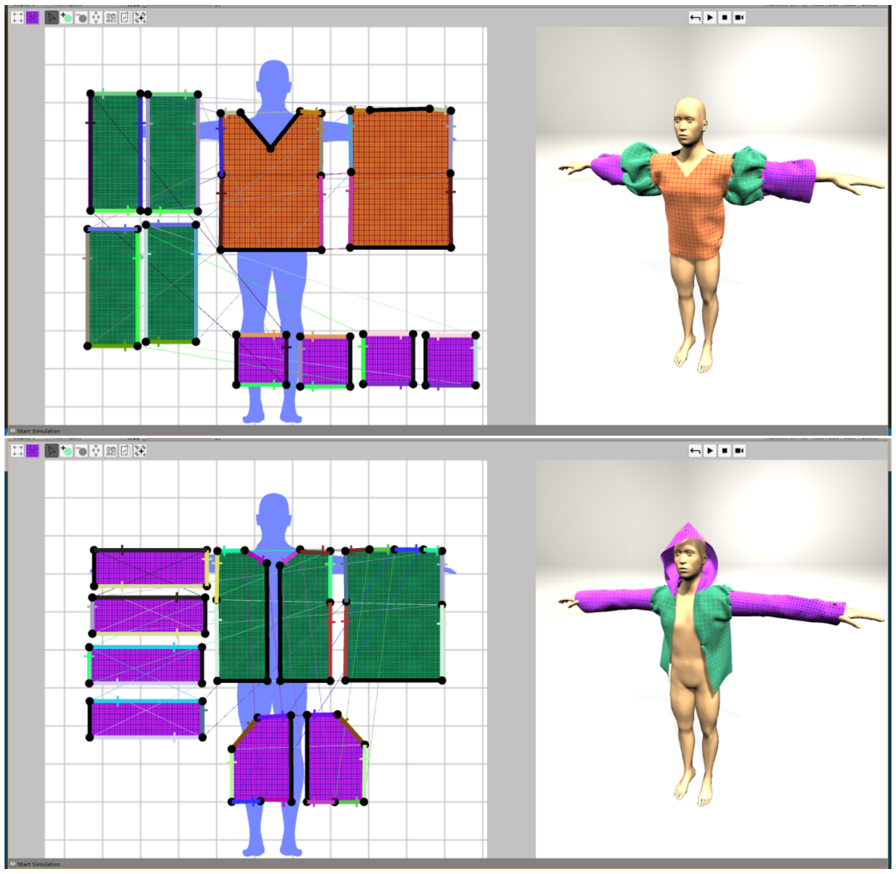

Vibbi - An Interactive Garment Modeling Tool

What to consider when making a interactive garment modeling tool

Master's Thesis in Interaction Design & Technologies

ANTON FREUDENTHALER, MALIN THELIN

Department of Computer Science and Engineering
CHALMERS UNIVERSITY OF TECHNOLOGY
UNIVERSITY OF GOTHENBURG
Gothenburg, Sweden 2018

Master's Thesis 2018

Vibbi - An Interactive Garment Modeling Tool

What to consider when making a interactive garment modeling tool

ANTON FREUDENTHALER, MALIN THELIN

Department of Computer Science and Engineering
Division of Interaction Design
Chalmers University of Technology
University of Gothenburg
Gothenburg, Sweden 2018

Vibbi - An Interactive Garment Modeling Tool
What to consider when making a interactive garment modeling tool
ANTON FREUDENTHALER, MALIN THELIN

Supervisor: Marco Fratarcangeli, Computer Science and Engineering
Examiner: Staffan Björk, Computer Science and Engineering

Master's Thesis 2018
Department of Computer Science and Engineering
Division of Interaction Design
Chalmers University of Technology and University of Gothenburg
SE-412 96 Gothenburg
Telephone +46 31 772 1000

Cover: Two screenshots of Vibbi, an interactive modeling tool, in action. To the left are the pattern designs which sewn together becomes the garment simulated on the right.

Typeset in LaTeX
Gothenburg, Sweden 2018

Vibbi - An Interactive Garment Modeling Tool
What to consider when making a interactive garment modeling tool
ANTON FREUDENTHALER, MALIN THELIN
Department of Computer Science and Engineering
Chalmers University of Technology
University of Gothenburg

Abstract

Designing garments is an iterative process which includes pattern creation, draping, and tailoring, which are difficult skills to master. Garments have a 3D shape but are created from patterns which are 2D; this introduces a gap in the design process between prototypes and the finished result. It is believed that this can be shortened in part by visualizing both patterns and the corresponding garment in virtual space.

For over 30 years there has been research about digital cloth design [1]. Still, there are few tools in which the user can both model patterns and simulate the corresponding garment in an interactive way. To address this issue we: conducted a literature study in the area of CAD tools for garment design, explored related and available tools on the market, and created Vibbi, an interactive garment modeling tool. With Vibbi, a user can model differently shaped patterns, choose how these patterns should be sewn together, visualize the patterns together in a 3D environment, and finally simulate the patterns as sewn garments.

Although Vibbi is not a completely finished product the development of this high fidelity prototype has provided knowledge. Thus, our findings include considerations to take when creating an interactive modeling tool for designing virtual clothes. These considerations can be summarized as: what external tools to use, how to work, how to model patterns, how to sew patterns together, how to dress the avatar, and how to simulate the clothes. While these considerations are not the only ones needed to be taken when creating this type of tool, as they are heavily based on the specific development of Vibbi, we consider them a good start and guide for anyone who is considering creating this type of tool.

Keywords: Cloth Modeling and Simulation, Computer Aided Design, Interactive design.

Acknowledgements

We would like to thank our supervisor Marco Fratarcangeli for his help and advice during the course of the project, as well as for involving us in his research. We would also like to thank Pontus Pall for his support with the Deform Plugin.

Malin Thelin, Anton Freudenthaler, Gothenburg, June 2018

Contents

1

Introduction

Making garments is a profession almost as old as mankind itself [2]. Today, companies in the fashion industry make a significant profit from being good at designing clothes, taking user needs and requirements into consideration [3, 4]. In addition, there are also companies outside of the fashion industry interested in designing clothes. Namely companies that design clothes for virtual characters in video games, movies, and other media.

The skills required to create a garment, such as tailoring and pattern making, can take years to master. In essence, designing clothes means that one has to: understand how a sketch of a 3D object can be turned into 2D panels, and finally how these panels will act when sewn into the previously sketched 3D shape, see Figure 1.1. A daunting task that produces unintentional results, even by professionals; which leads to the process of designing clothes being quite iterative, and looking at Figure 1.1 most of these iterations occur between B and D, but also between B and F.

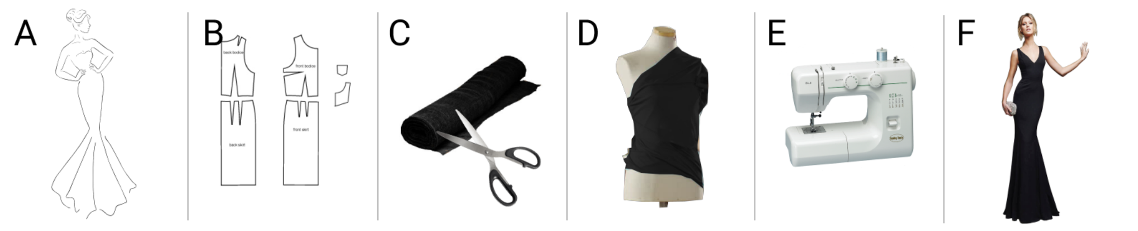

Figure 1.1: Illustration of the design process of clothes. A) Fashion designer creates sketch. B) Pattern maker creates pattern. C) Pieces are cut out. D) Pieces are put on a mannequin and altered. E) Sew final pattern pieces together. F) Garment can be worn.

