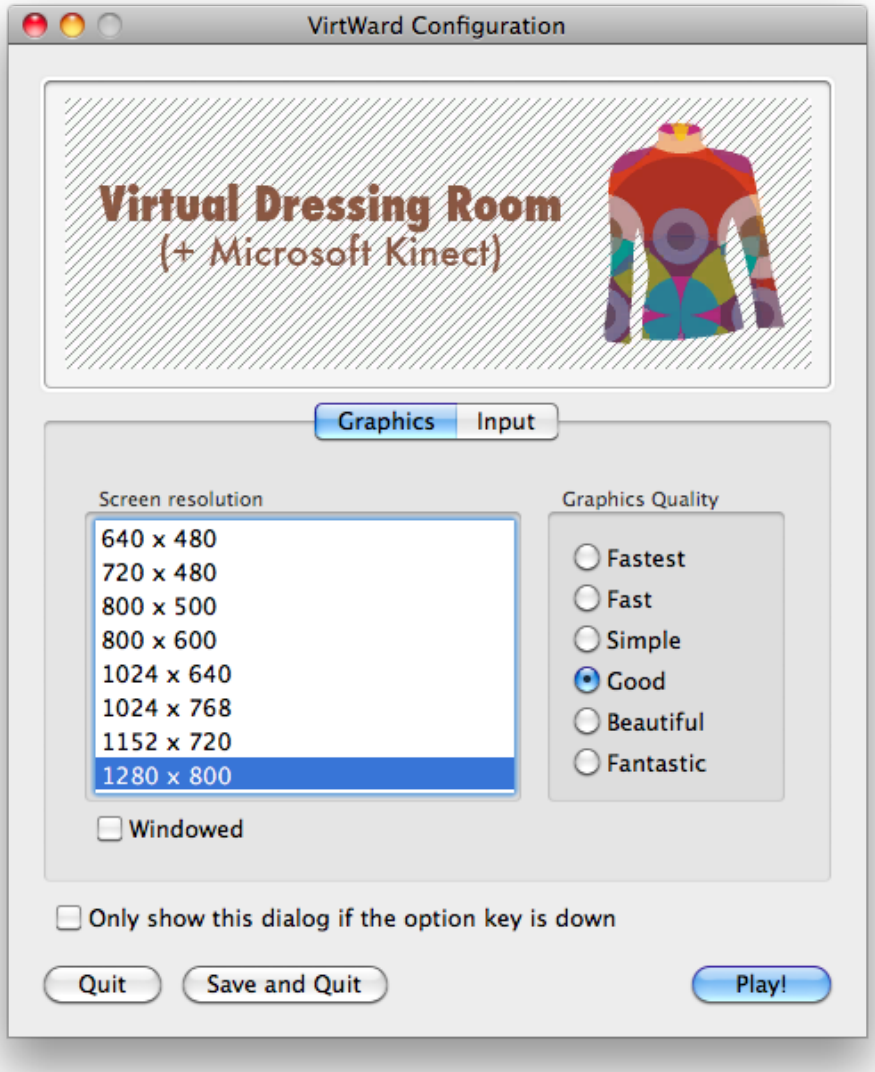

Figure 6.1: The configuration dialog appearing after startup with the screen resolution and graphics quality both set to proper values.

The texture in the middle of Figure 6.2 is shown every time a new user is entering the area in front of the Kinect. It is prompting the user to perform the calibration pose.

