

## I. OSOBNÍ A STUDIJNÍ ÚDAJE

Příjmení: **Ryabova** Jméno: **Margarita** Osobní číslo: **474429**  
Fakulta/ústav: **Fakulta elektrotechnická**  
Zadávající katedra/ústav: **Katedra počítačové grafiky a interakce**  
Studijní program: **Otevřená informatika**  
Studijní obor: **Počítačové hry a grafika**

## II. ÚDAJE K BAKALÁŘSKÉ PRÁCI

Název bakalářské práce:

**Virtuální zkoušební kabinka / Magické Zrcadlo**

Název bakalářské práce anglicky:

**Virtual try on - Magic Mirror**

Pokyny pro vypracování:

Prostudujte aktuálně používané postupy pro realizaci tzv. magického zrcadla (jeden z druhů rozšířené reality), tj. člověk se dívá do monitoru s kamerou, což připomíná zrcadlo. Obraz člověka je doplněn virtuálními objekty (např. 3D modely oblečení). Prostudujte možnosti snímání lidské postavy a realtime výpočtu kostry člověka a aplikujte toto pro vytvoření virtuální kabinky pro vizualizaci oblečení.  
Navrhnete a implementujte aplikaci realizující magické zrcadlo pro použití v muzeu, kde si návštěvníci budou moci virtuálně vyzkoušet historické kusy oblečení, které jsou rekonstruovány pomocí fotogrametrie (3D rekonstrukce z fotografií).  
Předpokládejte, že oblečení se skládá z více kusů, např. klobouk, sako, košile, rukavice, kalhoty, boty (3D modely překrývajících se kusů, mohou být předem spojeny - např. košile a sako). Navrhnete systém pro přepínání oblečení uživatelem (5 různých skupin oblečení, 1 x dětské, 2x mužské, 2x ženské). Definujte postup, jak rozšířit aplikaci o další kusy oblečení.  
Systém otetuje s uživateli s rozlišnými fyziologickými rysy (přibližně 10 uživatelů) a definujte limity systému (jak z výkonostního pohledu, tak z vhodnosti páru uživatel-model oblečení).

Seznam doporučené literatury:

- 1] Joseph J. LaViola, Jr. et all. 3D User Interfaces: Theory and Practice, second edition. 2017. Addison Wesley Longman Publishing Co.,
- 2] Dieter Schmalstieg, and Tobias Hollerer, Augmented Reality: Principles and Practice (Usability), Addison Wesley 2016
- 3] T. Blum, V. Kleeberger, C. Bichlmeier and N. Navab, "mirracle: An augmented reality magic mirror system for anatomy education," 2012 IEEE Virtual Reality Workshops (VRW), Costa Mesa, CA, 2012, pp. 115-116.

Jméno a pracoviště vedoucí(ho) bakalářské práce:

**Ing. David Sedláček, Ph.D., katedra počítačové grafiky a interakce FEL**

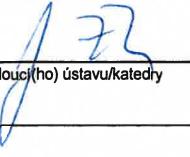
Jméno a pracoviště druhé(ho) vedoucí(ho) nebo konzultanta(ky) bakalářské práce:

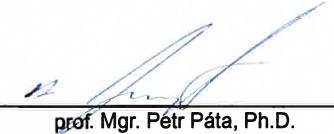
Datum zadání bakalářské práce: **04.03.2021**

Termín odevzdání bakalářské práce: \_\_\_\_\_

Platnost zadání bakalářské práce: **19.02.2023**

  
**Ing. David Sedláček, Ph.D.**  
podpis vedoucí(ho) práce

  
podpis vedoucí(ho) ústavu/katedry

  
**prof. Mgr. Petr Páta, Ph.D.**  
podpis děkana(ky)

### **III. PŘEVZETÍ ZADÁNÍ**

Studentka bere na vědomí, že je povinna vypracovat bakalářskou práci samostatně, bez cizí pomoci, s výjimkou poskytnutých konzultací.  
Seznam použité literatury, jiných pramenů a jmen konzultantů je třeba uvést v bakalářské práci.

---

Datum převzetí zadání

---

Podpis studentky

CZECH TECHNICAL UNIVERSITY IN PRAGUE

FACULTY OF ELECTRICAL ENGINEERING

DEPARTMENT OF COMPUTER GRAPHICS AND INTERACTION

BACHELOR THESIS

---

## Virtual Try On - Magic Mirror

---

*Author:*

Margarita Ryabova

*Supervisor:*

Ing. David Sedláček, Ph.D.

2021





---

## Acknowledgements

---

Firstly, I would like to thank my supervisor Ing. David Sedláček, Ph.D. for all the suggestions and help given during my work on this thesis. Secondly, I would like to thank the Department of Computer Graphics and Interaction of Czech Technical University in Prague for providing me with Kinect v2 device for my work on this thesis. Then, I would very much like to thank my family members for inspiring me during my study at Czech Technical University in Prague, this thesis would not have been finished without their invaluable support. Lastly, I would like to thank my close friends, who have agreed to participate in the final application testing, thank you for your priceless feedback and criticism.



---

## **Declaration**

---

I hereby declare that the presented thesis is my own work and that I have cited all sources of information in accordance with the Guideline for adhering to ethical principles when elaborating an academic final thesis. I acknowledge that my thesis is subject to the rights and obligations stipulated by the Act No. 121/2000 Coll., the Copyright Act, as amended, in particular that the Czech Technical University in Prague has the right to conclude a license agreement on the utilization of this thesis as school work under the provisions of Article 60(1) of the Act.

In Prague on 20th of May, 2021 \_\_\_\_\_

Czech Technical University in Prague  
Faculty of Electrical Engineering

© 2021 Margarita Ryabova. All rights reserved.

*This thesis is school work as defined by Copyright Act of the Czech Republic. It has been submitted at Czech Technical University in Prague, Faculty of Electrical Engineering. The thesis is protected by the Copyright Act and its usage without author's permission is prohibited (with exceptions defined by the Copyright Act).*

**Citation of this thesis**

Ryabova, Margarita. *Virtual Try On - Magic Mirror*. Bachelor's thesis.  
Czech Technical University in Prague, Faculty of Electrical Engineering, 2021.

---

## Abstrakt

---

Hlavním cílem této práce je analyzovat možné způsoby realizace virtuální zkušební kabinky a implementovat takovou aplikaci. Na základě výzkumu bylo rozhodnuto provést implementaci pomocí Rozšířené reality (AR) ve spojení s zařízením Kinect v2 a Unity Engine. Výsledný program byl úspěšně otestován na osobním počítači s operačním systémem Windows.

**Keywords:** Rozšířená realita, AR, Kinect, Virtuální zkušební kabinka, Avatar, Unity, C#, Off-Axis projekce, Inverzní kinematika, IK

---

# Abstract

---

The main goal of this thesis is to analyze the possible solutions of a virtual fitting room application and implement such an application. Based on the research, it was decided to carry out the implementation in Augmented Reality with the conjunction of Kinect v2 and Unity Engine. The final program was successfully tested on a personal computer with Windows OS.

**Keywords:** Augmented Reality, AR, Kinect, Virtual fitting room, Virtual changing room, Virtual dressing room, Avatar, Unity, C#, Off-Axis Projection, Inverse Kinematics, IK

---

# Contents

---

|          |   |           |
|----------|---|-----------|
| <b>1</b> | <b>Introduction and Motivation</b>                      | <b>1</b>  |
| <b>2</b> | <b>Problem Analysis</b>                                 | <b>4</b>  |
| 2.1      | Application in 2D . . . . .                             | 4         |
| 2.2      | Application in 3D . . . . .                             | 6         |
| 2.3      | Application in AR . . . . .                             | 8         |
| 2.4      | Conclusion . . . . .                                    | 10        |
| <b>3</b> | <b>Solution plan</b>                                    | <b>11</b> |
| 3.1      | Overview of Kinect key features . . . . .               | 11        |
| 3.2      | Game Engine as an application basis . . . . .           | 12        |
| 3.3      | Mirror simulation . . . . .                             | 12        |
| 3.4      | Data visualization . . . . .                            | 13        |
| 3.5      | Outfit visualisation and mapping . . . . .              | 14        |
| 3.6      | User interaction . . . . .                              | 16        |
| 3.7      | Conclusion . . . . .                                    | 17        |
| <b>4</b> | <b>Implementation</b>                                   | <b>18</b> |
| 4.1      | Off-Axis Projection . . . . .                           | 18        |
| 4.2      | Kinect Data Visualization . . . . .                     | 21        |
| 4.2.1    | 2D visualization . . . . .                              | 21        |
| 4.2.2    | Point cloud visualization . . . . .                     | 22        |
| 4.3      | Inverse Kinematics . . . . .                            | 25        |
| 4.3.1    | FABRIK Algorithm . . . . .                              | 25        |
| 4.4      | Outfit Rendering . . . . .                              | 27        |
| 4.4.1    | Rendering order . . . . .                               | 27        |
| 4.4.2    | Usage of stencil buffer for outfits rendering . . . . . | 28        |
| 4.4.3    | Avatar Attunement . . . . .                             | 31        |
| 4.5      | Clothing Combination System . . . . .                   | 31        |
| 4.5.1    | Clothes Layering . . . . .                              | 32        |
| 4.5.2    | Clothing layering using stencil buffer . . . . .        | 33        |
| 4.5.3    | Outfit Models . . . . .                                 | 35        |
| 4.6      | User Interface . . . . .                                | 36        |
| 4.6.1    | Interactions With Gestures . . . . .                    | 38        |
| 4.7      | Playground room . . . . .                               | 39        |

|  |           |
|--|-----------|
| <b>5 Application Testing</b>                   | <b>41</b> |
| 5.1 User testing . . . . .                     | 41        |
| 5.1.1 User #1 . . . . .                        | 43        |
| 5.1.2 User #2 . . . . .                        | 43        |
| 5.1.3 User #3 . . . . .                        | 43        |
| 5.1.4 User #4 . . . . .                        | 44        |
| 5.1.5 User #5 . . . . .                        | 44        |
| 5.1.6 User #6 . . . . .                        | 45        |
| 5.1.7 User #7 . . . . .                        | 45        |
| 5.1.8 User #8 . . . . .                        | 45        |
| 5.1.9 Multiple users test . . . . .            | 46        |
| 5.1.10 User Testing Conclusion . . . . .       | 47        |
| 5.2 Performance Testing . . . . .              | 48        |
| 5.2.1 Performance Testing Conclusion . . . . . | 48        |
| <b>6 Conclusion</b>                            | <b>49</b> |
| <b>Bibliography</b>                            | <b>50</b> |
| <b>A Contents of the enclosed flash disk</b>   | <b>54</b> |

## Chapter 1

---

# Introduction and Motivation

---

The contemporary pace of life dictates its own rules for every field, including shopping. Modern people rarely have enough time to spend on this activity, thus, the computer industry comes up with new ideas and applications to make the shopping process more effective and accessible.

People are increasingly fond of online shopping as it becomes possible to make quick and efficient purchases from anywhere. Distance shopping is also a better option for those who prefer to avoid crowds, which proved to be a reasonable decision during the Covid-19 pandemic. Considering the changes taking place in the world, it would be better to transfer as many activities as possible in digital format. Online shopping has another important advantage, as it gives customers more freedom of choice. Unfortunately, even the biggest offline stores are unable to store an unlimited selection of clothes in different sizes. The assortment of the store also strongly depends on the store's location itself; sometimes a wanted item may not be available for sale in a particular city or country, which makes offline purchase impossible in this case.

Despite the fact that online shopping seems to be a much more convenient way to buy clothes, there is a negative trait, which turns out to be decisive for many customers. Some people avoid purchasing outfits online without trying them on, as it is impossible to fully evaluate the fit or the fabric quality. Besides, it can be uneasy to pick the right size, because not every online store provides size charts, making it difficult to make the right decision. One of the possible solutions for the mentioned problem could be a so-called *virtual fitting room application* [figure 1.1] (also known as *virtual dressing room* or *virtual changing room*), that allows users to try on outfits virtually and receive a better picture of clothes.

First virtual fitting rooms appeared in the early 2000s in a number of stores, and their main purpose was to help consumers with shopping [1]. The system allowed users to try on outfits instantly, making it faster for them to decide on specific items than trying the clothes on physically. As a result, the users could choose different clothing styles, sizes and colors, they could also save created outfits and compare them, to pick the most appealing one. Moreover, it

was a much easier way for customers to search through the catalog and look for specific items. The *Technology visibility and consumer adoption of virtual fitting rooms (VFRs): a cross-cultural comparison of Chinese and Korean consumers* by H. Lee [2] analyzes the popularity of virtual fitting rooms from perspective of marketing in Asian region and compares it to Western market and concludes that "the size of the VFR market is expected to grow consistently".



Figure 1.1: Example of virtual clothes fitting in store with Microsoft Kinect [3].

In the last years with the development of technology it became possible to improve the virtual fitting room concept, giving the users an option to try new clothes in the comfort of home without visiting stores. This approach can be successfully combined with online sales, helping the customers to pick the most suitable items and reducing the possibility of purchase returns. The *Tendency to Use the Virtual Fitting Room in Generation Y* by M. Moroz [1] widely analyzes the advantages of virtual fitting rooms in online shopping sector in order to reduce the number of clothes returns.

Such systems would address some of users' concerns about online shopping, but not all of them. Unfortunately, the described method cannot provide users with the full idea of a clothing piece, as there is no technology to properly translate fabrics texture or quality of tailoring yet. Given this fact, virtual fitting rooms may solve this problem only partially.

Although the mentioned concept seems to be effective in marketing, the idea can be also applied in other fields. Virtual dressing rooms have a significant advantage offering the user an ability to try on any type of clothing, even the outfits, that in modern realities cannot be purchased or even manufactured. It is possible to create a historical outfit from ancient eras in digital form and offer users access to try it on. That way they will receive an experience, that cannot be repeated in real life. It is also possible to model non-existing fictional outfits that simply cannot be created in practice.