Naturally, performing certain steps in this process - e.g., cutting out cloth and sewing it together - requires a lot of time and material, especially if one is forced to do it several times in order to reach a fulfilling outcome. Thus, the gap between the prototype and the finished result of a garment needs to be bridged. One possible solution to this problem is to use the aid of computers to design patterns in combination with physically correct simulation of the resulting garment.

1.1 Background

The computer has provided invaluable aid for countless different industries such as the automotive, construction, and fashion industry. Today, in the fashion industry, *computer aided design* (CAD) software is used. Mainly to create the 2D patterns in an exact, savable, printable, and replicable manner; more seldomly, the CAD software can show the result of the design on a character.

Using CAD improves the garment design process by for example being able to optimize the amount of fabric needed to cut out all the pieces of cloth. When it comes to showing the result on a character, the whole process of cutting the fabric and sewing it together can potentially be skipped, something that saves a lot of time in the concept development.

In addition to existing CAD software, research into new and different tools to aid in the garment design process have been going on for the last 30 years [1], and is still highly relevant. Recent approaches such as Sensitive Couture [5] and DressUp [6] have each provided different insights into how the process can be improved with the help of software; exploring the possibilities of simulation at interactive rates, bidirectionality, as well as utilizing our real physical space in order to design garments.

With recent breakthroughs in fast simulation of soft bodies [7], such as cloth, together with an ever growing interest in tools that can improve the process of designing clothes, there exists many possibilities to further explore this area.

1.2 Purpose & aim

Designing garments today is done in a mostly analogue way, with some aid from computers, which introduces a gap between the prototypes and the finished result. It is believed that modeling patterns, using CAD, while simulating garments at interactive rates can narrow this gap. The reason being that simulation while modeling is bringing prototypes and the result into the same medium which will shorten the amount of iterations needed in the design process. It is also believed that simulation while modeling can help designers gain intuition about the design space [5]. Also worth mentioning is that for the virtual garments to be seen as actual results it is important that the simulation is reliable and physically correct.

Thus, the aim of the project is to create and design Vibbi: an interactive modeling tool for designing virtual clothes; in order to explore:

What should be considered when making an interactive modeling tool for designing virtual clothes?

The purpose of Vibbi is primarily to provide practical feedback of the difficulties of creating a CAD tool for garments but potentially it can also be used to explore what interactive, bidirectional modeling of clothes teach designers about the transformation between patterns and real garments.

1.3 Outline

Chapter 2: Technical Background delves deeper into the following topics: the design process of garments, the research area and previous work on CAD tools for garments, simulation of garments, and the tools that were used during the project: Unity and Vivace.

Chapter 3: Methodology elaborates on agile software development methods which ties in with how the work were conducted during the project. Topics include: Scrum, Feature Driven Development, Rapid Application Development, Pair Programming, Version Control Systems, and Kanban.

Chapter 4 Process explains what the process looked like during the project from start to finish. The steps include: doing a Literature Study, Tool Exploration, Implementation Planning, and finally Implementation.

Chapter 5: Implementation goes through each phase from the Implementation Plan (which can be found in Appendix A) and explain how and what has been implemented in each phase. To be noted is that while the Implementation Plan consists of six phases, the Implementation chapter goes through four phases.

Chapter 6: Results presents different garments created using Vibbi in order to showcase its ease of use, flexibility, and correctness in simulation. The chapter also explains step by step how the design process in Vibbi works.

Chapter 7: Discussion discusses Vibbi, the process, the tools used, the possibilities and requirements of CAD tools for garments, and lastly future work. Finally, chapter 8: Conclusion concludes this thesis.

2 Technical background

In this section the technical concepts behind the project will be explained. First off: the design process behind creating a piece of garment in a more general sense without the involvement of computers. Then different approaches using computer aided design, including available tools and different digital garment modeling concepts such as Tangible Modeling, Sketch-Based Editing and Interactive Editing will be more thoroughly explained. After that, a pipeline for interactive simulation of cloth and garments is explained step by step. Finally, tools that was used during the project like Vivace and Unity are shortly presented.

2.1 Designing clothes

What will be described here is one typical process of designing and creating a piece of garment (see Figure 1.1) out of many. Designing a piece of garment can be done individually or by a group of people with or without the aid of a computer. In order to both:

1. Provide the reader with the necessary insight into the analogue garment design process.
2. Later draw comparisons to, and proper conclusions about computer aided design in garment design.

This text will explain the garment design process as performed by a group of people unaided by computers.

The people involved in creating a piece of garment are a fashion designer, a pattern maker, and a tailor. While these different roles can be, and often are, overlapping - e.g., one person can have two or all three roles - in a professional setting the roles are typically divided since each role requires a significant amount of expertise to perform.

Typically, the first step in creating a piece of garment is for the fashion designer to create sketches of the finished product. The fashion designer usually sketches a rough model of a human body and then draws the garment on top. Often incorporating some kind of pose or motion that showcases the garment in a lively manner.

The sketch is then given to and interpreted by the pattern maker. Their task is to translate this sketch of a 3D object into flat, 2D panels, *patterns*. These are consequently used as blueprints for cutting out the fabric which is then sewn together, by the tailor, to create the garment.

However, the step from sketch to patterns tend to not provide the most favourable of results. In actuality, the fashion designer and pattern maker will need to work iteratively together to reach a pattern shape that will truthfully turn into the initial, sketched design when sewn together. This iterative process is usually carried out by *draping*, a process where one cuts out the fabric pieces and put them up, with needles, on a mannequin to try to achieve the desirable result [8].

Here the fashion designer and tailor will together try to change the garment into what is desired and then the pattern maker can create new patterns based on these changes. Naturally, it is not unusual for the fashion designer to also update the initial sketched design based on insights earned from working with real fabrics. Eventually, this process will lead to satisfactory patterns which are sewn together into a finished garment.

2.2 Computer aided garment design

Fashion being a huge field, computer-aided garment design has grown into an industry of its own [5]; something that is evidenced by the myriad of CAD-tools available such as VStitcher, Marvelous Designer, GRAFIS, OptiTex (see Figure 2.1), StyleCAD, GeminiCAD, and PadPattern.