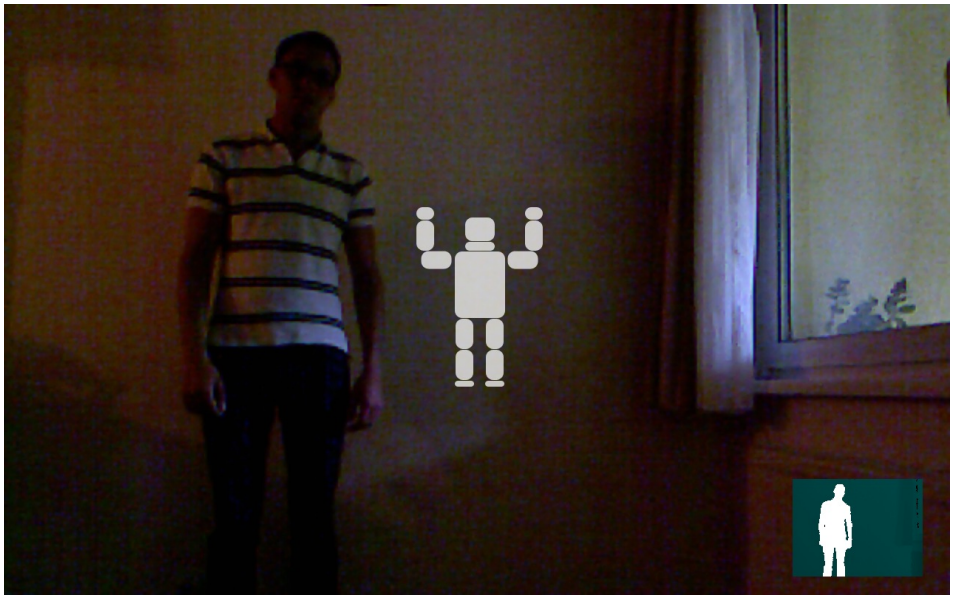

Figure 6.2: A screenshot that shows the running application. If a person has entered the area in front of the Kinect, the user is recognized and a flashing illustration that is prompting the user to change into calibration pose is shown. On the bottom right is the depth map that is showing the current user in white color.

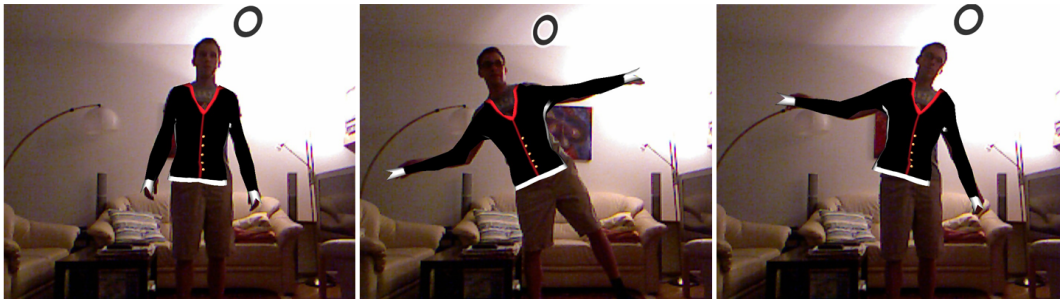

Figure 6.3: Some screenshots that were randomly taken during the testing of a long shirt. The user is trying several positions to demonstrate the behavior of the garment. Depending on the position of the cloth colliders the adaption works really well.

Of course, the application also has to take into account special positions of the body and of the extremities like the hand or the feet, which have to be be handled by the Virtual Dressing Room as well. As already mentioned in the design chapter of this thesis, a special treatment of the knee and the elbow is needed in order to behave accordingly. Figure 6.3 is presenting

Figure 6.4: The behavior of physics can be seen quite good on the skirt which is swinging when the person in front of the Kinect is moving up and down.