Thus, there is another field where the use of virtual dressing rooms is becoming more and more popular: the museums create unique AR installations, allowing users to "try on" costumes from the past (for example, [4] and [5]). Such systems in museums are still very rare at the moment, but provided that the developer gets access to 3D reconstructions of historical costumes, it will be possible for him to turn his virtual fitting room application into a system for trying on clothes from past eras.

Additionally, virtual dressing rooms can be used in clothes designing, as it would help designers to visualize their drafts and modify them easily. The program would present the general impression of the product before the manufacturing.

Lastly, such applications may be applicable for educational purposes, for example the *mirracle: An augmented reality magic mirror system for anatomy education* [6] presents the usage of a similar application for human anatomy studies.

Virtual fitting rooms allow users to solve a number of mentioned problems, but unfortunately, this solution is not optimal. Even though the indicated method is capable of displaying outfits on top of the user's representation, it does not present an idea of fabric quality and comfort of clothing yet.

## Chapter 2

---

# Problem Analysis

---

There are several ways to implement the concept of a virtual changing room, to do so it is possible to apply simpler ideas and technologies or more advanced ones. This section considers and compares the realizations of these systems in 2D, 3D and augmented reality based on the research on existing applications I have done.

### 2.1 Application in 2D

First of all, I would like to analyze the most primitive implementations of virtual clothes fitting - the solutions in 2D space. Let us consider the applications that allow users to put clothes on a mannequin, both the dummy and the outfits being presented in the form of two-dimensional images. In this approach, the mannequin is representing the user, playing the role of the avatar [figure 2.1]. Such applications are user-friendly and easy to implement, they also do not have heavy hardware demands and would work fluently on any modern device. These dress up programs are designed mainly to help customers to combine fits, prints and colors, however because the clothes are represented by flat images, it is not possible to fully convey all of the outfits' qualities. The other issue in this concept would be "the border" between the avatar and the user, that prevents the second from imagining how the clothes will look on them. From the user's perspective, there is no connection between him and the mannequin in the mentioned solution. Nonetheless, this approach is widely used by online stores and its popularity was described in *Tendency to Use the Virtual Fitting Room in Generation Y* by M. Moroz [1, p. 242].

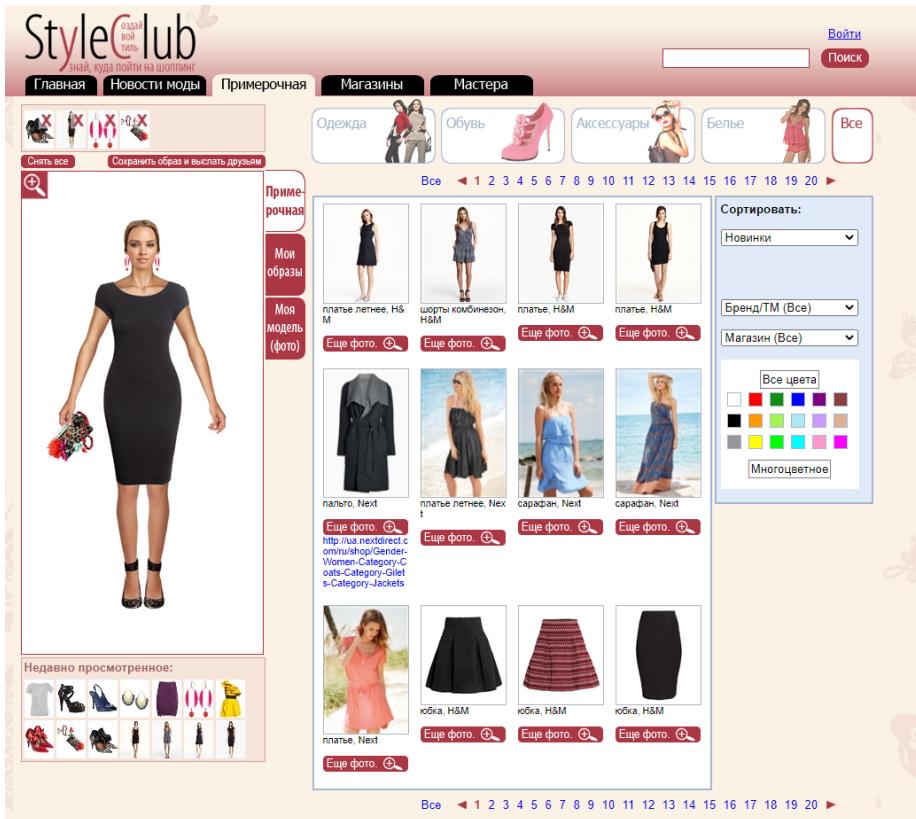


Figure 2.1: Example of virtual clothes fitting in 2D with a standard two-dimensional avatar. Style Club web application [7].

The previously described approach can be improved by allowing the user to personalize the mannequin, for example, by using a user's photo instead of the picture of a prepared character [figure 2.2]. This upgrade will solve the problem with the “facelessness” of the mannequin, making it easier for the user to associate himself with the avatar. This method will not affect the application's performance and it will improve the “trying on” experience the user gets. This way, the customers will be able to evaluate not only the colors and designs of clothes, but also the shapes and the fits. Nevertheless, such minor changes in product design will have an impact on the core mechanics of the application. Because users' bodies are not standardized, in this solution the developers are unable to create universal clothes that fit everyone. They would have to deform the images, that represent outfits, to match the user's constitution, and to do it right, they might need to limit the user on uploading photos with one specific pose. This, in turn, complicates the process of the application usage for the user himself.



Figure 2.2: Example of virtual clothes fitting in 2D with a customized avatar, made from the user’s photo. Zeekit android application [8].

## 2.2 Application in 3D

Next, let us consider the solution in 3D space. It is possible to do this in a similar way, as in the first case, but instead of using a two-dimensional mannequin, let us prepare a 3D model of the avatar and map selected clothes onto it [figure 2.3]. Such an approach is easy to implement, and if we compare the performance of the 2D and 3D methods, the difference would not be dramatic. Even though the 3D solution requires better hardware than the previous application, it will still easily work on the modern platforms. The three-dimensional approach would let customers examine clothes meticulously, as they would be able to rotate the avatar and inspect the outfits from different points of view. Unfortunately, the users may be still unattracted to the idea of dressing up a mannequin that has nothing in common with them.

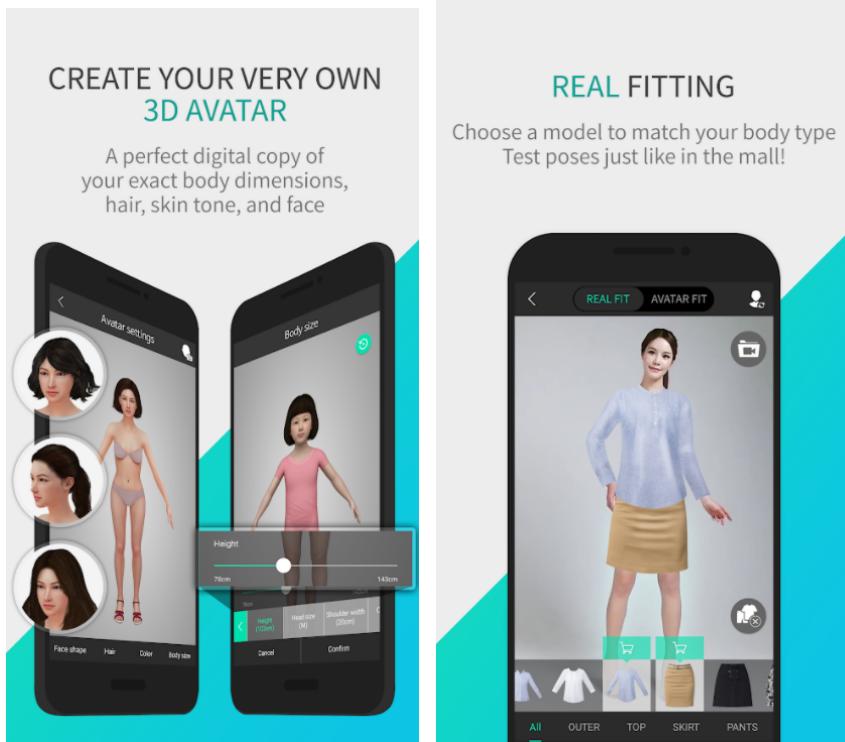


Figure 2.3: Example of virtual clothes fitting in 3D with a standard three-dimensional avatar. FIT'N SHOP android application [9].

This approach can also be improved by allowing the user to dress up not a foreign avatar, but a three-dimensional copy of himself. To do so, the application would need to scan the user and create a 3D model from the received data. The resulting model will replace the standard mannequin in the application, helping the user to associate himself with the customized avatar easily [figure 2.4]. In this approach, the concept of a virtual dressing room starts to open up. The solution would guide users to pick clothes' fits, designs and probably even recommend sizes. However, the process of user scanning in this solution is not a simple operation and if done wrong, the resulting model turns out unusable. It should also be mentioned that the quality of a scanned model is typically much worse than the quality of a manually prepared avatar, especially if the resolution of used images is low. The second down side of this solution is the need to dynamically deform 3d meshes of clothes to match the user's model (that was not necessary with a standard mannequin). Doing so would be much more complicated than editing 2D images in the one of the previously mentioned solutions.

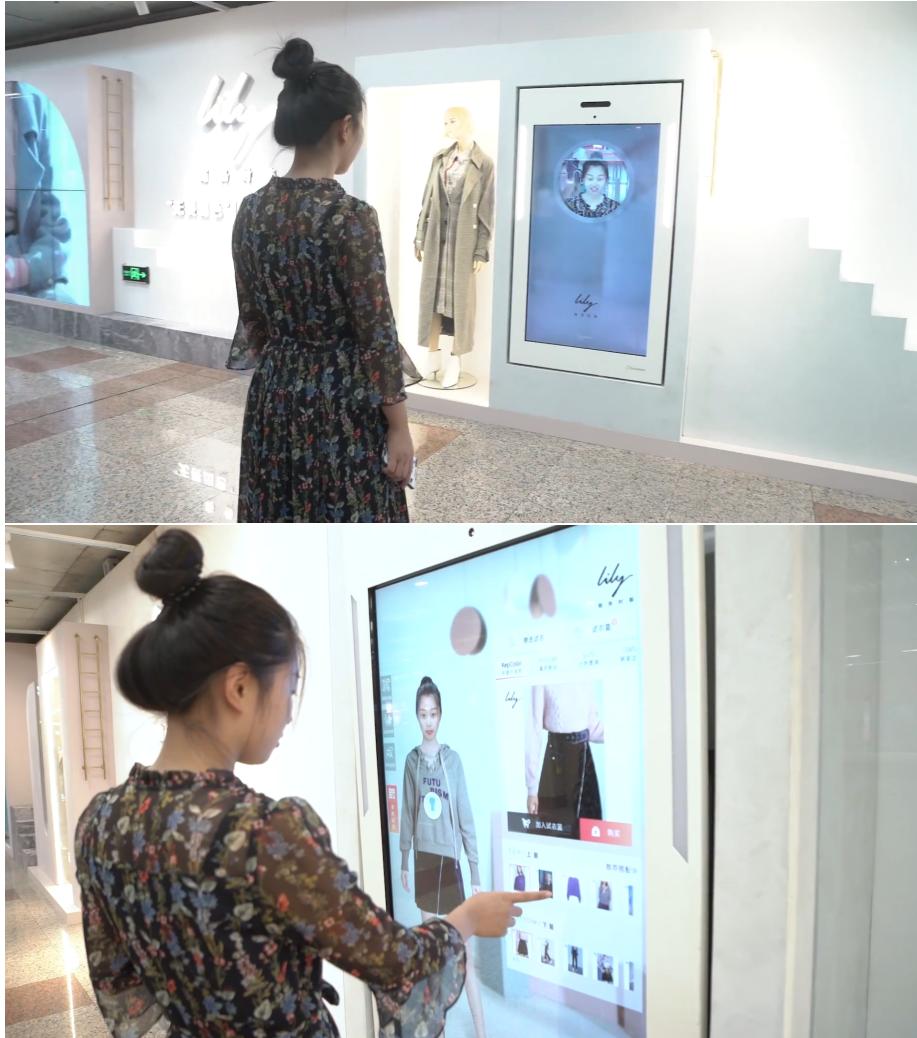


Figure 2.4: Example of virtual clothes fitting in 3D with a customized avatar made with user scanning. Virtual fitting room of brand Lily at in Shanghai metro. Source video: [10].

### 2.3 Application in AR

The last option we will consider allows to make the virtual fitting process even more natural. To make this happen, we will not force the user to dress up an artificial avatar or convince the user that the avatar is similar to him. Instead, let us project the clothing directly onto the user's image [figure 2.5]. It can be done by implementing an interactive mirror-like application with a clothes try-on feature. The user sees himself in the display as if the content on the screen was the mirrored reality, but augmented by an interface, that allows the user to pick specific clothes and map them on his “reflection” in the

monitor. To do so, the application will require a device with a camera, that will send captured data to the program for a further processing.



Figure 2.5: Example of virtual clothes fitting in AR. Fitnect Kinect application [11].

Such a concept can be implemented with use of a classic web camera. This device is easily accessible, most people own a web camera at home, so they will not need to purchase special equipment to make the application function. Unfortunately, most of the web cameras can not have the ability to capture high quality videos or to trace the movement well. However, this is not the main problem of this solution, because in case of using the web cameras for movement tracing it would be necessary to implement the whole motion capture system. Such algorithms are complicated and will require much time and knowledge to implement.

The other possibility is to implement the application on smartphones, that are also accessible in our daily life, but tend to have a rather better quality of captures, compared to web cameras. Unfortunately, the body tracking problem is still present in this case and the developers would have to deal with it on their own.