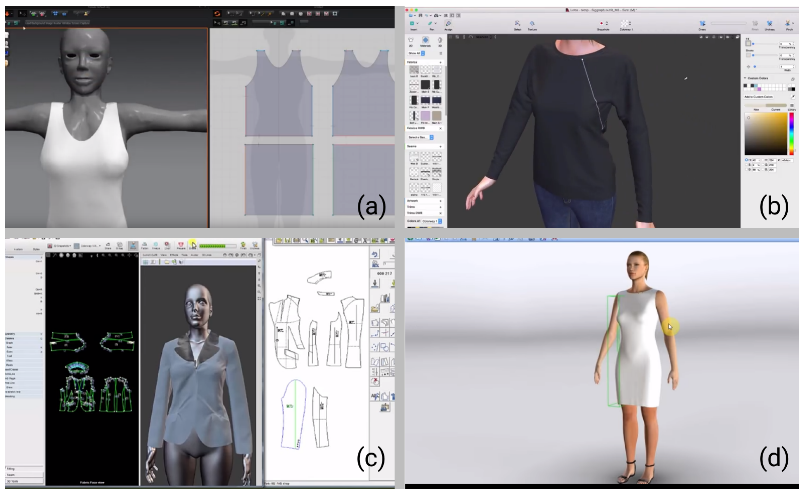

Figure 2.1: Four different CAD programs for designing garments. (a) Marvelous Designer. (b) VStitcher. (c) GRAFIS. (d) OptiTex.

VStitcher and Marvelous Designer are well established tools on the market for creating garments for virtual characters. These tools provide the user with both a 2D and 3D user interface, which allows the designer to bidirectionally, i.e., both in 2D and 3D, edit the clothes and then observe the result represented in the other dimension. This approach makes it possible for a designer to directly see the impacts of their edits, cutting down on the time between iterations. These tools are primarily made to provide the designer with the opportunity to explore, try, and design without having to care as much about time and material cost. A shortcoming with these tools is that the simulation they provide is not interactive, i.e., takes several seconds to compute, which interrupts the flow of the user. They can also be considered inaccessible for small business and individuals due to their cost and business models.

The other tools mentioned StyleCAD, GeminiCAD, GRAFIS, OptiTex, and PadPattern are more focused on creating patterns for the purpose of ultimately sewing the clothes in reality. Thus, some are purely for pattern drawing and grading (i.e., making different sizes for the same kind of garment), lacking the possibility of simulating the result on a 3D avatar - with a few exceptions. Ultimately providing the professional pattern maker with a comprehensible tool that proves more useful than drawing by hand, but at the same time not taking full advantage of the computer. Which, understandably, is due to both earlier and current lack in both computer- and software-capabilities to simulate cloth in 3D; a very computationally heavy task.

Beyond these different tools on the market, there is also a lot of research surrounding different approaches to further develop and improve the process of designing clothes. These different approaches have been dealt into three broader categories: Tangible Modeling, Sketch-based Modeling, and Interactive Modeling.

2.2.1 Tangible modeling

Tangible modeling offers a computerized, physical model in order to offer the user a more comprehensible way of designing clothes. In regards to iteratively improving the design and doing the creative work, one of the more important steps in the physical process of designing clothes is draping. Together with a digitalized, tangible model it is possible to offer the designer the prospect of using their creativity in draping together with the versatility of a computer.

One such approach is Dress-Up, which uses a physical mannequin as a 3D interface to design clothes [6]. To interact with the mannequin the users are supplied two tools - in the form of computer mice - one for drawing surfaces of cloth, and one for cutting the cloth. Thus, they interact with the tools on the physical mannequin and can at the same time see their results on the computer screen. On the computer, the GUI offers tools such as making seams, undo, and help with symmetry.

2.2.2 Sketch-based modeling

Sketch-based modeling can be conducted in many different ways. Typically, the goal is to generate virtual 3D models given 2D sketches. The process of sketch-based modeling can be broken down into three steps, namely, sketch acquisition, filtering and interpretation [9].

Sketch acquisition is the step of acquiring the sketch from the designer, be it that they sketched something on paper, a tablet, or with a mouse on a computer screen. If the sketch is drawn on paper for example, sketch acquisition would be the act of scanning the paper to bring the sketch in to the computer.

The second step includes noise reduction, which is meant to ease the process of the third and last step: interpretation. The interpretation is done by mapping the filtered sketch to a sequence of 3D modeling operations such as drawing points and lines at appropriate positions. The most difficult interpretation to accomplish is to figure out at which depth the points are.

One approach for a sketch-based interface for clothing virtual characters lets the user sketch clothes on the front and back side of a character separately, in order to later be able to generate 3D clothes by measuring the distances between the character and the sketch as well as the curvature of the sketched lines [10].

While the Sketch-based editing provides a designer with the opportunity to make their hand-drawn sketches into an immediate reality it provides little accuracy as the software has to approximate the sketches into something it can actually render on the screen. Additionally, hand-drawn illustrations tend to be, especially when compared to digitally created drawings, not symmetrical. As symmetry is essential in the world of designing garments, this is an issue.

2.2.3 Interactive modeling

Interactive modeling implies that the user is provided with immediate feedback of the actual result while the model is being edited. Compare it with document printing previews. Before printing the document, it is possible to see how the printed result will look like and thereby necessary changes can be made before valuable time and paper is consumed. Desirably, but not always necessary, the feedback should be provided as the product is being created, since it will save resources in each individual step. The reasoning becomes more apparent if the modeling is conducted in a different medium from which the actual result is going to end up in, such as in the case of pattern and garment making.

Patterns are drawn on paper or cloth, which is 2D, and garments are sewn and fitted for humans, which are 3D. Sensitive Couture is an interactive garment modeling and editing tool which simulates garments while they are being modeled and edited [5].

Plushie is another interactive design system which simulates the result, which in this case is a stuffed plush toy, as it is being designed [11].

To achieve interactive editing is to provide the user with a simulation that can update itself at interactive rates. Interactive rates is set at a minimum of 30 frames per second (FPS) and maximum of 60 FPS which gives the simulation ∼33ms and 16ms per frame respectively to do all of its calculations. Achieving this type of rapid simulation of cloth is a challenge addressed with several different techniques, some of which are described in the following section.

2.3 A simulation pipeline of cloth & garments

Simulation of cloth and garments can be done in different ways. In this section a simulation pipeline for achieving interactive simulation is described in order to provide the reader with necessary insights of the novel technology used in this project. This particular pipeline was chosen due to its ability to achieve interactive cloth simulation. Other pipelines exists but were not considered. Comparison of simulation methods is outside the scope of this thesis.

Simulation is a way to test things out without using the real world. This comes with some different, obvious advantages such as cost, time, safety, and ethics [12]. For example, in the real world, one cannot try out every possible configuration of how a house should be built - it would be too expensive. But with a simulation there is an opportunity to try out many more options, for figuratively no cost.

According to professor David. M Gaba:

> "Simulation is a technique... to replace or amplify real experiences... that evoke or replicate substantial aspects of the real world in a fully interactive manner [13]."

Simulation is used in many different areas such as health care [13], entertainment, education, manufacturing, communication, space technology, and economics. Considering the iterative nature of garment design, using simulation here is applicable as well.

To simulate a piece of garment, one first has to simulate a piece of cloth. The simulation of cloth is highly complex due to the properties and behaviour of fabric [14, 15], such as wrinkles, bending and shearing [16]. The next step of simulating a garment is to model the behaviour of cloth sewn together with other pieces of cloth.

There are three types of cloth-modeling techniques, geometrical, physical, and hybrid [14]. Geometrical techniques do not consider the physical properties of cloth. This gives them the advantage of being very fast, but inaccurate. The earliest visualization of cloth in computer graphics was done using geometrical approaches, such as the method presented by Weil in 1986 where they modeled clothes using Catenary curves [1]. Other geometrical based modeling methods include cylindrical folding [17], 3D panels [18] and mapped sinusoidal functions [19].