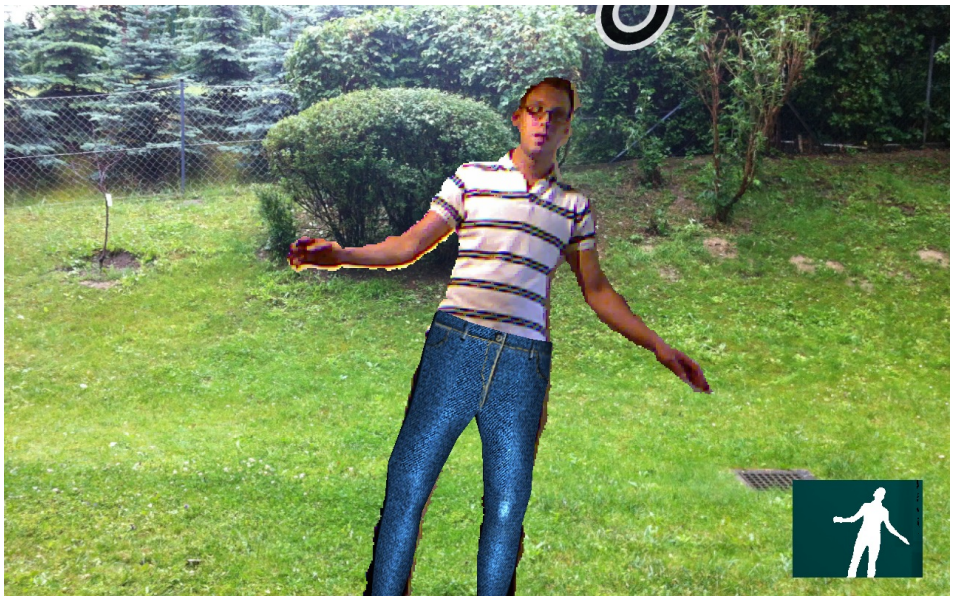

Figure 6.5: The figure is showing a scene with an exchanged background. The user is separated from the rest of the image and drawn in front of a new background with superimposed trousers.

an overview of some special cases that were randomly produced during the tests of the Virtual Dressing Room.

Another illustration is depicting the behavior of the physics that are applied on the garment.