The last option we will consider is the use of one of Microsoft Kinect devices. The Kinect devices have a significant advantage over the previously discussed options, because they were designed to deal with body tracking tasks primarily. The devices are supported by an SDK, specially made for development of the programs and games, that rely on body tracking. The devices offer a complete implementation of motion capture and depth and color records. The downside of this solution is that Kinect is not a popular device and not every person has one at home. It is also not a cheap system and people might not be interested in buying it only to try out one application. Nonetheless, this specific technology is popular between developers, who implement virtual dressing rooms and there is a high number of materials on this topic, such as:

- The *Experiences in the Development of an Augmented Reality Dressing Room* by U. Erra [12] characterizes an implementation of a virtual dressing room application with Kinect technology and Unity Engine, explaining the possible instruments that can be used for this approach.
- The *An Efficient Magic Mirror Using Kinect* by J. Uddin [13] proposes a concept of real time virtual dressing room with gesture based interaction techniques.
- The *Virtual Dressing Room Application* by A. Masri [14] shares an implementation of such application using Windows Presentation Foundation and contains useful approach on clothes adjustment.
- The *Augmented Reality Platforms for Virtual Fitting Rooms* by I. Pa-choulakis and K. Kapetanakis [15] compares Kinect for Windows to Asus Xton Pro Live and analyzes already existing AR virtual fitting room applications.
- The *Augmented Reality Principles and Practice* by D. Schmalstieg and T. Höllerer [16] as well mentions a virtual dressing room to be an interesting example of Augmented Reality applications.

## 2.4 Conclusion

Based on the research, I came to the conclusion that the ideal option would be to implement the virtual dressing room application using AR technology. Of all the options given, this implementation seems the closest one to the user's everyday life, using such a system would make clothes fitting more natural. Choosing between web cameras, smartphones and Kinect, I opted for the latter device, since it has the best functionality and motion capture accuracy to complete the task and is ready for development.

## Chapter 3

---

# Solution plan

---

This chapter contains a detailed solution to the problem of virtual clothes fitting in augmented reality using Microsoft Kinect. For this thesis, I used a Kinect v2 in conjunction with the official Kinect SDK[17]. First of, I have started with a brief analysis of the device key features, secondly, I decided on a game engine to be used for the task solution. Next, I have analyzed the mirror projection that would provide a simulation of the reflection effect. Afterwards I have decided on ways to visualize the user and outfits. Lastly, I have reviewed the possible ways to implement user interactions.

### 3.1 Overview of Kinect key features

Microsoft Kinect v2 [figure 3.1] is a sensing device “composed by a RGB camera with resolution of  $1920 \times 1080$  pixels, an infrared camera with resolution of  $512 \times 424$  pixels and an infrared emitter”[18]. The device provides access to images from infrared and color cameras, a depth map, as well as to the position of the user’s body (coordinates of his joints).

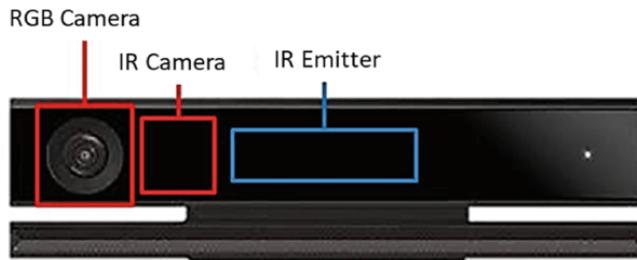


Figure 3.1: The Kinect v2 sensor front with cameras and emitter positions[18].

The mentioned data can be used in development of projects that require body tracking. It can be easily done with help of an official Microsoft Kinect SDK

that contains all necessary libraries and lets developers work with Kinect's data in different programming languages and game engines. The device has proven to be used in many different fields (for example, in robotics and medicine [18][6]), even though it was developed as a gaming controller.

### 3.2 Game Engine as an application basis

It is clearly not an optimal solution to develop a new engine for the given task, as there already are great engines compatible with Kinect. For further implementation I had to decide what third-party engine to use to make the work as effective as possible.

Unity3d Game Engine [19] is a well documented gaming engine, that is popular among game developers. Because of this fact, there is a significant number of studying materials such as books, articles, and tutorials (official ones and the others, made by enthusiasts). The existence of official Microsoft support of Kinect on Unity made this option the most suitable for our implementation.

Unreal Engine [20] is a well-known alternative to Unity, but in case of this thesis it was not a proper option, as it is not supported by Microsoft and there is no official SDK, that provides developers to work with Kinect on this engine. On the other hand, there is a Kinect 4 Unreal project [21] that provides the necessary instruments to start developing with these two components. The mentioned API puts limits on developers, allowing them to use it for free only for low-budget projects; otherwise, the developers have to purchase a licence. The number of available tutorials and materials for this topic is much smaller, than in case of Unity, that is why the previous engine seems to be a better option.

It was also possible to use other less popular engines for the implementation, but in this case it would require certain effort in connecting Kinect to the workspace. For efficiency of the solution, I have decided to avoid such low-level settings.

Based on the research, Unity engine was the ideal option for the given task, as it provides stable work with Kinect sensors and a large amount of studying materials on similar topics.

### 3.3 Mirror simulation

In the project I was planning to create a mirror-like version of a virtual fitting room. To do so, I had to consider projecting the content of the application onto the monitor in the specific way.

Normally, if we look into a mirror from different angles, we are able to see reflections of the opposite sides of our surroundings. This is the type of projection that is not used in most of the applications and games on a daily basis, because there is no need to implement it that way. Most of the time the developers apply a classic perspective projection, that in calculations does not consider the user's position according to the monitor or the angle at which the user is looking

at the screen. Generally, the standard perspective projection assumes that the user is located right in front of the display, looking at its center [22]. However, the thesis task required a different approach, in which new projection deforms the content of the screen based on the user's position. "This behaviour may be achieved by using the off-axis perspective technique to simulate distortion when the viewpoint is shifted relative to the mirror plane" [22].

The off-axis projection principles are discussed and explained in the next chapter as a part of the implementation process [4.1].

### 3.4 Data visualization

Let us assume that we receive the data from the Kinect and visualize it in the 3D scene with a camera set up. After applying the projection matrix on the camera's content, the result of such transformation is displayed on the screen. Such output is a reflection of the environment that the Kinect is capturing. Let us call the final result of this pipeline a *virtual reflection*.

Because I have already found the projection solution in the previous section, to create a virtual reflection I only needed to discover a way to visualize the data Kinect collects. There are several approaches to solve this problem, each of them has its own advantages and disadvantages. This section reviews the possible solutions and compares them with each other.

The first way to solve the task of data visualization is to combine a 2D image from the Kinect RGB camera and map a 3D outfit on top of it. Such an approach is easy to implement, as the camera output is used as a base of the visualization. However this solution conflicts with the primary task of mirror simulation. Because the image is taken directly from the camera output, the perspective projection on captured frames will stay fixed. For the mentioned reasons, this method would not work with mirror simulation, but it could be used to represent captured data without mirror-like deformation [figure 2.5].

The second approach that could be potentially applied in this work is a concept of a point cloud [figure 3.2]. "A point cloud is a set of data points in space. The points represent a 3D shape or object. Each point has its set of X, Y and Z coordinates" [23]. It is possible to create the point cloud from Kinect output data and apply a projection transformation on it.

Even though this solution is the most suitable for the given task, it also has significant disadvantages. Firstly, the final rendered image looks grainy, as infrared sensor resolution is limited, which may look unpleasant from the user's perspective. Secondly, the processed data is not filling the display most of the time, except for the situation, when the position of the virtual camera and the position of real sensor match.



Figure 3.2: Example of a point cloud, image source: [24].

### 3.5 Outfit visualisation and mapping

In most of the virtual fitting rooms there is a user's avatar in the scene, the picked clothes map onto this avatar. In the case of an AR application, I had to map the outfits directly onto the user's image. It is possible to pretend that this is happening, while using an avatar in the background, but not rendering it. This is why I have decided to place the clothes on the avatar, that would be animated by user's movements but never rendered. With this approach only the clothes will remain on the screen. Furthermore, an invisible avatar could be used as a stencil to hide those parts of clothing that should not be visible.

Since Kinect provides a set of the user's joints, it is possible to position the avatar in virtual space and mimic the user's pose. However, there issues that would have to be fixed [figure 4.14]:

- the proportions of the user's body in general do not match the proportions of the avatar
- due to limitations of the Kinect sensor the lengths of the user's limbs in provided data may vary from frame to frame

The solution to these problems would be a use of inverse kinematics for consistent animation of the avatar [25].

Unity has a built-in IK implementation but it is not entirely clear which algorithm was used to achieve it; from the documentation it is seen, that the algorithm is implemented in native code and I could not find any specific information on the algorithm itself. Moreover, due to its focus primarily on games, it has a number of disadvantages in case of the given task:

- unity IK can only be applied to arms (brachial bone and forearm) and legs (thigh and shin)
- access to avatar bones is only possible through the Mecanim system
- animator component is needed and all manipulations should be provided inside OnAnimatorIK method

Although the limitations of the built-in IK system could be overcome, in the thesis I have decided to use my own implementation of inverse kinematics, as it allows fine tuning and does not operate on other systems, such as Animator and Mecanim. As a result, in case of this thesis it was ideal to implement IK from scratch, the custom implementation must have the following advantages:

- IK chains include hands and feet
- IK not only for the limbs but also for the spine

In interactive applications (such as games), inverse kinematics is calculated with iterative heuristic methods, due to the ease of implementation and the low cost of computation [26]. “The most popular heuristic algorithms are: Cyclic Coordinate Descent (CCD), and Forward And Backward Reaching Inverse Kinematics (FABRIK)” [27]. It was possible to use one of the two algorithms in the implementation, let us briefly characterize both of them.

Cyclic Coordinate Descent (CCD) operates with rotation angles to reach target position. However it is necessary to implement constraints to achieve more realistic animation. The algorithm has further advantages and disadvantages [3.1]:

| Pros  | Cons   |
|---|--|
| <ul style="list-style-type: none"> <li>• simple to implement</li> <li>• extremely fast algorithm, provides a linear-time complexity in the number of degrees of freedom [26]</li> <li>• returns a numerically stable solution [26]</li> </ul> | <ul style="list-style-type: none"> <li>• difficult to implement for multiple end effectors [26]</li> <li>• often produces unrealistic animation [26]</li> <li>• suffers from erratic discontinuities [26]</li> <li>• the closer a joint is to the end of the chain, the more it bends [26]</li> <li>• found solution is not optimal, not suitable for physically correct simulation</li> </ul> |

Table 3.1: Pros and cons of Cyclic Coordinate Descent heuristic algorithm

Forward And Backward Reaching Inverse Kinematics (FABRIK) relies on previously calculated state of a chain of joints and manipulates with joints positions. FABRIK analysis [3.2]:

| Pros  | Cons   |
|---|--|
| <ul style="list-style-type: none"> <li>• simple to implement</li> <li>• “computational cost for each joint per iteration is low” [26]</li> <li>• if the target is in range, the algorithm always returns a solution [26]</li> <li>• returns smooth motion without erratic discontinuities [26]</li> <li>• the closer a joint is to the root of the chaint, the more it bends [26]</li> <li>• can be easily modified to solve a task with multiple end effectors [26]</li> </ul> | <ul style="list-style-type: none"> <li>• constraints implementation is more complicated than in CCD [26]</li> <li>• found solution is not optimal, not suitable for physically correct simulation</li> </ul> |

Table 3.2: Pros and cons of Forward And Backward Reaching Inverse Kinematics heuristic algorithm

## 3.6 User interaction

Interactive programs need to be controlled by the user in some way. In case of this project, the user should be able to select the outfit and, if necessary, adjust its mapping.

Most applications use classic input interfaces: keyboard, mouse or controller. Kinect provides the ability to use the entire body of the user as a ”controller”, this idea eliminates the need to set up the stand with classic input devices in front of the virtual fitting room. Instead, the user will navigate through the interface using gestures.

I have decided to implement both techniques: the primary way of interaction would be using a computer mouse, and the additional way would be using simple gestures for navigation and picking clothes. The gestures would be based on simple ”raise hand” or ”grab” actions to make the navigation through the UI natural and intuitive, as suggested in [28][p.214].

### **3.7 Conclusion**

The given solution plan contains main terms and ideas used in the implementation process. Some approaches had to be modified to correspond to the given task.

## Chapter 4

---

# Implementation

---

This chapter presents in detail the process of implementing each of the approaches mentioned in the last chapter.

### 4.1 Off-Axis Projection

In the vast majority of games and 3d applications it is enough to rely on the default camera object provided by an engine or framework. However in the case of simulating reflection in a magic mirror application, I had to consider a special type of projection. The following explanation is heavily based on *Generalized Perspective* [22] by Robert Kooima, *Off-axis projection in Unity* [29] by Michel de Brisas and *Implementation of Generalized Perspective Projection on the Unity* [30] by Hiruma Kazuya.

The Unity3d camera class produces a set of matrices used for further processing in the rendering pipeline in order to translate points from world coordinates to screen coordinate system. But I could not use them as they put the user's eyes position directly in front of the screen.

Let us assume the plane with points  $p_a$ ,  $p_b$ ,  $p_c$  defines the screen. Let us suppose that a set of some vectors  $v_r$ ,  $v_u$  and  $v_n$  is an orthonormal basis of the screen space [figure 4.4].



Figure 4.1: Screen space definition [22].

These vectors can be calculated from the following formulas[22]:

$$v_r = \frac{p_b - p_a}{\|p_b - p_a\|}$$

$$v_u = \frac{p_c - p_a}{\|p_c - p_a\|}$$

$$v_n = \frac{v_r \times v_u}{\|v_r \times v_u\|}$$

In general the eye position  $p_e$  is projected on the center of the screen [figure 4.2]. Let us call this point a *screen space origin*. According to the task for this thesis, I need to implement a mirror-like projection, and to do so, I had to let the eye position not be fixed. From now on let us suppose that screen space origin can be located anywhere on the screen plane [figure 4.3].

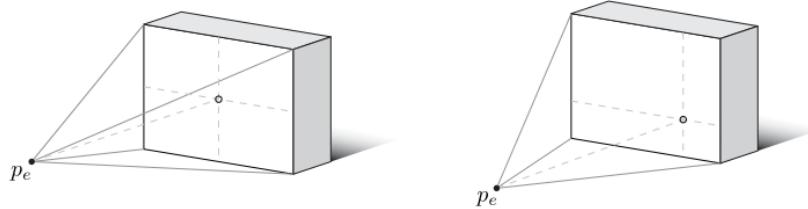


Figure 4.2: On-Axis projection [22].    Figure 4.3: Off-Axis projection [22].