In contrast, physical techniques try to model the way cloth behaves in reality, something which can be done in many different ways, typically depending on whether there is a need for speed or accuracy. The first application for physical cloth simulation appeared in 1987 with the work of Terzopoulos et al., while the first garment simulations first appeared in the 1990s [15]. Physically based modeling methods include finite element [20, 21, 22], finite volume [23], elasticity [24, 25], particle system [26, 27, 28] and mass spring model [29, 30, 31, 32, 33, 34, 35].

Generally, in computer graphics and animation, appearance and visual realism is more important than physical accuracy [14]. However, with the purpose of simulating the behaviour of garments when put on a model, e.g., to visualize how the cloth drapes and folds, physical techniques have to be utilized. Thus, in this project a physical model is considered.

2.3.1 Physical cloth model

There are several physically based techniques to create cloth models. A relatively easy and pragmatic way is to use particle systems, position based dynamics, and to utilize mass-spring models [15]. These concepts will be further explained in the following sections.

A piece of cloth is usually constructed on computers as a 2D or 3D surface. Such a surface can be modeled as a geometric shape using Bezier curves or B-splines [36], which can more generally be interpreted as a polygon mesh [37]. Polygon meshes are built up by a set of vertices and edges in 3D space. A triangle is the most simple example of a polygon mesh, where the corners are the vertices and the lines between them are the edges. Thus, a more complex polygon can be constructed using a set of triangles (see Figure 2.2) and the process of doing so is called triangulation [38].

2.3.2 Triangulation

Triangulation is considered a trivial problem and several different algorithms for it has been described [38, 39, 40]. Instead, focus in research has been on creating fast algorithms and it has been proven that the lower bound on the time required to triangulate n points in two dimensions or higher is $\mathcal{O}(n \log n)$ [41]. More triangles means better looking cloth and more realistic simulations but it also comes with higher computational cost.

Sometimes in computer graphics it is also desired to have control over the size of the triangles being created in order to increase or decrease granularity [42]. De-

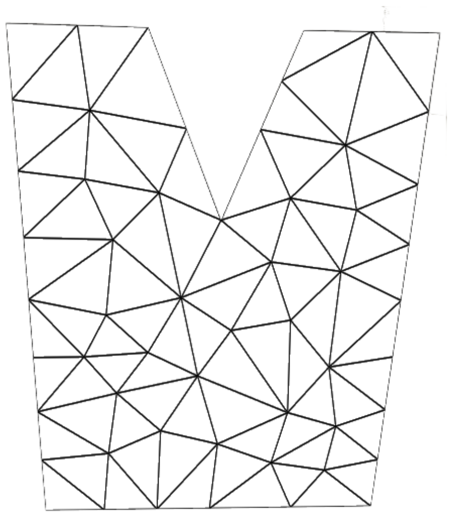

Figure 2.2: Triangulated mesh of a polygon, shaped like a tank top.

launay Triangulation is a constrained and robust triangulation that, in particular, avoid narrow triangles that can form when using naive triangulation algorithms [43]. There exist algorithms which constructs Delaunay Triangulation which also are asymptotically optimal [44].

2.3.3 Particle system in a cloth mesh

In a triangulated cloth mesh the vertices become the particles of the particle system (see Figure 2.3). Each of these particles are then considered as point masses, which are interacting with the neighboring particles using forces computed from their relative positions. Particle systems are among the simplest and most efficient ways to define rough models of cloth with relatively small computation times, while still providing visual realism [15].

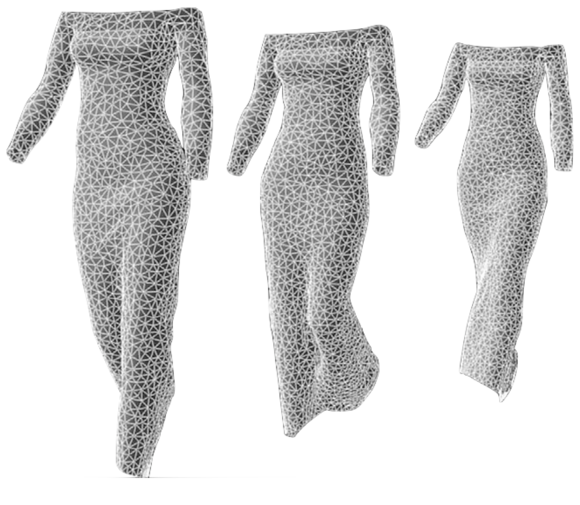

Figure 2.3: Particle system modelling a dress [15].

2.3.4 Position based dynamics

An approach presented by Müller et al. position based dynamics is a framework for modelling a constrained particle system [45]. Before position based dynamics was presented in 2007, one approach to solve the simulation of dynamic systems in computer graphics was to compute forces according to Newton's second law of motion (see Equation 2.1). Then applying these forces to the particles computing their velocity, and ultimately their new positions (see Equation 2.2).

$$F = ma \tag{2.1}$$

$$\begin{aligned} v &= v_0 + a\Delta t \\ x &= x_0 + v\Delta t \end{aligned} \tag{2.2}$$

In position based dynamics however, the approach is to omit calculation of the velocities, instead working directly on the positions of the particles. The main advantages of this approach is its controllability, easier collision handling, as well as providing a simple way to resolve penetration. According to Volino et al. collision detection is a bottleneck in the speed of cloth simulation [15].

2.3.5 Mass spring model

The Mass Spring Model is a physically based model which is proven to be useful for real time simulation of cloth [16]. It models cloth as a network of elastic springs which hold points of mass together. To determine how the points move the acceleration a is determined according to Newton's Second Law of motion (see Equation 2.1).

The mass m is given and the force F is calculated as the sum of internal and external forces that acts on the point. Internally, the points are pulled and pushed by the connected springs, which want to be in a state of rest. It is common that the internal forces are linearly approximated by using so called Hookean springs [16]. Hook's law states that the force F applied by a spring is directly proportional to the stiffness k and the displacement of the spring (see Equation 2.3). Externally, the points are pulled and pushed by different environmental conditions and actors, such as gravity and wind.

$$F = -k(L - L_0) \tag{2.3}$$

The Mass Spring Model is considered to have more accuracy than known geometrical models and also to be faster than other physical models [16].

2.3.6 Basic iterative methods for solving linear systems

Basically, clothes can approximately be physically simulated as systems of linear constraints. There exist plenty of ways to solve linear systems, among them some iterative methods which are frequently used in computer graphics since they tend to require less memory usage than direct methods. Two basic iterative methods for solving linear systems are Jacobi and Gauss-Seidel [46].

2.4 Vivace: A Practical Gauss-Seidel Method for Stable Soft Body Dynamics

Vivace is a recently presented solver for sparse systems of linear constraints, which employs a practical Gauss-Seidel method for stable soft body dynamics [7]. Vivace enables the animation of 3D soft objects discretized as large and irregular, triangular or tetrahedral meshes and has been demonstrated to be beneficial for parallel implementation of Projective Dynamics [47] and Position Based Dynamics [48], which are two methods that can be formulated as linear systems.