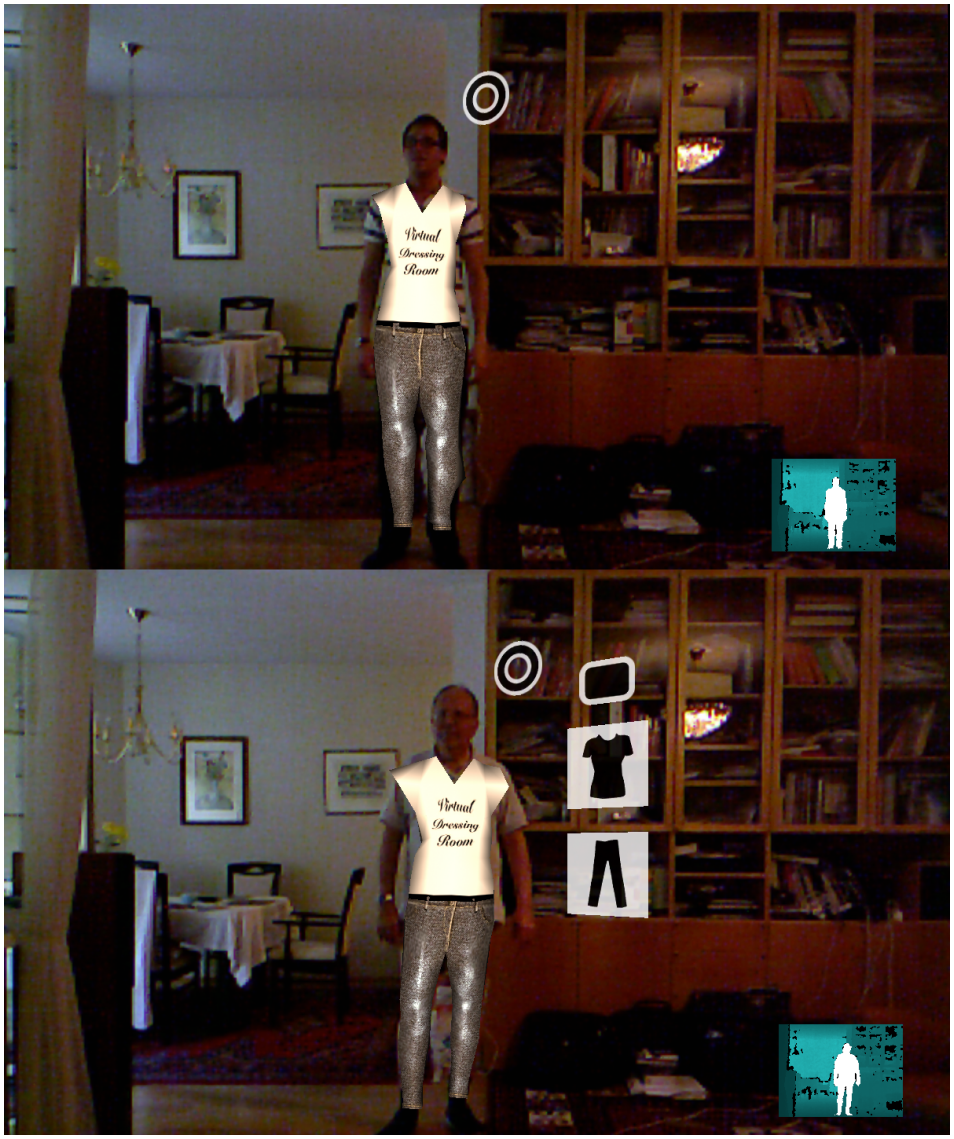

Figure 6.6: A comparison between the varying treatment of the garment on diverse persons. You can clearly see how the cloth is adapted differently depending on the particular body measurements. In the lower figure the garment is stretched even more in comparison to the upper one. The lower illustration also depicts the menu elements that are always next to the person to switch between the shirts, the trousers or to change the background.

You can see the skirt that is realistically reacting to the movement if a person is moving up and down in Figure 6.4. Sometimes customers also want to take a look not only on the front, but also on the side of the cloth. This has to be considered as well. Figure 6.7 shows the movement of a skirt if a person is turning to the side, for example.

Figure 6.5 is presenting different background images to simulate the fitting of clothes in different environments. Furthermore the image shows that the separation of the body from the rest of the image works out quite well. Certainly some bright borders especially on the right arm of the person can be seen. They occur sometimes since the depth image resolution is quite low to achieve a 100 percent correct separation.

Figure 6.6 is showing off a comparison of the same dresses on different persons. As shown in the illustration, both colliders are adapted individually to the specific sizes of the particular human body. The screenshot clearly shows that the cloth of the upper person is stretched less to the side than the garment of the lower person. The pieces of cloth are perfectly aligned to the interacting person.

Figure 6.7: The behavior of the physics engine and a view from the side can be seen here. Furthermore the problem that the hand can not be rendered in front of the body is illustrated since we are drawing the 2D texture of the person on a simple plane.

6.2 Feedback

The users were given a questionnaire to answer some questions regarding the Virtual Dressing Room after they have tested it. First of all, all participants tested the Virtual Dressing Room on a standard setup. Special attention was paid to a large screen for testing purposes. No advice or introduction was given since it should be self-explanatory. Even if it is built up in a cloth store, no explanation can be given to every single user. After the participants became familiar with the interaction part they began to choose different pieces of garment. In a next step the participants took a closer look on the clothes of the Virtual Dressing Room and the various features (e.g. changing the background). At last, a questionnaire was handed over to complete. The response was measured with the Likert scale and the questionnaire consisted of the questions specified in Table 6.1 (questions translated in English).