Figure 4.4: On-Axis and Off-Axis projections [22].

Let us divide the sides of the plane according to the screen space origin and name the sides  $l$ ,  $r$ ,  $b$ ,  $t$  [figure 4.5]. Let us also add a parameter  $d$ , that stores the distance from  $p_e$  to the screen-space origin, and parameters  $n$ ,  $f$  for distances from  $p_e$  to near and far clipping planes.

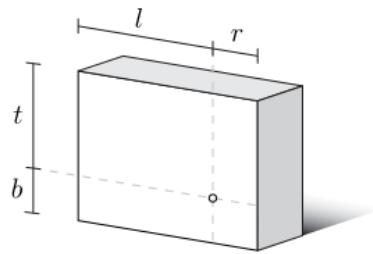


Figure 4.5: Screen division and parameters definition [22].

Assume non-unit vectors from  $p_e$  to the corners of the screen(  $v_a$ ,  $v_b$ ,  $v_c$ ) [figure 4.6].

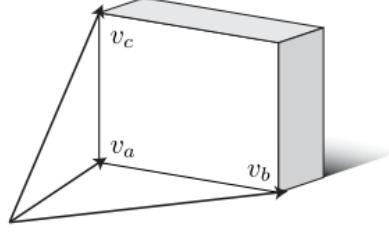


Figure 4.6: Vectors  $v_a$ ,  $v_b$ ,  $v_c$  introduction [22].

$$v_a = p_a - p_e$$

$$v_b = p_b - p_e$$

$$v_c = p_c - p_e$$

These parameters are calculated in the following way:

$$d = -(v_n \cdot v_a)$$

$$l = (v_r \cdot v_a) \frac{n}{d}$$

$$r = (v_r \cdot v_b) \frac{n}{d}$$

$$b = (v_u \cdot v_a) \frac{n}{d}$$

$$t = (v_u \cdot v_c) \frac{n}{d}$$

And use them in Perspective projection matrix[22, p. 4]:

$$P = \begin{vmatrix} \frac{2n}{r-l} & 0 & \frac{r+l}{r-l} & 0 \\ 0 & \frac{2n}{t-b} & \frac{t+b}{t-b} & 0 \\ 0 & 0 & -\frac{f+n}{f-n} & -\frac{2fn}{f-n} \\ 0 & 0 & -1 & 0 \end{vmatrix}$$

Projection plane orientation matrix:

$$M^T = \begin{vmatrix} v_{rx} & v_{ry} & v_{rz} & 0 \\ v_{ux} & v_{uy} & v_{uz} & 0 \\ v_{nx} & v_{ny} & v_{nz} & 0 \\ 0 & 0 & 0 & 1 \end{vmatrix}$$

And view offset transformation matrix:

$$T = \begin{vmatrix} 1 & 0 & 0 & -p_{ex} \\ 0 & 1 & 0 & -p_{ey} \\ 0 & 0 & 1 & -p_{ez} \\ 0 & 0 & 0 & 1 \end{vmatrix}$$

Finally, all previous matrices are composed together:

$$P' = P \times M^T \times T$$

On practice, it was used in Unity to define `worldToCameraMatrix` and `projectionMatrix` fields of camera component. To determine the position of the camera, I used the position of the user's head reflected from the surface of the projection frame.

Assuming that the Kinect sensor is in the origin of the coordinates, by moving and rotating the projection frame in the scene, it is possible to match its position with the relative position of the screen and the sensor in the real world. This will achieve the effect of a "magic mirror".

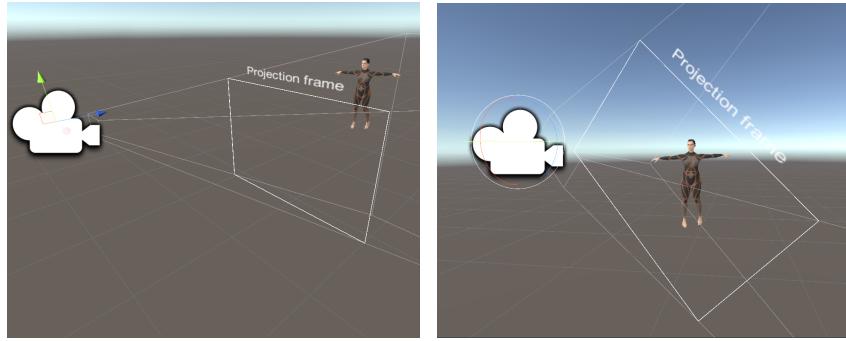


Figure 4.7: Projection frame in Unity.

## 4.2 Kinect Data Visualization

### 4.2.1 2D visualization

To accomplish this basic visualization method it was enough to ensure that the user's avatar is covering the user on the image obtained from the color camera of the sensor. To achieve this, I have placed a flat plane with a texture to the 3D scene and updated the texture with the output from the color camera of the Kinect every frame. With the correct placement of the camera, the plane and the avatar, the outfit will be covering the user's body from the camera output, creating the illusion that the user is wearing the clothes.

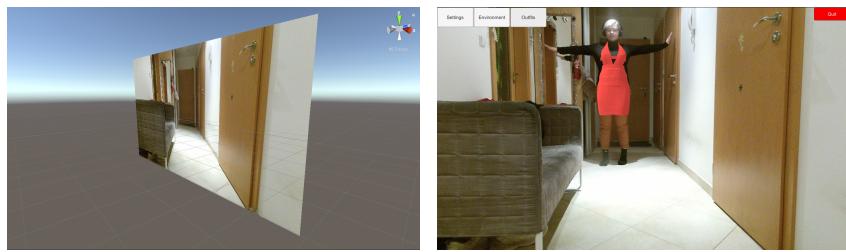


Figure 4.8: 2D visualization demonstration.

#### 4.2.2 Point cloud visualization

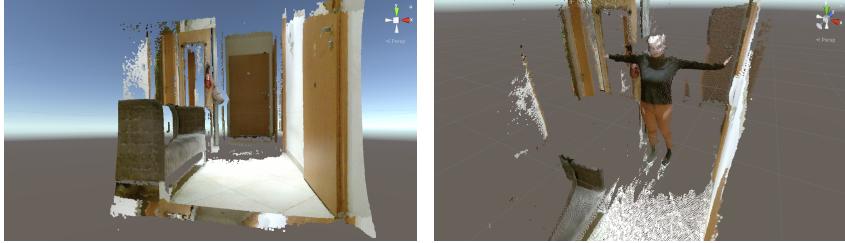


Figure 4.9: Point cloud demonstration.

To create a three-dimensional visual representation of Kinect data I decided to implement a point cloud [figure 4.9]. It was done by using Kinect SDK *DepthFrameReader* and *ColorFrameReader* classes to acquire and process data from sensors and *CoordinateMapper* class to project that data on 3D space.

|                 | Resolution | Framerate |
|-----------------|------------|-----------|
| Color camera    | 1920x1080  | 30        |
| Infrared camera | 512x424    | 30        |

Table 4.1: Kinect v2 sensor resolutions [31].

Let us first describe the way Kinect stores the data of each frame [table 4.1]. The frame is captured by the infrared camera has resolution of  $512 \times 424$  pixels, and Kinect internally calculates depths for each pixel in the form of a one-dimensional array.

Next, the point depth buffer was then converted using the *CoordinateMapper* into two new buffers: a 3D point array and a color camera image space coordinate array to use as UV coordinates.

The data from the first buffer were used as the coordinates of the Point cloud points, and the data from the second buffer were used to sample the color of this point from the image output of the color camera. To turn the points into squares, I used a geometry shader. This allowed me to achieve filling the gaps between the points by adjusting the size of the squares.

To further improve the image quality, I had to increase the number of points in the point cloud. However, for the number of points greater than what Kinect v2 can provide, it was necessary to interpolate their positions, this required additional processor resources. Additionally, I kept in mind that if the desired number of points is not a multiple of the original number of points, the implementation would be complicated.

To implement such approach I have decided to put the coordinate buffer data and the UV buffer data into textures. That way, the interpolation was done in the shader by sampling the texture.

It is a common practice to save normals as a texture. Acting on the same principle, I decided to write 3D space coordinates and UVs into textures. To

to avoid loss of precision, the texture format which stores color in float values (such as ARGBFloat or RGFloat) was chosen.

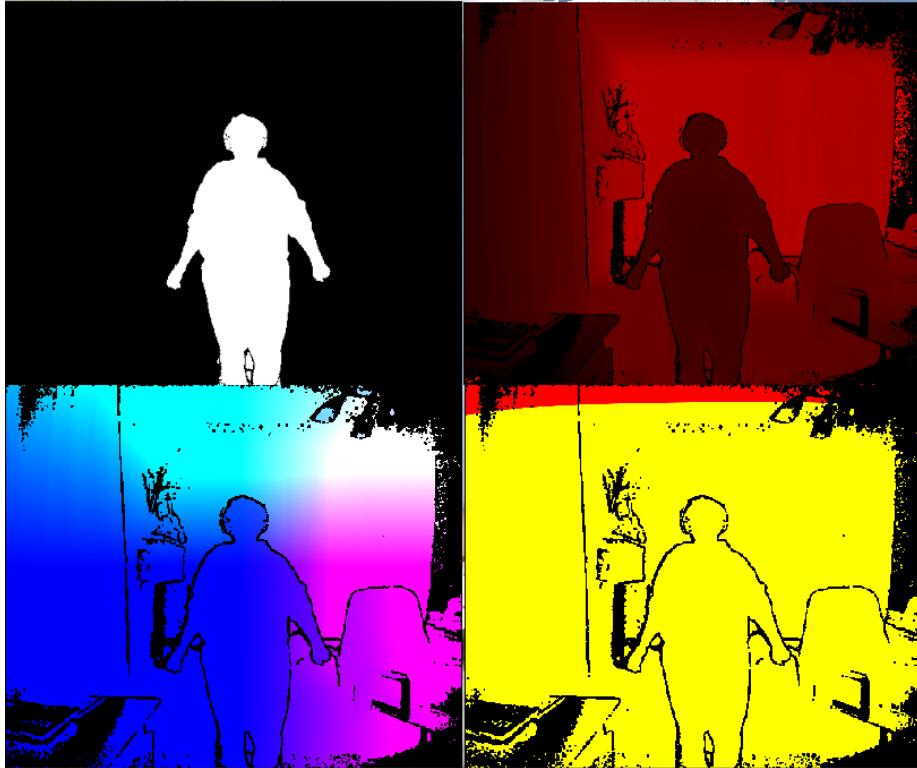


Figure 4.10: Body indexes, Depths, Positions, UVs as they will be rendered as textures

Next, to get a position and a UV coordinate of a point [figure 4.10], the coordinates for sampling are calculated as

$$\begin{aligned}x &\equiv idx \mod width \\y &\equiv idx \div width\end{aligned}$$

where  $idx$  is the point index and  $width$  is the number of points horizontally.

Thus, it was possible to set an arbitrary number (bigger, less or equal to the original number) of points in the point cloud. They will be evenly distributed and their coordinates will be interpolated between the values that Kinect can provide [figure 4.11].

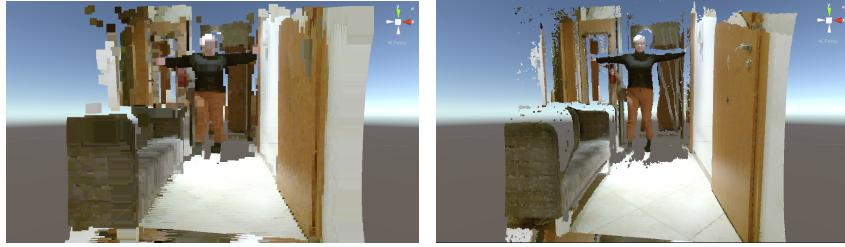


Figure 4.11: Point cloud density comparison. Left image: small number of points of bigger size; right image: bigger number of points of smaller size

There is a way that may improve the data visualization quality by using KinectFusion (described in detail in [16] and [32]) and scan the background environment in real time. This technology recreates the environment captured by Kinect cameras using 3D reconstruction of higher quality. This approach has not been implemented in my thesis yet, but it can become a possible future improvement of this application.

Kinect provides a body index buffer [figure 4.10], storing the index of the detected body; if there is no user body in that point, 0 is stored at this point. These data were used to remove the background from the point cloud.

To remove points not related to the user's body, body index texture was used as a stencil. Then, discarding those points where the index value is equal to zero, I have obtained the point cloud without a background [figure 4.12].

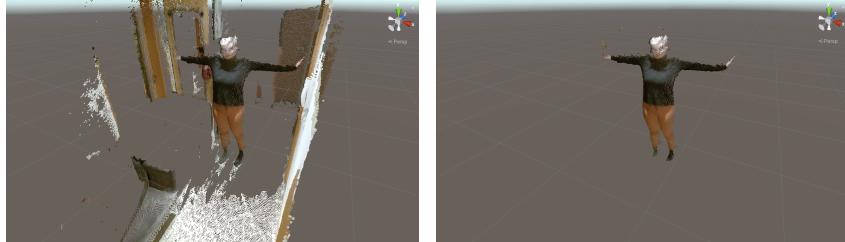


Figure 4.12: Background removal feature demonstration.

This allowed me to create custom backgrounds [figure 4.13] using free 3D assets from Unity Asset Store [33][34]. There are three of them in the application and the user can switch between them easily, these backgrounds are called "rooms" in the application.



Figure 4.13: Point cloud with removed background and 3D meshes in the scene.

### 4.3 Inverse Kinematics

With the help of the Kinect SDK, it was possible to access the position of the detected user and, as a result, the position of his joints. I could add a rigged 3D avatar to the scene and bind the position of its joints to the user's joints, but this leads to distortions of the avatar's geometry [3.5]. Due to the differences in the proportions of different parts of the user's body and the skeleton of the avatar, the latter will inevitably be deformed. Worse, due to inaccuracies in capturing the user's movement, the proportions of the Kinect's body are constantly changing by little. This was the reason why I decided to use inverse kinematics for avatar animation [figure 4.14].