2.5 Unity

Unity is a game engine and a development platform with great support for creating 2D and 3D games. This includes a real time framework in the form of a *game loop*. One of Unity's strengths is that it allows for deployment to plenty of different platforms. Unity provides its user with easy to use, drag and drop design possibilities, as well as scripting using C#. It has a complete framework for rendering, applying physical conditions, and collision handling.

Unity also has its own asset store where extensions made by the public can be shared, which allows developers to reuse implementations done by others. More importantly, Unity models surfaces and 3D objects as triangulated meshes and includes plenty of functionality and documentation for creating a 2D or 3D modeling tool.

2.5.1 GameObjects & Components

Unity uses an architectural, programming pattern known as *Entity-component system* (ECS). The gist of this pattern is that all *objects* present in a scene are simply empty containers for different *components*. These components provide different types of functionality to an object, such as rendering, physics, collision detection etc. Consequently, ECS provides great flexibility when adding or removing functionality from objects.

In Unity the objects are called `GameObjects` and the components are simply known as `Components`. For example, a Component that every GameObject has is '`Transform`' which describes the object's position, rotation, and scale - see Figure 2.4.

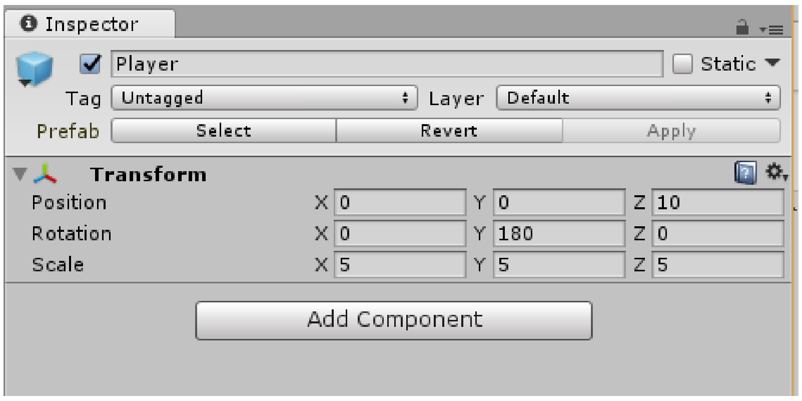
Inspector
Player — Static
Tag: Untagged — Layer: Default
Prefab: Select | Revert | Apply
Transform
	X	Y	Z
Position	0	0	10
Rotation	0	180	0
Scale	5	5	5

Figure 2.4: A GameObject known as 'Player' which only contains the Component 'Transform' which states that the GameObject is positioned ten steps in the Z-direction, rotated 180 degrees around the Y-axis, as well as scaled five times its original size in every direction.

Adding or removing Components from aGameObject, as well as changing the variables can be done both in the Unity editor as well as with C# scripts.

2.5.2 Game loop

Every GameObject has three main functions, `Awake()`, `Start()`, and `Update()`. Awake() and Start() is only run once in the lifetime of a GameObject - as soon as it is activated - while Update() is continuously called by the Unity game loop. Awakes are called before Starts, usually Awake() is not needed unless for example another GameObject's Start() function needs a variable initialized.

2.6 Deform Plugin

Vivace can be accessed in Unity through the `DeformPlugin`. The DeformPlugin provides the Vivace engine by exposing the API of the underlying technology. While there are eleven different classes provided with the plugin - see Figure 2.5 - only five of them are of interest: `DeformManager.cs`, `DeformBody.cs`, `DeformObject.cs`, `DeformCloth.cs`, `DeformCollider.cs`, and `MeshUtils.cs`.

The DeformManager is responsible for managing the simulation between Vivace and Unity. It manages this by keeping track of the gravity and wind of the simulation as well as all of the DeformBodies - i.e., objects created in Unity that have either DeformCloth or DeformObject as a Component -, and the DeformColliders. A

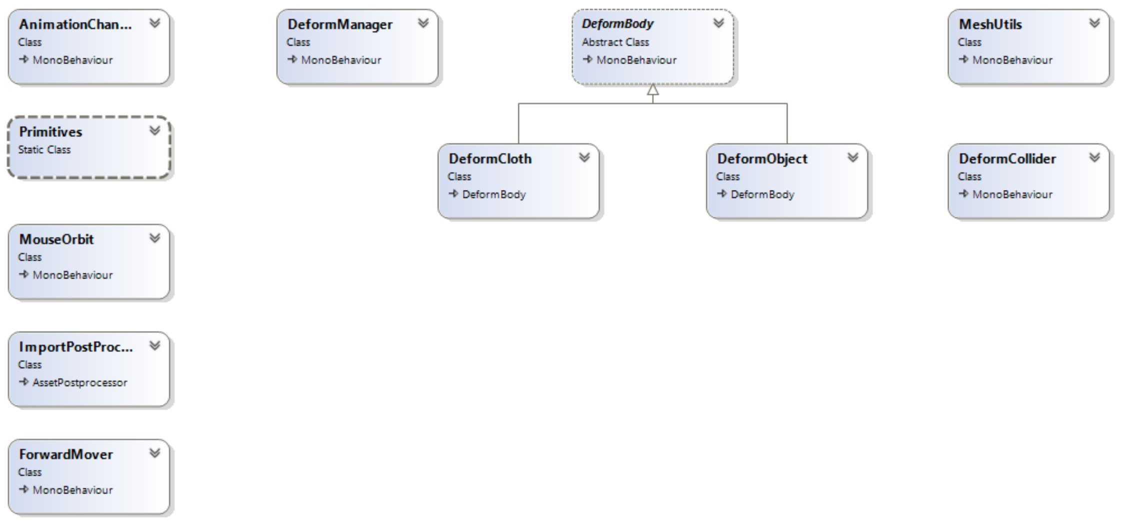

Figure 2.5: Class diagram of Deform Plugin, which exposes the functionality of the Vivace engine.

representation of the DeformBody and DeformCollider objects, i.e., their meshes, positions, rotation and scale is sent to Vivace. In turn, Vivace applies the set gravity and wind to all of the DeformBodies, solves the linear relationship between all the vertices in the scene, and then continuously sends back information to the DeformManager of their updated state.

The difference between a DeformCloth and DeformObject is the mesh. DeformCloth gets its mesh from `MeshUtils.CreateClothMesh()` which returns a rectangular 'patch' of a user defined size and resolution. In contrast, the DeformObject can have any type of mesh which can either be entered directly in the Unity editor or by using scripts.

A DeformCollider can be in the shape of either a box, sphere, capsule, or a plane. It is also possible to set the position, size, and rotation of the collider. As the name implies, a GameObject with the DeformCollider component will act as a collision object for any DeformBodies in the scene. The position, shape, and size is sent to Vivace at initialization, but as the colliders are static, rigid objects they are never updated after that.

3 Methodology

Being a software development project, the method used in the project is a combination of several different agile software development methodologies. Therefore, agile software development as well as the relevant methodologies using the same concepts therein will be described in this chapter.

3.1 Agile software development

Agile Software Development (ASD) is a category of software development methods that includes several different approaches, which have some common denominators that define them as agile. The most prominent of these denominators is that the developers work in iterations, usually short iterations that generally lasts one to four weeks. These iterations tend to include all the different steps needed to produce a full-scale piece of software such as planning, requirements analysis, design, coding, testing, and documentation [49]. Other denominators include: face-to-face communication, have the developers in the same location, and emphasize working software as the primary measure of progress.