	+ +	+	+/-	-	- -
Are you interested in fashion?	☐	☐	☐	☐	☐
Do you think a virtual fitting room can be helpful?	☐	☐	☐	☐	☐
How well was the overall appearance of the virtual clothes?	☐	☐	☐	☐	☐
Did you like the aspect that clothes automatically fit to your body?	☐	☐	☐	☐	☐
Was the interaction with the Virtual Dressing Room intuitive and easy to understand?	☐	☐	☐	☐	☐
Do you think trying on clothes in different environments (using different surroundings) can help you finding a cloth?	☐	☐	☐	☐	☐
Do you want to use a virtual person (a virtual avatar) rather than seeing yourself in the mirror?	☐	☐	☐	☐	☐
Would you use a virtual fitting room in a shop?	☐	☐	☐	☐	☐
Would you buy a cloth directly after virtually trying it on?	☐	☐	☐	☐	☐

Table 6.1: The questionnaire with a Likert scale, ranging from strongly agree (+ +) over neutral (+/-) to strongly disagree (- -).

The application was tested by seven people (five men and two women) and all of them were answering the questionnaire thereafter.

Nearly all participants (approx. 85 percent) noted that they are at least very interested or interested in fashion, one person showed almost no interest. Considering the question of helpfulness, all participants provided positive (completely agree / agree) results. Furthermore also the appearance of the pieces of the modeled and textured garment models was quite satisfying to the users testing the application. Concerning the individual adaption of clothes to various bodies most of the participants were pleased with the results. Sometimes it happened that a piece of their own cloth showed up and so the virtual garment did not cover up 100 percent of their upper body, for example. The reason for this may also be that the clothes are designed upon a female body. The interaction was well received by the participants, one user gave it a neutral rating. This may lie in the fact that not very computer literate persons have to play a while to discover all the offered functionalities. Five out of seven persons also completely liked the concept of

trying on clothes in front of different environments. It seemed that the participants liked the fact that you can switch the background. This can help people to find an outfit for going out at night, for example. Another interesting aspect is that the majority completely agreed that they want to see themselves in the mirror rather than a virtual person. More than half of the persons agree or completely agree to use the application in a shop. One person was not convinced using the application in a shop because he nonetheless has to try on the clothes, the participant said later. This showed up in the next question, since some of the users that were playing with the application cannot imagine to buy a cloth directly after virtually trying it on in the fitting room.

The exact answers of the questionnaire are plotted in Figure 6.8.

Talking to the participants after they've handed over the questionnaire revealed some interesting facts. A few mentioned that the time-efficient try-on of clothes was seen as a very positive aspect. Overall many of the participants said that they like the general idea of the Virtual Dressing Room. They especially noted that they can take a quick look on several clothes in a very short period of time. If they would try on every piece of garment in the real world this would take a lot more time. They also said that a realistic appearance and behavior of the garment is an important aspect for a virtual fitting room.

As a downside some participants mentioned that when trying on clothes they cannot determine if the real cloth would exactly fit to their body size as the virtual cloth does. So additionally trying on a chosen cloth in a real dressing room to determine the exact fitting might be inevitable. Also some of the participants were not very computer literate so it took them more time to get used to the whole application and to learn the interaction with the fitting room.