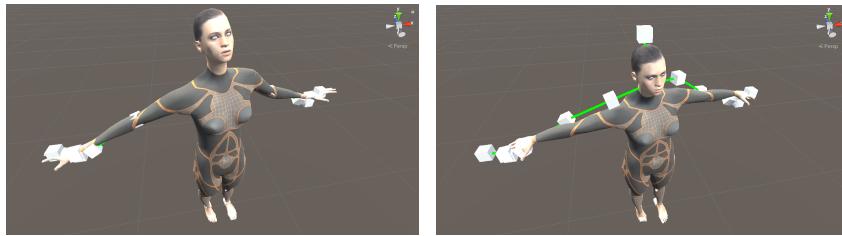


Figure 4.14: Animation without (left) and with (right) inverse kinematics.

#### 4.3.1 FABRIK Algorithm

In this implementation, it was decided to use the FABRIC algorithm without constraints. The constraints are increasing the execution time of the IK algorithms and there are also problems with deadlock when using the constraints,

i.e., a solution is never found because of the constraints[35], limiting the chain's rotation, even though a solution exists.

To describe the mechanics of the inverse kinematics, let us define some terms that are necessary for the following explanation. The point, to which the chain of joints is reaching, is called *the target*. *The end effector* is the last node in the chain which strives to target position. *The hint* is the point where the middle chain node aims.

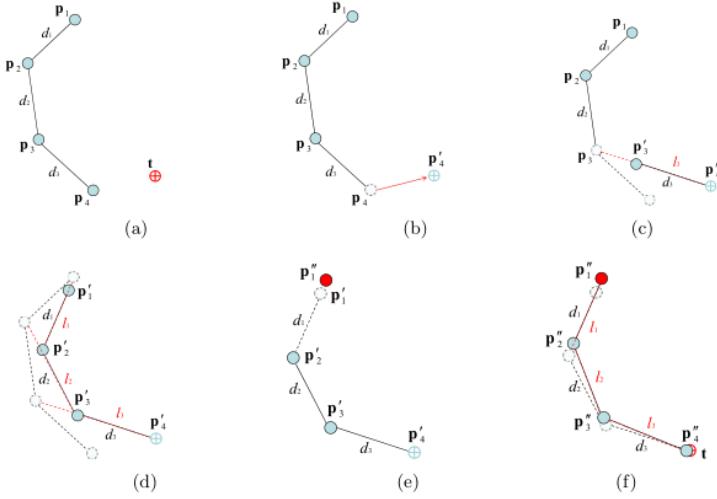


Figure 4.15: FABRIK algorithm. Image source: [26].

“An example of a full iteration of FABRIK for the case of a single target and 4 manipulator joints. (a) The initial position of the manipulator and the target, (b) move the end effector  $p_4$  to the target, (c) find the joint  $p_3$  which lies on the line  $l_3$ , that passes through the points  $p_4$  and  $p_3$ , and has distance  $d_3$  from the joint  $p_4$ , (d) continue the algorithm for the rest of the joints, (e) the second stage of the algorithm: move the root joint  $p_1$  to its initial position, (f) repeat the same procedure but this time start from the base and move outwards to the end effector. The algorithm is repeated until the position of the end effector reaches the target or gets sufficiently close“ [26] [figure 4.15].

I have considered positions of last joints, such as wrists and feet, as targets for limb effectors. In this thesis, I used IK chains to animate legs, arms and spine of the avatar. Orientation of pelvis and chest was calculated based on position of neighbour joints, as the three joints form a plane, and I could use its normal as a forward direction. Since the Kinect body data does not contain the orientation of the user’s head or any associated joints to calculate it (as it did for the pelvis and the chest), I was forced to use KinectFace plugin. And with its help, to match the rotation of the user’s face to the orientation of the avatar’s head.

The following image compares my implementation of FABRIK algorithm to default Unity IK [figure 4.16].

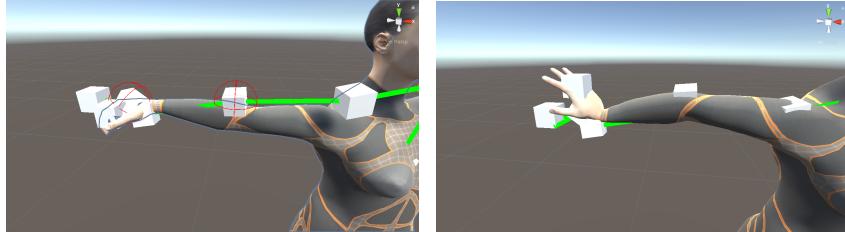


Figure 4.16: Unity inverse kinematics (left) compared to FABRIK (right).

In the future, it is possible to switch to the IK method with multiple end effectors (i.e. turn the torso using the IK branched chain)[25].

By applying FABRIK I have achieved the proper animation of the avatar based on the user's movements.

## 4.4 Outfit Rendering

When rendering clothes, it was necessary to take into account that the outfits had to be displayed on top of the user's image [figure 4.17]. Additionally, I had to hide the parts of the clothes that are not visible in reality; for example the insides of the clothes are covered by the person wearing them and are not seen in real life.

### 4.4.1 Rendering order

Point cloud and avatar with clothes meshes are placed on different render layers. Thus, the avatar and the clothes meshes are always rendered on top of the point cloud, each layer is rendered with its own camera separately [figure 4.18].

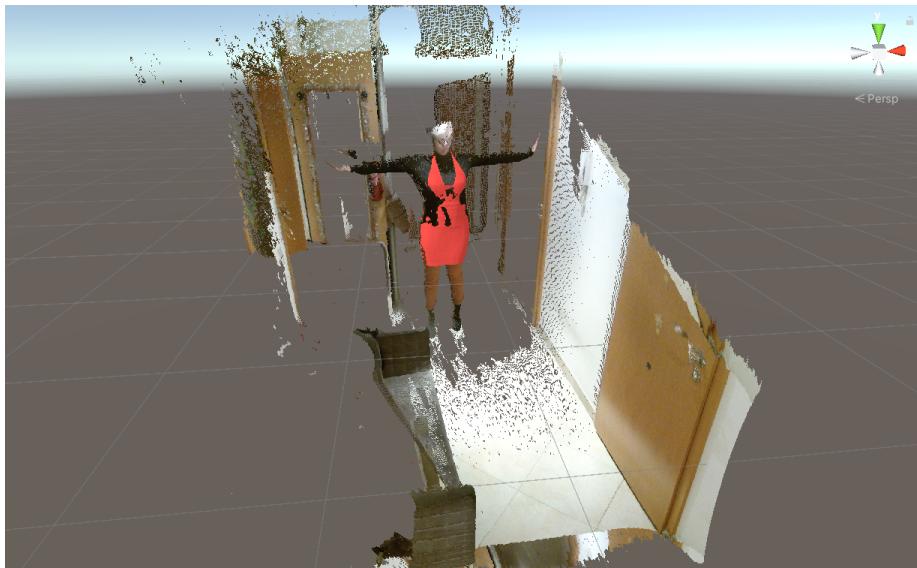


Figure 4.17: Outfit and the point cloud rendered on the same layer.

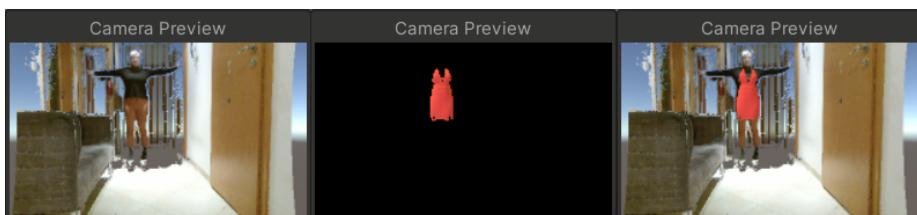


Figure 4.18: Different layers rendering. The last image is the result of combination of the previous two images.

#### 4.4.2 Usage of stencil buffer for outfits rendering

During clothes rendering the following issue occurred: when we draw a mesh of clothes on an invisible body, those parts that are closed by the body should not be visible.

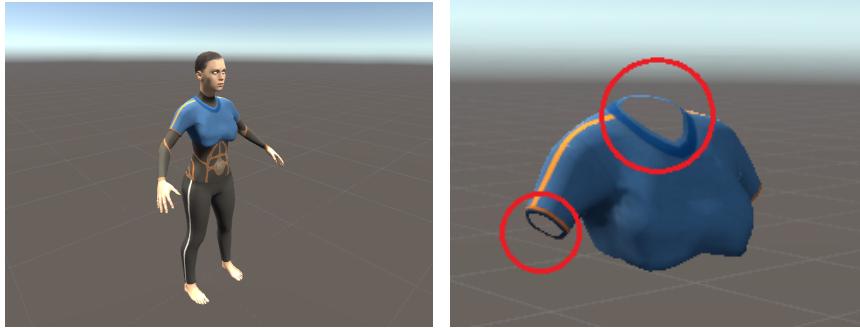


Figure 4.19: Clothes visualization issue.

At first glance, it seems I could rely on face culling, which imposes additional requirements on the geometry of the model. However, as you can see in the picture on right, there are triangles (faces) turned towards the camera, that must not be rendered, because they are positioned behind the “invisible” body.

The solution I came out with was to use the stencil buffer to set the correct stencil for rendering. To do this, I had to implement three different shaders, each for its own step.

The green color on the following pictures show the values of the stencil buffer after shader execution.

### 1. Rendering the clothes mesh with shader\_1.

- Stencil buffer is filled for the whole mesh;
- Mesh rendering is done in transparent color and writing to the depth buffer is enabled.

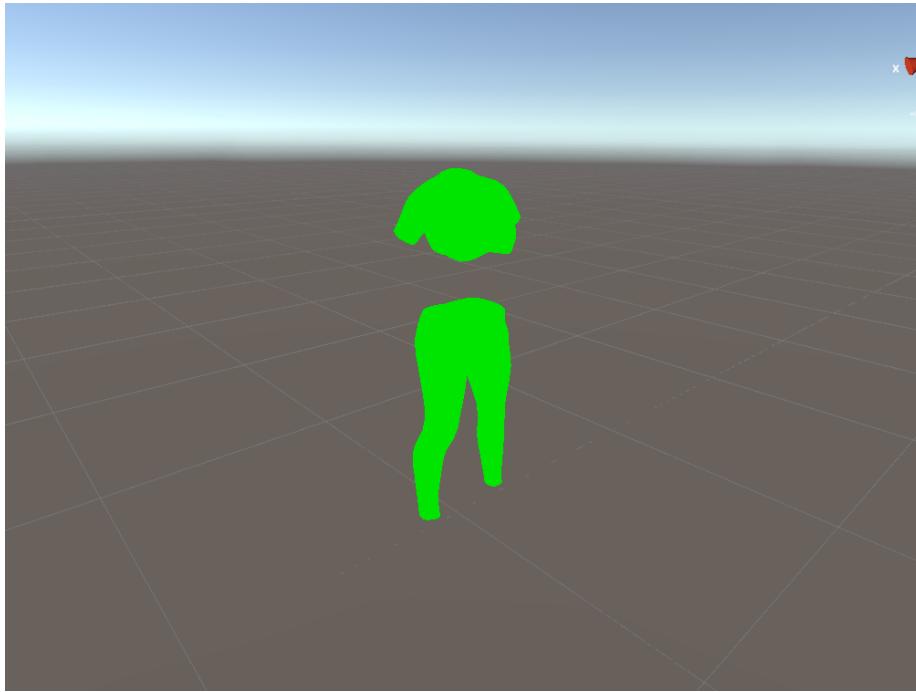


Figure 4.20: stencil buffer values after execution of shader\_1.

## 2. Rendering the clothes mesh with shader\_2.

- The current values of the stencil buffer are used as a stencil to prevent doing beyond its scope;
- Where the depth check passes (the body is closer to the camera than the mesh vertex), the stencil buffer value is set to 0;
- Writing to the depth buffer is disabled, because we do not want other objects in the scene to be overwritten with the “transparent background color” of the body.

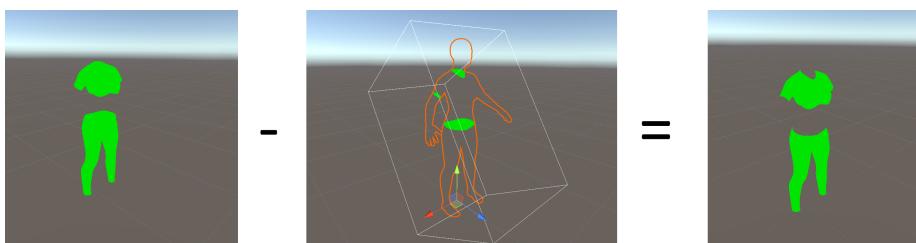


Figure 4.21: The final picture on the right represents the stencil buffer values after execution of shader\_2. The image in the center is placed here only for demonstration purposes, such buffer state never exists in the implementation.

### 3. Rendering the clothes mesh with shader\_3.

- Normal rendering using stencil buffer values.