Working in iterations provide several advantages to having one long software development life-cycle, such as in the counterpart to agile software development methods known as 'Waterfall' methods: see Figure 3.1. For example, it allows the developers to continually during the process get feedback from their customers, thus increasing the chance of ultimately creating a piece of software that the customer actually wants. Other advantages is the increased quality and polish that comes from continually planning and testing the product. Since the developers are provided with a much clearer focus due to short iterations they are more probable to produce more focused work as well. In conclusion, using ASD methods reduces the risk of producing poorly made software.

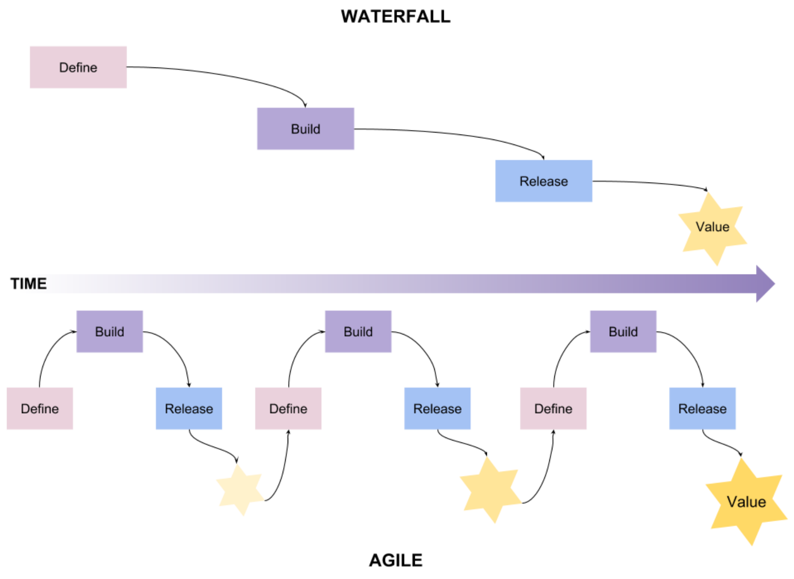

Figure 3.1: Illustrating the difference between a waterfall and agile working process.

3.2 Scrum

Scrum is a framework used to achieve ASD. Using Scrum consist of having a team, conducting certain events and managing certain artifacts [50]. The team consist of developers, who do the work of delivering a product, and a product owner, who is responsible for maximizing the value of the product delivered by the developers. To assist the organization, the developers, and the product owner, there should also be a Scrum Master who is responsible for supporting and promoting Scrum as it is described in the *Scrum Guide*.

The events conducted by the team are called Sprints, Sprint Plannings, Sprint Reviews, Sprint Retrospectives and Daily Scrums. Sprints are short time periods in which a usable increment of the product is delivered. During Sprint Planning it is decided what should be delivered in the upcoming Sprint. Sprint Reviews are conducted after Sprints to manage what is done and what potentially has to be continued in the next Sprint. During the Sprint Retrospective the team can evaluate their previous work process and decide upon improvements for the future. Daily Scrum is a short meeting where the work of the day is brought up.

During the process, a Product Backlog, which contains descriptions of what is to be delivered, and a Sprint Backlog, which contains detailed descriptions of what is to be delivered during the current Sprint, is managed to provide an overview. A Sprint Backlog is created during Sprint Planning by selecting a set of tasks from the Product Backlog.

3.3 Feature driven development

Feature driven development (FDD) is a feature-centric, ASD model with focus on design and construction of quality software [51]. Like other agile models, FDD is describing an iterative process where the final product is being built incrementally. As the name implies, the product is delivered feature by feature. A feature is a piece of functionality that brings value to the client, has an estimated cost, which also can be scheduled and prioritized [52]. Features are typically derived from requirements, user stories and use cases but also from emerging issues such as bugs or dependency changes.

The motivation behind FDD is to ease the effort of planning, managing and monitoring an ASD project. This is done by unifying the requirements and the planning with a feature-centric approach. User needs and requirements are defined as piece wise features which serves as work packages and tasks during development. The process is divided into fixed time periods ”timeboxes” in which a set of features are developed.

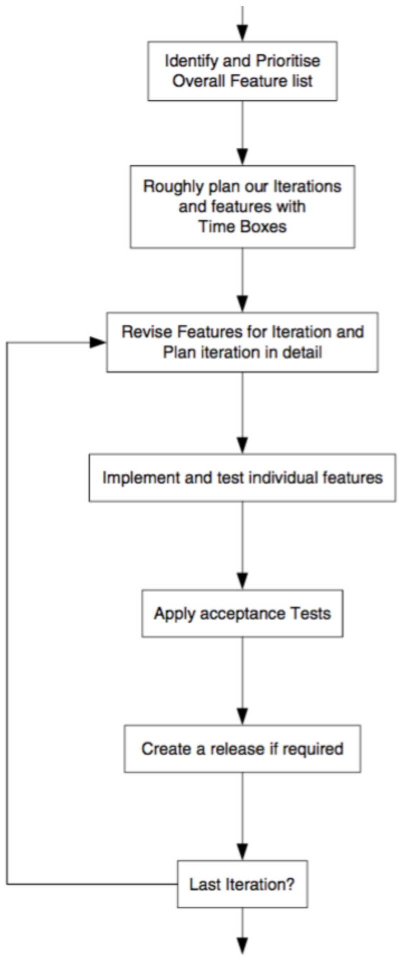
Identify and Prioritise Overall Feature list
Roughly plan our Iterations and features with Time Boxes
Revise Features for Iteration and Plan iteration in detail
Implement and test individual features
Apply acceptance Tests
Create a release if required
Last Iteration?

Figure 3.2: Overall structure of an FDD project [52]. Illustrating how features are revised, implemented and tested in iterations.

FDD consists of five phases:

1. Develop an Overall Model
2. Build a Features List
3. Plan by Feature
4. Design by Feature
5. Build by Feature

During the first three phases it is estimated how many time boxes is necessary to complete the project. This is made possible by determining the content, time needed and order of each time box. This includes figuring out a scope and context for the software project as a whole, as well as deriving specific features and grouping them. A set of features with similar priority will be the content of a specific time box. When the project reaches the designated start of a time box, the included features are designed according to their complexity and dependencies to later be built by coding and code inspection.

3.4 Rapid application development

Rapid application development (RAD) is an ASD approach to develop software, and just like ASD it puts less emphasis on planning and more on an adaptive process [49]. More specifically, RAD proposes that products can be developed faster and of higher quality by:

- Continuously producing prototypes of the intended software.
- Re-using software components.
- Following a schedule that defers design improvements to the next iteration.
- Keeping communication between team members informal [49].

Typically, when using RAD the developers will use a *Rapid Development Language* (RDL). These are programming languages that offers speedier implementation than do traditional third-generation languages such as C and C++ [49]. RDLs usually offer the developers built-in tools and libraries that speed up certain parts of software development, the drawback being that they tend to reduce the flexibility of what can be developed using the language.

3.5 Pair programming

Pair programming is a software development technique where programmers code in pairs rather than individually [53]. Typically, one person is coding while the other one is actively reviewing and suggesting different courses of action. This practice is believed to improve both software quality and learning [54].