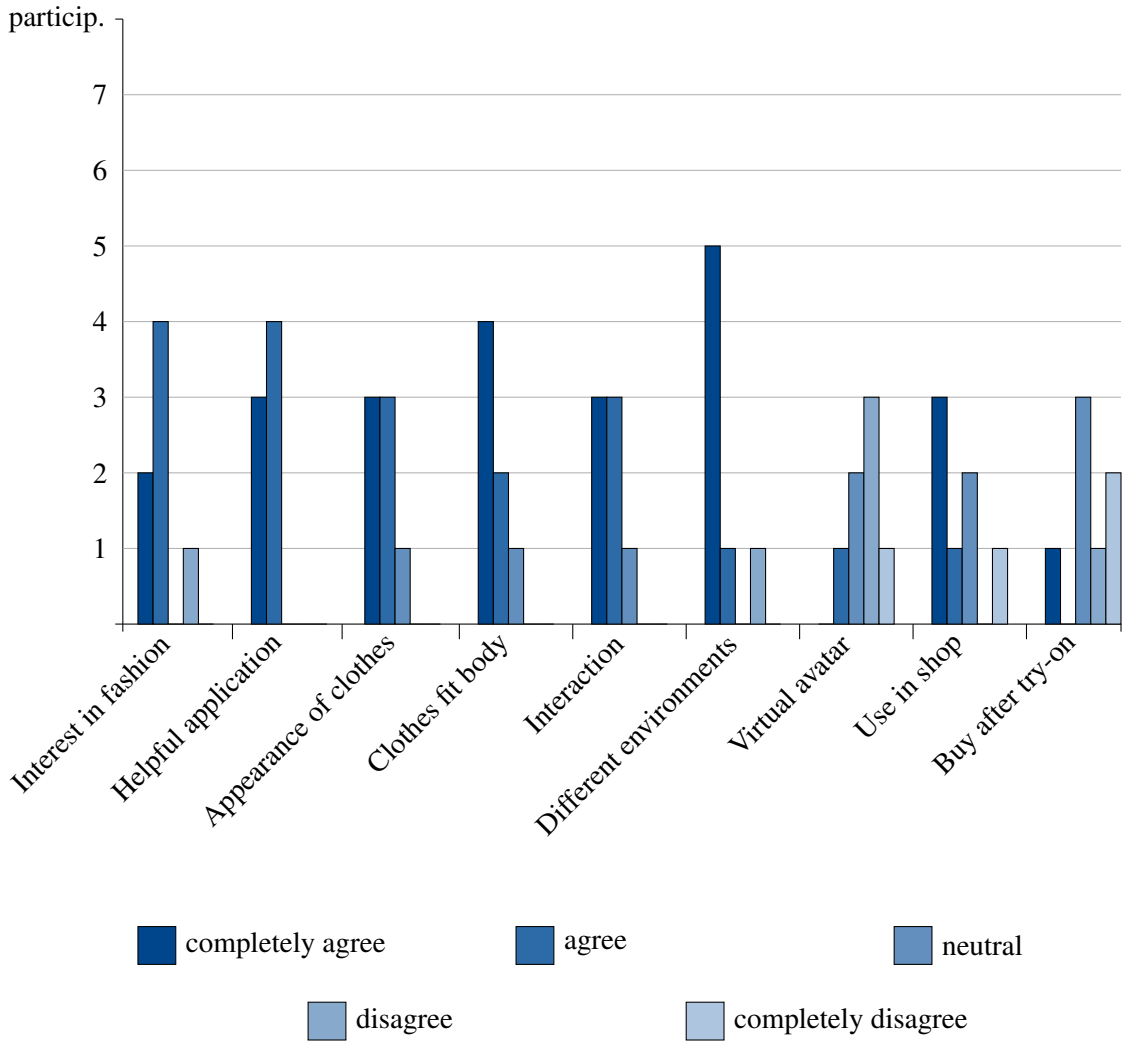

Figure 6.8: The exact results of the questionnaire that was handed over to the participants after using the Virtual Dressing Room. The answers are based on a Likert-scale ranging from completely agree to completely disagree. The ordinate is showing the participants, a short version of the questions is plotted on the abscissae.

CHAPTER 7

Conclusion

After an introduction, the related work was presented, starting with avatar and cloth generation, real time tracking technologies up to an overview of comparable virtual try-ons. Subsequently a closer look on the technologies and frameworks that were used for the implementation of the Virtual Dressing Room was taken. After this the different aspects of the design process up to the construction of the garment models were highlighted. This is followed by the implementation, describing the cloth colliders and the behavior of the garment, for instance. In the last section the tests were executed, also discussing the output, the appearance and the interaction with the Virtual Dressing Room.

Overall, the presented Virtual Dressing Room seems to be a good solution for a quick, easy and accurate try-on of garment. The Microsoft Kinect offers the optimal technology for a successful implementation. Compared to other technologies like augmented reality markers or real-time motion capturing techniques no expensive configurations and time-consuming build-ups are required. From this point of view it is an optimal addition for a cloth store.