**Final result:** only the "visible" parts of the mesh are drawn.

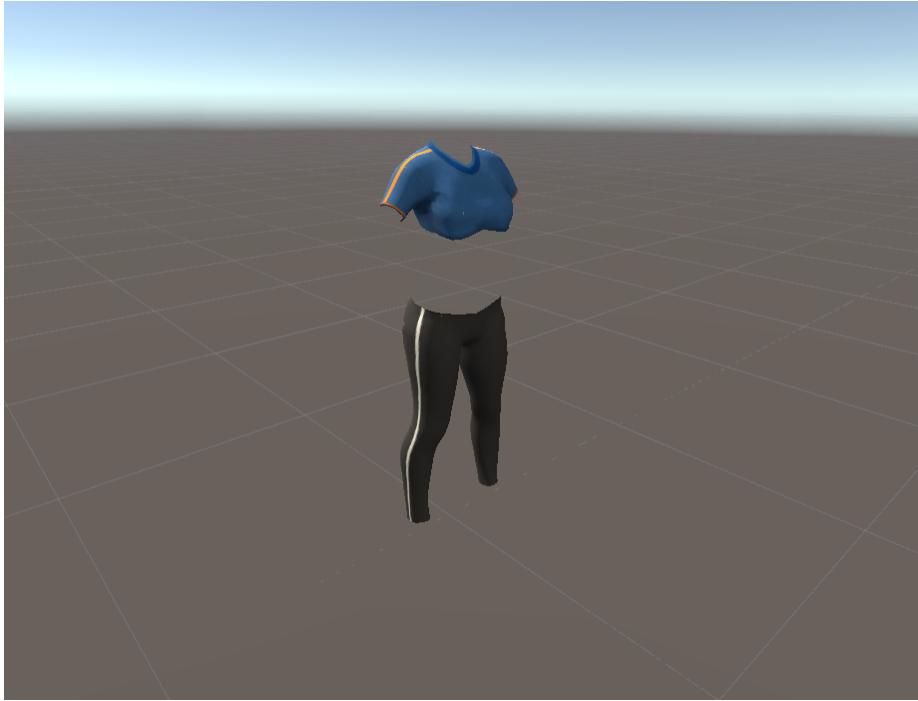


Figure 4.22: The outfit mesh rendered using stencil buffer.

#### 4.4.3 Avatar Attunement

Due to the variety of human bodies there is no way to prepare a universal avatar which will fit any user. Thus some kind of an attunement is required:

- in terms of height and constitution the simplest approach is just to scale the avatar;
- next step is stretching avatar bones to match the lengths of users limbs;

These avatar settings were implemented in the project; the scaling option and the bones stretching is computed in the application based on the user's joints positions.

## 4.5 Clothing Combination System

To fulfill the thesis task I had to implement a system, that would allow users to combine clothing pieces from different categories and try them on all together at the same time. This led to the following problems:

- the ability to wear one item of clothing over another (e.g. a jacket over a shirt);
- the body of the avatar should not be visible through clothing;
- clothes of the bottom layer should not be visible through the clothes of the top layer.

To generate different types of bodies and clothes for them, the Makehuman program was used. Furthermore, I used clothing models available for free use as well.

It was decided to divide all possible clothes into the following categories [figure 4.23]:

- hats
- gloves
- tops - lower layer
- bottoms - lower layer
- tops - upper layer
- bottoms - upper layer
- shoes



Figure 4.23: Clothes categories icons used in the application, icons sources: [36][37][38][39][40][41][42]

In this case, a particular piece of clothing can occupy several slots (categories) at once, for example, a business suit includes a shirt ( a top, lower layer), a jacket ( a top, upper layer) and trousers (a bottom, upper layer).

Thus, if the avatar is already wearing gloves, jeans and boots, and the user would like to wear a business suit, then the jeans will be removed. As a result, the avatar will be equipped with gloves, boots and a business suit.

#### 4.5.1 Clothes Layering

When the avatar moves, the body mesh may go through the mesh of the clothing, even if in the T-pose the clothing covered the body flawlessly without intersections. The MakeHuman application implements the removal of body faces in those places that are covered by clothes and should not be visible [figure 4.24]. For this thesis, I relied on meshes generated in MakeHuman and used this feature.

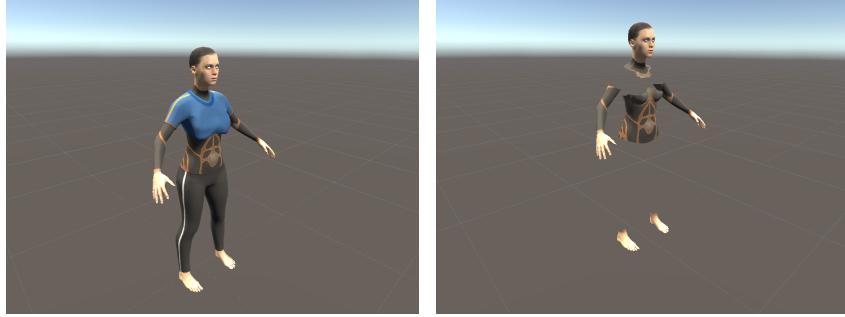


Figure 4.24: MakeHuman editing the body mesh.

Next, I had to solve the problem of layering clothes pieces on top of other clothes pieces, for example, combining two different objects: a shirt and a jacket, and prevent the 3D models from intersecting during animation. In general, the problem is complex and there is no universal approach for real-time applications [43].

The easiest way to make sure that different layers of clothing do not conflict with each other is to manually prepare a new mesh for each combination of clothing. Obviously, the number of combinations is rapidly increasing with the increase of the number of clothing items. For the purposes of the thesis, it was decided to prepare only combinations of (eg. shirts and jackets) for each of the body types. For  $n$  items of lower layer tops and  $m$  items of upper layer tops it is only  $n * m$  combinations.

This approach is implemented in the following way: after the user picks an item of clothing, a search is made among the prepared combinations. Thus, if the avatar is wearing, for example, jeans, a shirt and a jacket, and there is a prepared combination of these jacket and the shirt, the standalone objects of the jacket and the shirt will be replaced with the premade combination, and the jeans will remain intact.

#### 4.5.2 Clothing layering using stencil buffer

Each piece of clothing requires its own body mesh to correctly remove the parts that should not be visible. However, if several items of clothing are presented at the same time and, accordingly, several bodies, then faces removed from one body are replaced by faces from another body [figure 4.25].

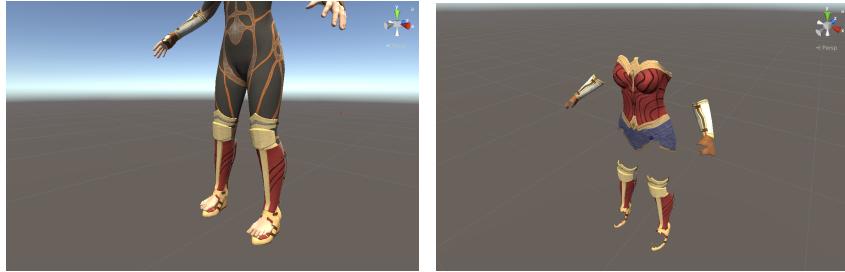


Figure 4.25: The toes are parts of the body provided by the piece of clothing other than the boots. This causes "holes" in the outfit.

To solve this issue, I have decided to store the category for each clothing piece in the form of flags. Thus, if the flag in binary representation 00000001 encrypts the top of the outerwear, 00000010 encrypts the top of the underwear, and 00000100 encrypts the bottom of the outerwear, then the business suit should have the category 00000111 [figure 4.26].

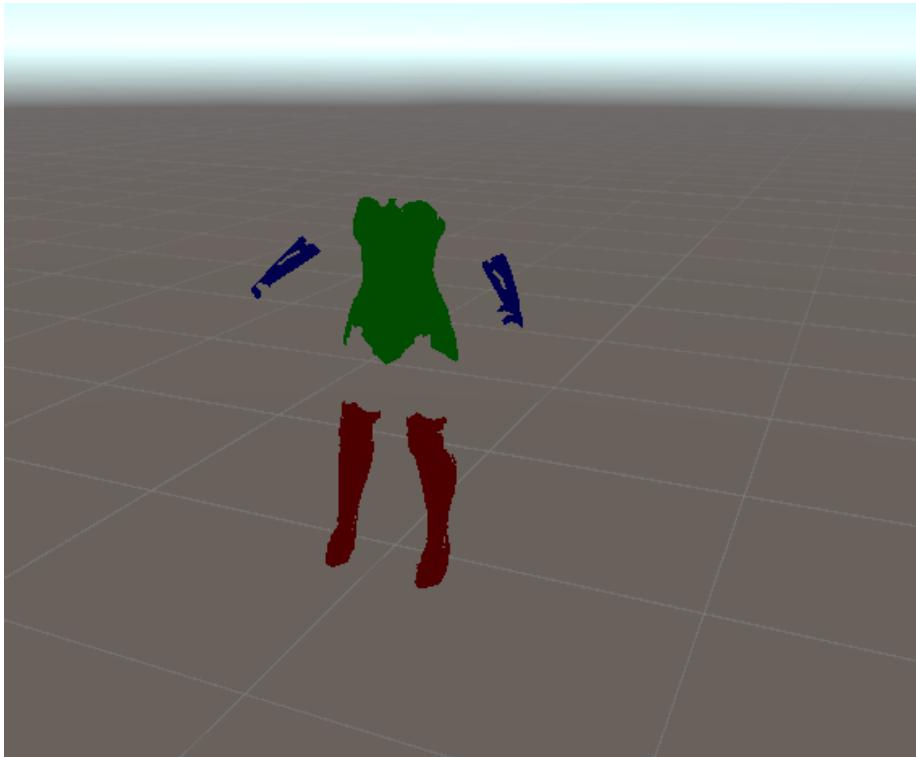


Figure 4.26: Demonstration of the stencil buffer after rendering each piece of clothing. Colors represent the state of the stencil buffer.

Since the stencil buffer allows bitwise reading and writing using a bit mask, I have used the bit representation of the clothing category as such. The stencil buffer cell size is 8 bits, which is enough for the needs of this work, since only 7

categories are used here. This approach allowed me to consider each bit of the stencil buffer individually and operate on the visibility of meshes of different clothing items independently of each other [figure 4.27].



Figure 4.27: The correct result of combination.

#### 4.5.3 Outfit Models

Most of the assets for this implementation were taken from MakeHuman application [44], that allows exporting human bodies and clothes [figure 4.28] and using them in non-commercial projects, some of the clothes assets were taken from MakeHuman community page [45].

These assets were used in the application for demonstration purposes, as I was not able to get access to any kind of historical clothing, that was recreated in 3D form using photogrammetry. However, the used 3D models could be replaced with the proper 3D reconstructed ones to make the application usable in the museum environment.

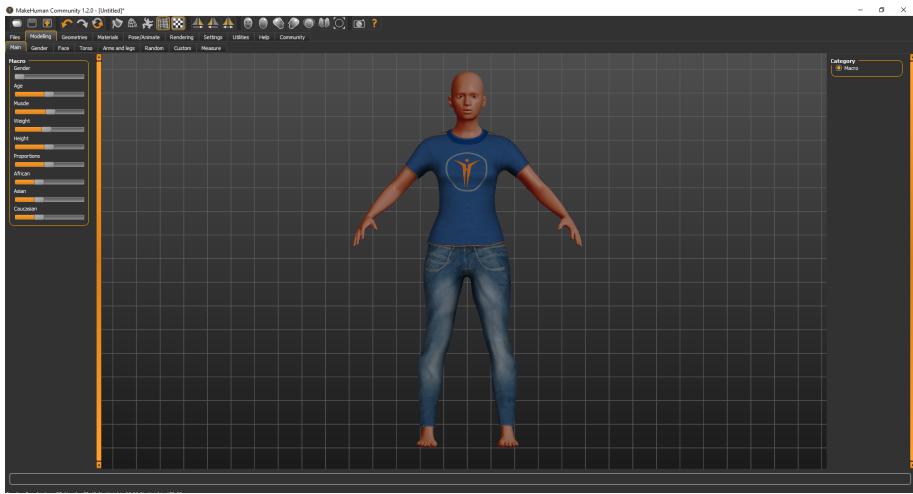


Figure 4.28: Example of an outfit from MakeHuman.

## 4.6 User Interface

At first, it was decided to implement a simple UI that is controlled by a mouse, all the panels, buttons and checkboxes were implemented using standard Unity User Interface design tools.

The UI consists of three main tabs:

- The Settings tab [figure 4.29] contains the camera settings (static camera with on-axis projection / dynamic camera with off-axis projection and a Field of view slider), debugging tools and hand cursors settings, that will be used for interactions with gestures.

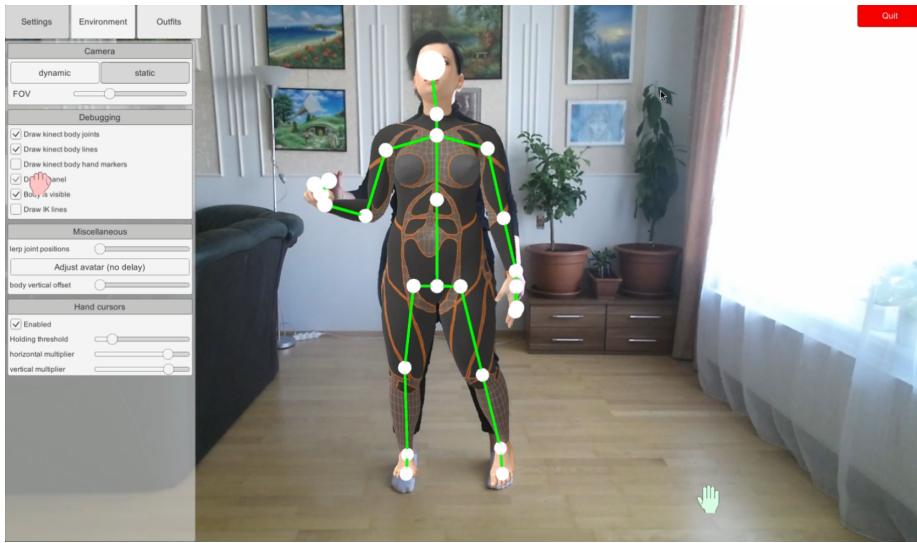


Figure 4.29: Settings tab demonstration.

- The Environment tab [figure 4.30] allows switching from 2D view to the Point cloud representation and contains Point cloud settings (the remove background feature and the background(room) selection feature).



Figure 4.30: Environment tab demonstration.

- The Outfits tab [figure 4.31] contains the avatar picking and adjusting features and the selection of clothing available for each avatar individually.



Figure 4.31: Outfits tab demonstration.

#### 4.6.1 Interactions With Gestures

Even though the implementation of this feature is basic and simple, it can be used as an additional way of interaction.

There are 2 pseudo cursors in the scene, one for each palm. The hand position is translated into screen coordinate space, then applied to the cursor. The state of the palm (open or closed) is interpreted as pressing the left mouse button [figure 4.32].