3.6 Version control systems

Version control systems (VCS) allow software developers to collaboratively write code and handle revisions. It is an important aspect during any software development process, and especially when using agile methods. The primary tool used by most software developers for revision control is Git [55]. Other tools include SVN, Mercurial, and Perforce.

3.7 Kanban

Kanban is a way of organizing the work during a project, and synergizes well with ASD methods [56]. In essence, Kanban is a 'to-do' list that works well for larger projects with many subtasks. The progress and process of the project is visualized, usually in the form of Post-its fastened on a Kanban board. Each Post-it has a simple description of some task that needs to be done for the project. The tasks can be organized according to their feature and importance, giving the team members a strong visual representation of the project.

New tasks are added to the board according to the agile method used in the project, and tasks are assigned by the members picking a Post-it that they want.

4

Process

This chapter describes our process to explore CAD tools for garment design in order to find out how the design process of garments can be improved through digital tools. The project was performed for twenty weeks during which there was a literature study of related work and technology, an exploration of relevant tools, a plan made for the implementation of a high-fidelity prototype of a CAD tool for garment design: Vibbi, and finally the implementation of Vibbi.

4.1 Literature study

During the first four weeks of the project a literature study was performed. Performing the literature study meant reading about the fundamentals of computer techniques for fashion textile modeling as well as exploring earlier work done in simulating user designed clothes such as Sensitive Couture [5] and DressUp [6]. It also included getting an understanding for the process of designing clothes, as well as getting the bigger picture of how a physics engine doing soft body animation, more specifically cloth simulation, works.

The research area as well as most of these technological concepts have been explored earlier in chapter 2: Technical Background.

4.2 Tool exploration

Simultaneously while doing the literature study there was an exploration of available tools. Both tools that exhibited some parts of our potential end result - i.e., tools to design clothes - as well as tools to be used during the course of the project, such as Unity and Vivace.

4.2.1 Exploring related work

A couple of existing design tools were explored in order to gain knowledge about the state of art. As explained in section 2.2: Computer Aided Garment Design there are plenty of tools which specifically aid the process of designing the patterns for clothes, as well as a few that also try to simulate the sewn garments. Most of the tools were explored by reading corresponding papers and watching tutorials on YouTube, while for example Marvelous Designer was downloaded and used.

Using Marvelous Designer inspired the planned end result of Vibbi. As proven by the numerous case studies of its use in designing clothes for different games, the interface and user experience of Marvelous Designer is well designed [57]. As the nature of the project did not allow for a full development of the more user centric parts of Vibbi, taking inspiration from an already well established tool proved useful.

4.2.2 Unity

Unity provide their new users with several tutorials of different difficulty levels together with the software itself. They also keep an extensive library of various video tutorials on their website [58]. Thus, during the beginning of the project several of the built-in tutorials were completed in order to get a feel for how Unity works. During the course of the project the online video tutorials were regularly consulted.

4.2.3 Vivace & Deform Plugin

Provided with the Deform Plugin were a couple of test scenes. These test scenes showcased the capabilities of Vivace, such as the DeformManager, DeformBodies, and DeformColliders, see Figure 4.1. Exploring these test scenes provided insight into the details of how the Deform Plugin works and possibilities of transforming the functionality of the plugin into Vibbi.

As Vivace - and therefore also the Deform Plugin - was continuously updated during the course of the project, new test scenes and functionality was opened up. For example sewing capabilities and control over the friction between DeformBodies and DeformColliders did not exist until week 10 of the project.

4.3 Planning

A plan was made describing the things to accomplish during the twenty weeks of the project, see Appendix A. In accordance with FDD the time was planned into six different phases, see Table A.1. Each phase had their own topic to address and tasks

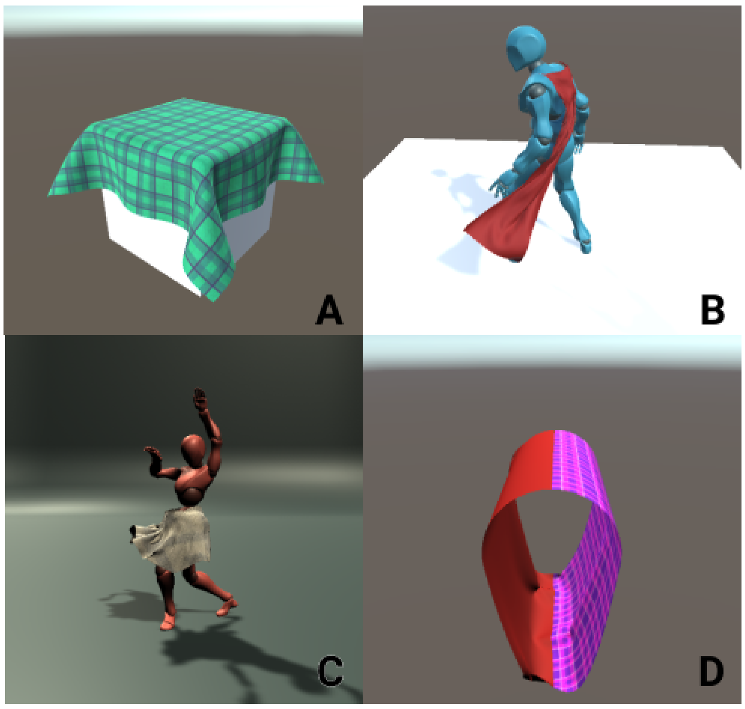

Figure 4.1: Test scenes provided with the Deform Plugin.
A) Shows the simple interaction between a piece of cloth (DeformCloth) and a DeformCollider in the shape of a cube.
B) Shows a DeformCloth attached to vertices of a DeformCollider on the character's back.
C) Shows a character that is dancing with a DeformObject-skirt attached to its hips.
D) Shows two pieces of DeformCloth sewn together.

that needed to be accomplished. At the same time it was naturally accounted for that some phases would run either longer or shorter than planned, and that much of the work of different phases can be done asynchronously.

Each phase consisted of a group of features which all had similar dependencies and functionality. The features were derived from the previous studies of literature and related work. Each phase was planned with the presumption that new knowledge about the technology would be gained from its execution which might result in having to change future plans accordingly, as in accordance with RAD. For the very same reason, features were further elaborated only when their particular phase was reached, as described in FDD. No particular roles were stated for the project.

4.4 Implementation

A major part of this project was the implementation of Vibbi. Vibbi was built incrementally and features were discussed and prioritized on a weekly basis. Features

Phase	Description
1	The simulation of a generic virtual cloth with any shape using Vivace.
2	A 2D user interface in which the user can create pieces of cloth.
3	A 3D view with an avatar to which the pieces of cloth can be added.
4	It shall be possible to do edits whilst the simulation is running.
5	Edits shall be possible directly on the simulated 3D garments.
6	Additional: Scaling or moving avatar or multilayered garments.

Table 4.1: The six different planned phases of the implementation.

were mostly worked upon individually but in some cases in pair. The Unity editor was used to build the user interface and to manage all the resources needed for the project. The ECS was used to create the different models and interactions in the tool. Particular behaviours were scripted using C#, making use of Unity's framework. The Deform Plugin was used to simulate the models as if they were pieces of cloth. The plan was used as a requirement specification, deciding what should be implemented and when it ought to be completed. The whole implementation process was managed using Kanban and the code and resources files were managed with Git as a VCS.