Beyond that a simple setup of the system can also be assembled at home since the minimum requirements are a computer with a screen and a Kinect. This can also result in an additional feature for a web shop, for instance. It would allow a virtual try-on of clothes before people are buying it online, taking a closer look at the garment and even conveying the actual behavior of the real cloth. This demonstrates a huge advantage over the common web shopping experience.

7.1 Future work

Regarding future work, the Virtual Dressing Room can be enhanced in some points. Primarily taking a look at the RGB camera of the Kinect, an improvement concerning its resolution can be meaningful. The current resolution of 640x480 is quite low which results in a bad quality of the recorded scene. Since we are not really dependent on the camera of the Kinect, an additional external camera could be added that captures the scene. Of course, this would need a careful calibration, also implicating the position of the camera regarding the Kinect's depth sensor.

Moreover adapting the illumination of the garment to the lighting conditions of the captured video stream of the real world could be an useful addition in regard to realism as well.

Another improvement could be the implementation of a complete scanning procedure of the body that is executed prior to the tracking process. This would result in a more precise adaption of the cloth and can further include the depth/side measurements of a body.

Bibliography

[1] Natsuha Araki and Yoichi Muraoka. Follow-the-trial-fitter: Real-time dressing without undressing. In *Third International Conference on Digital Information Management, 2008. ICDIM 2008.*, pages 33–38, 2008.

[2] avin2/SensorKinect GitHub. https://github.com/avin2/sensorkinect. Accessed: 2012-11-03.

[3] azt.tm's Blog | 2011 | April | 03. http://azttm.wordpress.com/2011/04/03/. Accessed: 2012-06-03.

[4] Caecilia Charbonnier and Clementine Lo. Virtual mirror: A real-time motion capture application for virtual-try-on. Master's thesis, MIRALab - University of Geneva, 2006.

[5] Collision detection in PhysX: Richard Tonge, NVIDIA. http://sglab.kaist.ac.kr/ sungeui/collision_tutorial/richard.pdf.

[6] Frederic Cordier, WonSook Lee, HyeWon Seo, and Nadia Magnenat-Thalmann. Virtual-try-on on the web. MIRALab, University of Geneva, 2001.

[7] Frederic Cordier, Hyewon Seo, and Nadia Magnenat-Thalmann. Made-to-measure technologies for online clothing store. In *IEEE Computer Graphics and Applications, IEEE Publisher*, pages 38–48. MIRALab, University of Geneva, 2003.

[8] A. Divivier, Dr. R. Trieb, and A. Ebert et al. Virtual try-on. Topics in realistic, individualized dressing in virtual reality. In *Virtual and Augmented Reality Status Conference 2004. Proceedings.*, 2004.

[9] P. Eisert, P. Fechteler, and J. Rurainsky. 3-d tracking of shoes for virtual mirror applications. In *Computer Vision and Pattern Recognition, 2008. CVPR 2008. IEEE Conference on*, 2008.

[10] Freedman et al. Depth mapping using projected patterns. United States Patent. US 2010/0118123 A1. PRIME SENSE LTD, May 2010.

[11] FaceCake Marketing Technologies, Inc. http://www.facecake.com/. Accessed: 2012-29-02.

[12] Stefan Hauswiesner, Matthias Straka, and Gerhard Reitmayr. Coherent image-based rendering of real-world objects. In *Proceedings of the 2011 ACM SIGGRAPH Symposium on Interactive 3D Graphics and Games*, 2011.

[13] Stefan Hauswiesner, Matthias Straka, and Gerhard Reitmayr. Free viewpoint virtual try-on with commodity depth cameras. In *Proceedings of the 10th International Conference on Virtual Reality Continuum and Its Applications in Industry*. Graz University of Technology, ACM SIGGRAPH, 2011.