While implementing such an approach I have faced the following issues:

- The palm state is best detected closer to the center of the screen.
- Frequent false detection of closing the palm and low accuracy in determining the state of the palm in general.

I have solved them with the following ideas:

- The cursor position is dynamically shifted due to multiplication of the cursor's coordinate by a constant. Thus, the user could, being in the middle of the Kinect's field of view, reach with the cursor to the edges of the screen and interact with the interface elements.
- While the hand is continuously defined as closed, the time spent in the closed position is accumulated. If this time is more than a certain value, then left mouse button (LMB) pressing is emulated. If the hand was open before the threshold value was reached, then no clicking occurs, and the time recorded is reset.



Figure 4.32: Gestures interactions demonstration.

## 4.7 Playground room

To make the application usage more entertaining, I have decided to add a special room called the Playground [figure 4.33], where the user could interact with 3D objects.

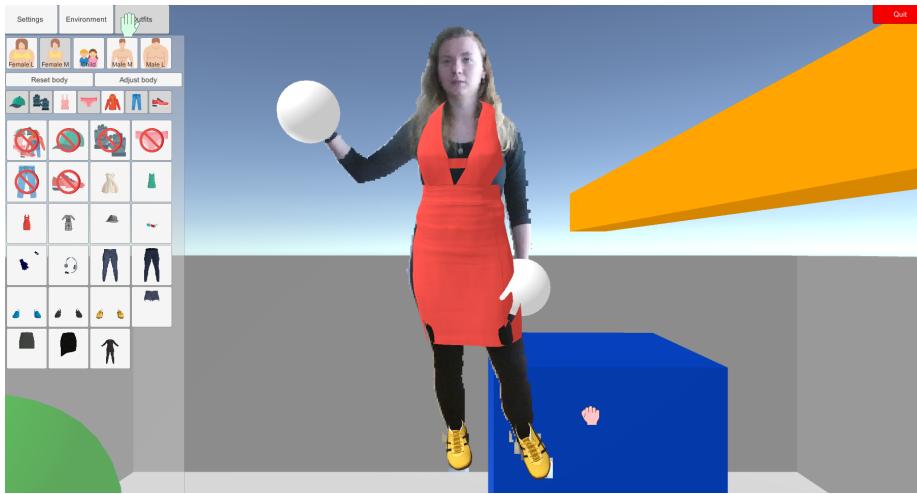


Figure 4.33: Playground room.

Currently there are two types of interactions available:

- If the user puts his palm inside the yellow cuboid, then random 3D primitives will start spawning in the scene until the user removes his hand.
- The user can grab any 3D object in the scene (except for the yellow cuboid) and move it around the space.

This specific room combines the possibilities of a virtual fitting room and an interactive game, allowing the users play it while wearing a virtual outfit.

## Chapter 5

---

# Application Testing

---

This chapter presents the results of the testing with people and an analysis of a technical side and performance of the application.

### 5.1 User testing

The application was widely tested by people with variant constitutions, the diversity of the testing group allowed me to analyze the down sides of the implementation and reveal bugs, that were not detected during the development process.

The users were asked to complete the following tasks:

1. Stand in front of the monitor, look at your mirrored image on the display. You have to see your body fully from head to toes.
2. Navigate to the Outfits tab using a computer mouse. Choose a preferable body type and adjust it to match your own body by clicking the Adjust Body button.
3. Select one piece of clothing from each category:
  - (a) top - lower layer
  - (b) top - upper layer
  - (c) bottom
  - (d) shoes
  - (e) hat
  - (f) gloves
4. Try moving around the room and/or posing.

5. Swap selected parts of clothing with any other ones that are available in the menu.
6. Try picking clothes with gestures. To do so, navigate to the preferred piece of clothing with a hand cursor and make a fist to click.
7. Either with a computer mouse or with a hand cursor navigate to the Settings tab. You may want to adjust hand cursors positions in the bottom section of the Settings tab.
8. Either with a computer mouse or with a hand cursor navigate to the Environment tab. Change Kinect Visualization type to Point cloud.
9. Using a PC mouse adjust points size and/or density to your liking.
10. Click on Remove Background check box and select the Futuristic 1 room. Navigate to the Outfits tab again and pick some clothes. Try to move around the area and/or pose.
11. Navigate back to the Environment tab and select the Futuristic 2 room. Repeat the try-on experience from the previous step.
12. Navigate to the Settings tab and select dynamic camera. Keep in mind that the gestures are not working in dynamic camera mode.
  - (a) Walk around the room to see, if this effect reminds you of the way mirror reflection works.
  - (b) Using a PC mouse navigate to clothes selection once more, pick some clothes.
  - (c) Navigate to the Settings tab and switch camera back to static.
13. Navigate to the Environment tab and select the Playground room.
  - (a) Put your right palm inside the big yellow object. This should spawn some random 3d primitives to the scene.
  - (b) Try to grab a 3d object with your hand.
14. You have now completed the test. To close the application, click on the Quit button.
15. Please share your thoughts on the application, give some criticism and suggest possible improvements.

Although the given tasks were meant to be completed in 30 minutes, most of the users have successfully finished them in lesser time with no hurry. The following sections are dedicated to each tester to summarize their experience with the program.

### 5.1.1 User #1

General personal information:

- |                    |                   |
|--------------------|-------------------|
| – Gender: Female   | – Height: 156 cm  |
| – Age Group: 20-25 | – Clothes size: M |

The user had difficulties with navigation in the application, she did not approve the UI layout, saying that it is not intuitive for a general user. She needed guidance during the testing process to complete the given tasks. The User #1 was frustrated with the clothes filtering system, she was also complaining about the precision on the body adjustment feature. According to this user, the selection of clothes in the current version is limited and the materials of the clothes are not obvious.

**Bugs and errors:** due to the user's height there were issues with the automatic body adjustment feature, the body was not scaling correctly and was shifted downwards. The problem was fixed right after this particular test.

**User's suggestions on future development:** the user would like to be able to manipulate with clothes, for example, to tuck in a t-shirt or any other top. On practice, that would mean adding physics for the fabric. The tester also suggested a feature to render user's hair on top of the clothing, if the user has long hair.

### 5.1.2 User #2

General personal information:

- |                    |                    |
|--------------------|--------------------|
| – Gender: Female   | – Height: 163 cm   |
| – Age Group: 50-55 | – Clothes size: XL |

The user[figure 5.1] needed guidance in the first half of the testing process, but after some time was able to navigate the application easily.

**Bugs and errors:** the user had difficulties with interactions using gestures; because this type of interactions is not precise, it was hard for this user to select clothes and click on buttons. The gestures system would definitely require to be upgraded.

**User's suggestions on future development:** the user sees the potential of such an application in the online shopping field and as a “big admirer of online purchases” would like to have a program like this, that would help her to buy clothes easily. She has mentioned that, in that case, it would also be helpful to see an image of a real model wearing the same piece of clothing.

### 5.1.3 User #3

General personal information:

- |                    |                     |
|--------------------|---------------------|
| – Gender: Female   | – Height: 169 cm    |
| – Age Group: 20-25 | – Clothes size: S/M |

The user had overall positive experience during the testing, she was able to complete all of the tasks quickly and effectively almost without guidance. The user enjoyed playing with the application in an unexpected way: out of curiosity she has chosen a child's body preset and adjusted it to her body size.

**User's suggestions on future development:** the user was surprised that when she turns around and stands with her back to the camera, the clothes do not turn with her. She would like to have such feature added to the project, if possible.

#### 5.1.4 User #4

General personal information:

- |                    |                     |
|--------------------|---------------------|
| – Gender: Female   | – Height: 162 cm    |
| – Age Group: 20-25 | – Clothes size: S/M |

The user was able to quickly adapt to the application interface and navigated through it using only interactions with gestures. The user enjoyed the playground and spent there more time than planned. She seemed interested in the details of the application implementation and asked questions about the development process.

**User's suggestions on future development:** a feature, that would allow users to fill in their body measurements manually and use them to adjust the avatar body more precisely.

#### 5.1.5 User #5

General personal information:

- |                    |                   |
|--------------------|-------------------|
| – Gender: Male     | – Height: 185 cm  |
| – Age Group: 30-35 | – Clothes size: L |

The user had a positive experience with the application, he complemented on the UI layout and gestures interactions, but had a small complaint about being unable to use gestures to interact with sliders. This user also was interested in the off-axis projection feature, but did not appreciate the inability to use gestures in conjunction with it.

**Bugs and errors:** the user complained about head tracking, mentioning that hats and glasses were not following the user's head correctly, compared to other clothing pieces.

**User's suggestions on future development:** an ability to manipulate with sliders with gestures.

### 5.1.6 User #6

General personal information:

- Gender: Male
- Height: 189 cm
- Age Group: 50-55
- Clothes size: L

The user [figure 5.2] needed guidance during the testing process, he had difficulties with navigation in the application. He was impressed with the body tracking quality, but had complaints about hands and feet body tracking, the shoes were not displayed correctly on the user's image most of the time due to Kinect v2 tracking limitations. The user also had complaints about head tracking.

**User's suggestions on future development:** hands and feet tracking improvement, more layers of clothes.

### 5.1.7 User #7

General personal information:

- Gender: Male
- Height: 172 cm
- Age Group: 25-30
- Clothes size: M/L

The user has tested body adjustment feature with all possible body types and compared the selection of clothes available for every body. He came to the conclusion that the clothes used in the project are "boring".

**User's suggestions on future development:** the user suggested adding futuristic sci-fi costumes or fantasy-styled armor sets with interesting animated visual effects (for example, fire effect on iron armor set would make it look like an enchanted item). He also wanted to see an expansion for the playground room.

### 5.1.8 User #8

General personal information:

- Gender: Female
- Height: 171 cm
- Age Group: 25-30
- Clothes size: XS/S

The user was dissatisfied with the quality of the clothes and with their size. Even though the outfits were mapped correctly to her body, it was obvious that the clothes had wrong proportions in certain areas and seamed too big for the tester. The User #8 complemented on point cloud visualization and the "remove background" feature.

**User's suggestions on future development:** the user asked for more sizes of the outfits.



Figure 5.1: User #2 testing the application.

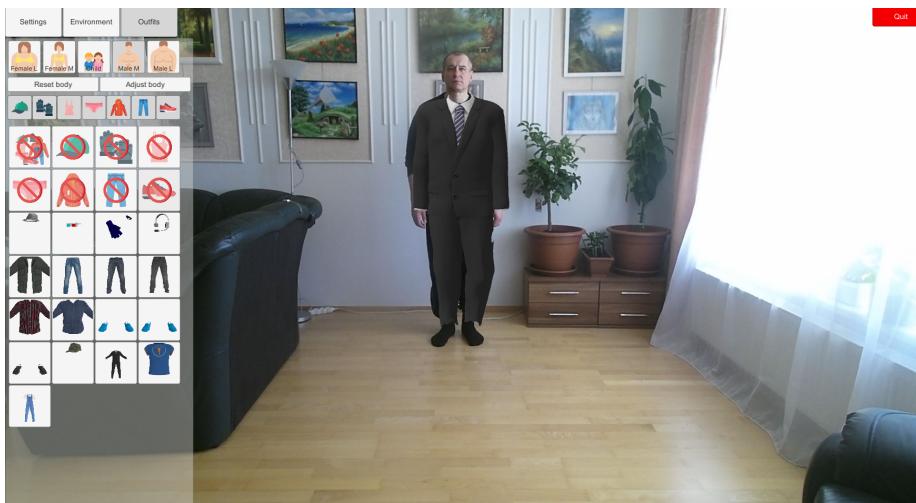


Figure 5.2: User #6 testing the application.

### 5.1.9 Multiple users test

The application was designed to be used by a single user and be stable with a situation, when more than one users appear in the field of view of the camera. To test the application behavior on this event User #2 and User #6 were asked to test it together at the same time.

When User #2 alone entered the field of view of Kinect cameras, Kinect has detected her body and the application recognized her as a current user assigning her an avatar. After this User #6 entered the area and was detected as a human by Kinect, but the application did not react to his appearance, allowing User #2 to continue using its features. As soon as User #2 left the field of view of

Kinect cameras, User #6 was recognized as the current user and was able to continue the try on experience started by User #2.

### 5.1.10 User Testing Conclusion

Based on the feedback I have received I came to the following conclusions:

- Even though the users were warned, that the 3D models used for the clothes in the current version of the application are there for demonstration purposes, some testers still were complaining about their quality. This proves the importance of high quality visualization for average users.
- The testers were not complaining about the quality of the point cloud, some of them even complemented it. The users also appreciated the ability to control its density. This could mean that the point cloud visualization does not need any further improvements.
- Most of the users call the avatar adjustment feature "not precise" in certain areas, this could be solved with increasing the selection of available body types.
- Out of 8 users only 1 was interested in the off-axis projection feature, that simulates mirror reflection. This fact may mean that mentioned feature is redundant for an average user.
- Due to basic implementation the interactions with gestures would definitely require additional work.
- Most of the users enjoyed the playground room, according to them, it was a "joyful and challenging experience". This means, the magic mirror application maybe not only useful, but also entertaining.

#### Current system limits:

- Only one user per time supported.
- Interactions with gestures do not work with sliders in the UI due to basic implementation.
- Interactions with gestures are not usable in dynamic camera mode.

## 5.2 Performance Testing

The application was tested on the following hardware for performance analysis:

| Hardware                       | FPS in Unity editor | FPS in built application |
|--------------------------------|---------------------|--------------------------|
| Intel i5 + Nvidia GTX 2070 GPU | 25-30               | 59-60                    |
| Ryzen 5 + Vega 11 iGPU         | 25-30               | 50-55                    |

Table 5.1: Performance test.

### Current system limits:

- The application requires Kinect v2 and Kinect SDK to work.
- The application requires OS with Kinect and Unity support.
- Graphics requirements: DirectX 10 or higher, OpenGL 3.2 or higher.

#### 5.2.1 Performance Testing Conclusion

The results of the performance testing are lacking, because I was unable to obtain more hardware for diverse testing. Nonetheless, the current results show us, that the application runs fluently on middle-end hardware

## **Chapter 6**

---

# **Conclusion**

---

In this thesis I have done a wide research on possible implementations of the virtual fitting room application, compared the possible solutions [2] and decided to carry out the implementation in Augmented Reality [2.3] with the conjunction of Kinect v2 [3.1] and Unity Engine [3.2].

I have divided the given task into individual sections, such as a "mirror-like" Off-axis projection [3.3], a data visualization [3.4] and a Kinect data visualisation [3.5], and analyzed the possible solutions for each problem to come up with the solution plan for the given task.

After this, I have adapted the already existing algorithms to solve the task of the thesis in the Implementation chapter [4].

Additionally, The final project was tested on different personal computers and with a testing group, the results of the testing were analyzed and listed in the Application Testing chapter [5].

---

## Bibliography

---

- [1] M. Moroz, “Tendency to Use the Virtual Fitting Room in Generation Y - Results of Qualitative Study,” *Foundations of Management*, 2019. doi: 10.2478/fman-2019-0020.
- [2] H. Lee, Y. Xu, and A. Li, “Technology visibility and consumer adoption of virtual fitting rooms (VFRs): a cross-cultural comparison of Chinese and Korean consumers,” *Journal of Fashion Marketing and Management ahead-of-print*, 2020. doi: <https://doi.org/10.1108/JFMM-01-2019-0016>.
- [3] KinectforWindows, *Kinect for windows retail clothing scenario video*. [Online]. Available: <https://www.youtube.com/watch?v=Mr71jrkzWq8&t=39s> (visited on 09/28/2020).
- [4] MagicMirror, *Augmented reality kiosk for museums*, 2007-2020. [Online]. Available: <https://www.magicmirror.me/Industry/Augmented-Reality-Kiosk-for-Museums/Augmented-Reality> (visited on 05/10/2021).
- [5] M. Szymczyk, *Ncma brings museum fashion exhibit to life with virtual dressing room tech*. [Online]. Available: <http://zugara.com/ncma-brings-museum-fashion-exhibit-to-life-with-virtual-dressing-room-tech> (visited on 05/10/2021).
- [6] T. Blum, V. Kleeberger, C. Bichlmeier, and N. Navab, “Mirracle: An augmented reality magic mirror system for anatomy education,” in *Virtual Reality Short Papers and Posters (VRW)*, 2012 IEEE, 2012. doi: <https://doi.org/10.1109/VR.2012.6180909>.
- [7] *Style club*. [Online]. Available: [http://www.styleclub.com.ua/model\\_wardrobe.aspx?SectionID=25](http://www.styleclub.com.ua/model_wardrobe.aspx?SectionID=25) (visited on 09/28/2020).
- [8] *Zeekit*, *Zeekit*. [Online]. Available: <https://play.google.com/store/apps/details?id=com.zeekit.client.android> (visited on 09/28/2020).
- [9] F. Inc., *Fit'n shop*. [Online]. Available: <https://play.google.com/store/apps/details?id=net.fxgear.fitnshop> (visited on 09/28/2020).
- [10] JCDecauxGroup, *Ar virtual fitting rooms for lily in shanghai metro — stdecaux*. [Online]. Available: <https://www.youtube.com/watch?v=zBKTZvy5r1A> (visited on 09/28/2020).
- [11] Fitnect, *Fitnect kinect - virtual clothes changing application*. [Online]. Available: <https://www.fitnect.com/> (visited on 09/28/2020).

- [12] U. Erra and V. Colonnese, “Experiences in the Development of an Augmented Reality Dressing Room,” 2015. DOI: [https://doi.org/10.1007/978-3-319-22888-4\\_35](https://doi.org/10.1007/978-3-319-22888-4_35).
- [13] M. Monir, N. H. Siddique, and N. E. Hashem, “An Efficient Magic Mirror Using Kinect,” 2016.
- [14] A. Masri, “Virtual Dressing Room Application,” 2019. DOI: 10.1109/JEIT.2019.8717410.
- [15] I. Pachoulakis and K. Kapetanakis, “Augmented Reality Platforms for Virtual Fitting Rooms,” *The International journal of Multimedia & Its Applications*, 2012. DOI: <https://doi.org/10.5121/ijma.2012.4404>.
- [16] D. Schmalstieg and T. Höllerer, *Augmented Reality: Principles and Practice*. Addison-Wesley, 2016.
- [17] *Kinect for windows sdk 2.0*. [Online]. Available: <https://www.microsoft.com/en-us/download/details.aspx?id=44561> (visited on 09/02/2020).
- [18] L. Caruso, R. Russo, and S. Savino, “Microsoft Kinect V2 vision system in a manufacturing application,” *Robotics and Computer-Integrated Manufacturing*, vol. 48, pp. 174–181, 2017. DOI: <https://doi.org/10.1016/j.rcim.2017.04.001>.
- [19] *Unity game engine*. [Online]. Available: <https://unity3d.com> (visited on 09/02/2020).
- [20] *Unreal engine*. [Online]. Available: <https://www.unrealengine.com/> (visited on 09/02/2020).
- [21] *Kinect 4 unreal project*. [Online]. Available: <https://www.opaque.media/kinect-4-unreal> (visited on 09/02/2020).
- [22] R. Kooima, “Generalized Perspective Projection,” *J. Sch. Electron. Eng. Comput. Sci*, 2009.
- [23] Wikipedia, *Point cloud*. [Online]. Available: [https://en.wikipedia.org/wiki/Point\\_cloud](https://en.wikipedia.org/wiki/Point_cloud) (visited on 11/12/2020).
- [24] J. Cummings, *Monmouth castle point cloud, created with photosynth*. [Online]. Available: <https://search.creativecommons.org/photos/a15ddd47-a94b-4654-b4d8-3dee48d7d221> (visited on 05/20/2021).
- [25] M. Parger, J. H. Mueller, D. Schmalstieg, and M. Steinberger, “Human upper-body inverse kinematics for increased embodiment in consumer-grade virtual reality,” *VRST 18: 24th ACM Symposium on Virtual Reality Software and Technology*, vol. 23, pp. 1–10, 2018. DOI: <https://doi.org/10.1145/3281505.3281529>.
- [26] A. Aristidou and J. Lasenby, “Inverse Kinematics: a review of existing techniques and introduction of a new fast iterative solver,” University of Cambridge, Tech. Rep. 632, 2009. [Online]. Available: [https://www.researchgate.net/publication/273166356\\_Inverse\\_Kinematics\\_a\\_review\\_of\\_existing\\_techniques\\_and\\_introduction\\_of\\_a\\_new\\_fast\\_iterative\\_solver](https://www.researchgate.net/publication/273166356_Inverse_Kinematics_a_review_of_existing_techniques_and_introduction_of_a_new_fast_iterative_solver) (visited on 10/18/2020).
- [27] Wikipedia, *Inverse kinematics*. [Online]. Available: [https://en.wikipedia.org/wiki/Inverse\\_kinematics](https://en.wikipedia.org/wiki/Inverse_kinematics) (visited on 10/23/2020).

- [28] S. Aukstakalnis, *Practical Augmented Reality*. Addison-Wesley, 2017, ISBN: 0134094239, 9780134094236.
- [29] M. de Brisis, *Off-axis projection in unity*, 2019. [Online]. Available: <https://medium.com/@michel.brisis/off-axis-projection-in-unity-1572d826541e> (visited on 10/12/2020).
- [30] H. Kazuya, *Implementation of generalized perspective projection on the unity*, 2019. [Online]. Available: <https://medium.com/@michel.brisis/off-axis-projection-in-unity-1572d826541e> (visited on 10/12/2020).
- [31] O. Wasenmüller and D. Stricker, “Comparison of Kinect V1 and V2 Depth Images in Terms of Accuracy and Precision,” 2016. DOI: [https://doi.org/10.1007/978-3-319-54427-4\\_3](https://doi.org/10.1007/978-3-319-54427-4_3).
- [32] M. Bengtsson, *Indoor 3D Mapping using Kinect*, 2014. [Online]. Available: <https://www.diva-portal.org/smash/record.jsf?pid=diva2%3A716061&dswid=9850%7D> (visited on 10/12/2020).
- [33] unity\_bfvNwgs4mDhzgw, *Free lowpoly scifi*. [Online]. Available: <https://assetstore.unity.com/packages/3d/environments/sci-fi/free-lowpoly-scifi-110070> (visited on 02/21/2021).
- [34] BarkingDog, *3d free modular kit*. [Online]. Available: <https://assetstore.unity.com/packages/3d/environments/3d-free-modular-kit-85732> (visited on 02/21/2021).
- [35] J. Bornold and J. Svantesson, *A realtime adaptation of inverse kinematics for motion capture*, 2015. [Online]. Available: <https://hdl.handle.net/20.500.12380/219734> (visited on 11/22/2020).
- [36] Smashicons, *Icon: Cap free icon*. [Online]. Available: [https://www.flaticon.com/free-icon/cap\\_1974214](https://www.flaticon.com/free-icon/cap_1974214) (visited on 05/02/2021).
- [37] Surang, *Icon: Gloves free icon*. [Online]. Available: [https://www.flaticon.com/free-icon/gloves\\_939216](https://www.flaticon.com/free-icon/gloves_939216) (visited on 05/02/2021).
- [38] Monkik, *Icon: Undershirt free icon*. [Online]. Available: [https://www.flaticon.com/free-icon/undershirt\\_2405872](https://www.flaticon.com/free-icon/undershirt_2405872) (visited on 05/02/2021).
- [39] Freepic, *Icon: Underwear free icon*. [Online]. Available: [https://www.flaticon.com/free-icon/underwear\\_4371685](https://www.flaticon.com/free-icon/underwear_4371685) (visited on 05/02/2021).
- [40] Smashicons, *Icon: Jacket free icon*. [Online]. Available: [https://www.flaticon.com/free-icon/jacket\\_705676](https://www.flaticon.com/free-icon/jacket_705676) (visited on 05/02/2021).
- [41] Monkik, *Icon: Jeans free icon*. [Online]. Available: [https://www.flaticon.com/free-icon/jeans\\_776623](https://www.flaticon.com/free-icon/jeans_776623) (visited on 05/02/2021).
- [42] Freepic, *Icon: Running shoe free icon*. [Online]. Available: [https://www.flaticon.com/free-icon/running-shoe\\_933695](https://www.flaticon.com/free-icon/running-shoe_933695) (visited on 05/02/2021).
- [43] T. Buffet, D. Rohmer, L. Barthe, L. Boissieux, and M.-P. Cani, “Implicit Untangling: A Robust Solution for Modeling Layered Clothing,” *ACM Transactions on Graphics*, vol. 38, pp. 1–15, 2019. DOI: <https://doi.org/10.1145/3306346.3323010>.
- [44] T. M. team, *Makehuman*. [Online]. Available: <http://www.makehumancommunity.org/> (visited on 09/28/2020).

- [45] MakehumanCommunity, *Makehuman clothes*. [Online]. Available: <http://www.makehumancommunity.org/clothes.html> (visited on 12/22/2020).
- [46] A. Aristidou, Y. Chrysanthou, and J. Lasenby, “Extending FABRIK with model constraints,” *Computer Animation and Virtual Worlds*, 2015. doi: <https://doi.org/10.1002/cav.1630>.
- [47] F. Rumen, *Kinect v2 tips, tricks and examples*, 2015. [Online]. Available: <https://rfilkov.com/2015/01/25/kinect-v2-tips-tricks-examples/> (visited on 09/08/2020).
- [48] S. Sneha, *Point cloud — post-processing of point cloud data*. [Online]. Available: <https://prototechsolutions.com/cad-notes/point-cloud-visualization/> (visited on 11/19/2020).
- [49] *Kinect for windows sdk documentation*. [Online]. Available: [https://docs.microsoft.com/en-us/previous-versions/windows/kinect/dn799271\(v=ieb.10\)](https://docs.microsoft.com/en-us/previous-versions/windows/kinect/dn799271(v=ieb.10)) (visited on 09/14/2020).
- [50] *Unity manual*. [Online]. Available: <https://docs.unity3d.com/Manual/index.html> (visited on 09/02/2020).
- [51] *Kinect sdk 2.0 documentation*. [Online]. Available: [https://docs.microsoft.com/en-us/previous-versions/windows/kinect/dn799271\(v=ieb.10\)](https://docs.microsoft.com/en-us/previous-versions/windows/kinect/dn799271(v=ieb.10)) (visited on 09/02/2020).
- [52] *Deprecated features (direct3d 10)*. [Online]. Available: <https://docs.microsoft.com/en-us/windows/win32/direct3d10/d3d10-graphics-programming-guide-api-features-deprecated> (visited on 09/02/2020).
- [53] *Blender the free and open source 3d creation suite*. [Online]. Available: <https://www.blender.org/> (visited on 09/06/2020).
- [54] *Rigify addon for blender*. [Online]. Available: <https://docs.blender.org/manual/en/latest/addons/rigging/rigify/index.html> (visited on 09/06/2020).
- [55] iconixar, *Icon pack: Diet and fitness*. [Online]. Available: <https://www.flaticon.com/packs/diet-and-fitness-7> (visited on 05/02/2021).
- [56] ——, *Icon: Kids free icon*. [Online]. Available: [https://www.flaticon.com/free-icon/kids\\_3460815](https://www.flaticon.com/free-icon/kids_3460815) (visited on 05/02/2021).
- [57] dDara, *Icon: Standing female free icon*. [Online]. Available: [https://www.flaticon.com/free-icon/standing-female\\_2789807](https://www.flaticon.com/free-icon/standing-female_2789807) (visited on 05/02/2021).

## Appendix A

---

### Contents of the enclosed flash disk

---

```
Build .....the directory of the application executable.  
Source .....the directory of source files.  
    |---UnitySource .....the directory of Unity source files.  
    |---LatexSource .....the directory of Latex source files.  
ThesisText .....the directory, containing thesis .pdf file.  
Videos .....the directory of video demonstration of the application.  
Contents.pdf .....the file with contents description.  
Readme.pdf.....the file with instructions for application user.
```