[14] Anna Hilsmann and Peter Eisert. Tracking and retexturing cloth for real-time virtual clothing applications. In *MIRAGE '09 Proceedings of the 4th International Conference on Computer, Vision/Computer Graphics CollaborationTechniques*. Fraunhofer Heinrich Hertz Institute, 2009.

[15] Home | Makehuman. http://www.makehuman.org/. Accessed: 2012-20-05.

[16] Furkan Isikdogan and Gokcehan Kara. A real time virtual dressing room application using kinect. 2012.

[17] Kinect - Wikipedia, the free encyclopedia. http://en.wikipedia.org/wiki/kinect. Accessed: 2012-05-03.

[18] Kinect plugin. http://forum.unity3d.com/threads/67982-kinect-plugin/page10. Accessed: 2012-20-05.

[19] Krista Kjaerside, Karen Johanne Kortbek, Henrik Hedegaard, and Kaj Gronbaek. ARDressCode: Augmented dressing room with tag-based motion tracking and real-time clothes simulation. Central European Multimedia and Virtual Reality Conference, The Eurographics Association, 2005.

[20] Juan Li and Jie Yang. Eyeglasses try-on based on improved poisson equations. In *International Conference on Multimedia Technology (ICMT)*, pages 3058 – 3061, 2011.

[21] Xiao Hu Liu and Yu Wen Wu. A 3d display system for cloth online virtual fitting room. In *Computer Science and Information Engineering, WRI World Congress*, pages 14–18. World Congress on Computer Science and Information Engineering, 2009.

[22] Nadia Magnenat-Thalmann, Bart Kevelham, Pascal Volino, Mustafa Kasap, and Etienne Lyard. 3d web-based virtual try on of physically simulated clothes. In *Computer-Aided Design & Applications, Vol. 8, No. 2*, pages 163–174, 2011.

[23] MAXON: home. http://www.maxon.net/de/home.html. Accessed: 2012-14-03.

[24] Microsoft Kinect for Windows | Develop for the Kinect | Kinect for Windows. http://kinectforwindows.org/. Accessed: 2012-05-03.

[25] G. Müller, J. Meseth, M. Sattler, R. Sarlette, and R. Klein. Acquisition, synthesis and rendering of bidirectional texture functions. In *Eurographics 2004, State of the Art Reports*, pages 69–94, 2004.

[26] OpenNI. http://openni.org/. Accessed: 2012-23-02.

[27] PhysX by NVIDIA - PhysX SDK Introduction. http://http.download.nvidia.com/developer/cuda/seminar/tdci_physx.pdf.

[28] PrimeSense Natural Interaction. http://www.primesense.com/. Accessed: 2012-05-03.

[29] Quality of Kinect RGB Camera? - Google Groups. https://groups.google.com/forum/?fromgroups#!topic/openni-dev/0rpxarkjl3k. Accessed: 2012-22-05.

[30] Jamie Shotton, Andrew Fitzgibbon, Mat Cook, Toby Sharp, Mark Finocchio, Richard Moore, Alex Kipman, and Andrew Blake. Real-time human pose recognition in parts from a single depth image. In *Computer Vision and Pattern Recognition (CVPR), 2011 IEEE Conference on*. Microsoft Research Cambridge & Xbox Incubation, 2011.

[31] The Narkissos Project. http://www.icg.tugraz.at/project/narkissos. Accessed: 2012-26-02.

[32] Time-of-Flight and Kinect Imaging: Victor Castaneda, Nassir Navab. http://campar.in.tum.de/twiki/pub/chair/teachingss11kinect/2011-dsensors_labcourse_kinect.pdf. Accessed: 2012-05-08.

[33] Unity - 3D Game Engine. http://unity3d.com/. Accessed: 2012-14-03.

[34] Virtual Fitting Room for Topshop | AR Door. http://ar-door.com/2011/05/virtualnaya-primerochnaya-dlya-topshop/?lang=en. Accessed: 2012-29-02.