5
Implementation

In this section it is described how features were implemented in the order of which phase they belonged to. To be kept in mind is that while the phases were approached in the order that they were planned, some features from certain phases were done earlier, and some features were done later. The implementation was conducted using Unity, Deform Plugin for Vivace and C#.

5.1 Phase 1: Simulation of a Generic Piece of Cloth

The first step in the process was to create a shape and then simulate it with Vivace. To achieve this there were four features that had to be developed in Vibbi:

1. Define a polygon shape that represents the boundary of the generic piece of cloth to be simulated.

2. Triangulate the shape in order to be able to simulate the generic piece of cloth, as explained in section 2.3: A simulation pipeline of cloth & garments.

3. Create a mesh in order to render the cloth piece in Unity

4. Give the mesh to Vivace in order to initiate the simulation.

5.1.1 Define a polygon shape

In the Unity Editor it is possible to create and edit a polygon shape using the component `PolygonCollider2D` which is provided by Unity. The polygon collider is instantiated as a pentagon and is defined by a list of coordinates in 2D space, which also serves as corners of the shape. With a provided editor mode it is also possible to create more complex polygons by adding and moving the corners of the shape. With this functionality it was possible to try out triangulation and simulation before implementing a way for the users of Vibbi to model their own shapes during run time.

Meanwhile, an editable polygon model for Vibbi, simply known as `EditableModel`, consisting of boundary points and lines had started to take shape; more about that in subsection 5.2.1: Cloth model.

5.1.2 Triangulate a shape

The boundary points of the polygon model also served as boundary points for the triangulation. In the beginning, a naive triangulation algorithm was used but it was eventually replaced with `Triangle`, a constrained Delaunay triangulation algorithm [59], referred to as CDT in the following text. Both algorithms takes a polygon, (list of 2D points), as input and gives vertices and indices as output. `Triangle` also conveniently handles holes in the polygon.

Simulation of cloth looks better the more triangles there are in the mesh. Also, more triangles means more vertices, which in turn means better looking collisions. However, using CDT to finely triangulate a larger polygon shape takes a lot of time, in the order of seconds, see Figure 5.1. Therefore there was an attempt at making a faster triangulation algorithm.

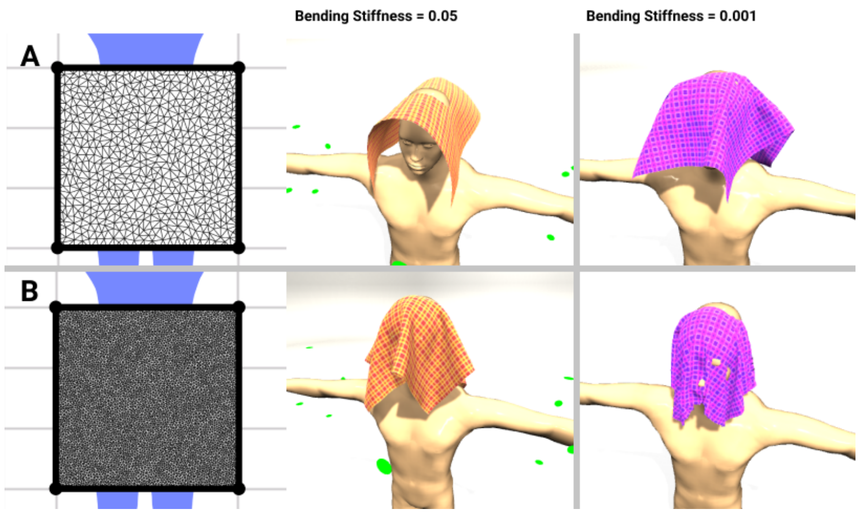

Figure 5.1: Comparison between bending stiffness and different granularity of triangulation of a piece of cloth 3x3 in size.
A) Triangulation granularity 0.01, number of triangles = 1218. Time to triangulate is 0.2 seconds. Subsequent result of simulation on the right.
B) Triangulation granularity 0.001, number of triangles = 11688. Time to triangulate is 17.4 seconds (87 times slower). Subsequent result of simulation on the right.

Both attempted solutions at a faster algorithm involved the function `CreatePatch` supplied by the plugin and reached through `MeshUtils.cs`. CreatePatch can create a rectangular mesh with uniform triangulation, and a high resolution really fast.

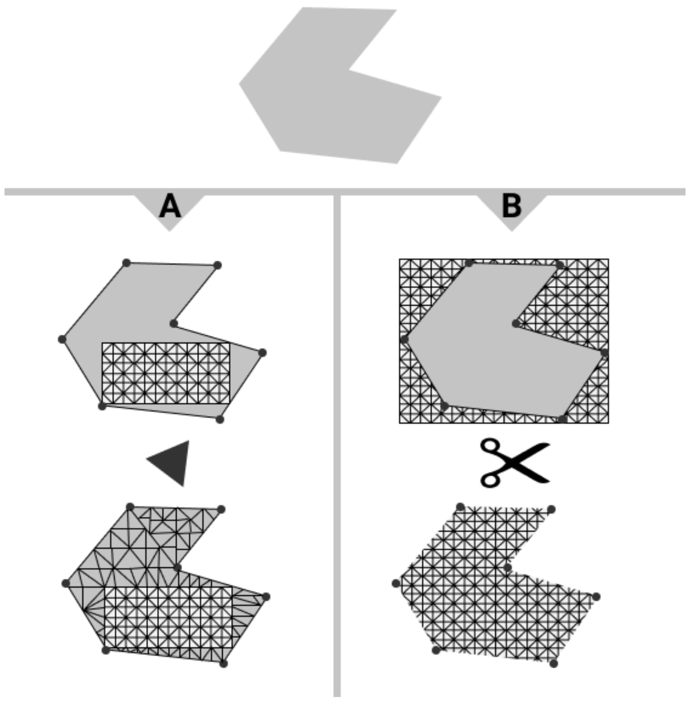

Figure 5.2: Illustrating the two different algorithms using the CreatePatch function to triangulate a polygon.
A) Shows a solution where the first step is to find the largest inner rectangle of the polygon. Then the second step is to triangulate the remaining pieces of the polygon.
B) Shows a solution where the first step is to create a patch as large as the bounding box of the polygon. The second step is then to find all the vertices of the bounding box that are also inside the polygon, virtually 'cutting off' the other vertices.

See Figure 5.2 for a visualization of these two proposed algorithms. The first of these proposed algorithms - algorithm A in the Figure - involved the following steps:

1. Find the largest patch possible in the polygon.

2. Then triangulate the remaining pieces using CDT.

3. Combine into one mesh.

This solution proved difficult to solve due to the fact that the problem of finding the largest inner rectangle of *any* polygon is an algorithm in itself that requires $\mathcal{O}(mn)$ time [60] (where m is the number of rows of vertices and n is the number of columns), i.e., worse than for CDT. Secondly, the foundation of the CDT algorithm would have to be reworked as it had to consider the already existing vertices on the boundary of the patch. These vertices would also have to be defined as being on the boundary at some point. In conclusion, the solution required advanced adjustments which were predicted to result in the same time, or even worse, needed for triangulation compared to CDT.

The second algorithm - algorithm B in Figure 5.2 - was basically the opposite and involved the following steps:

1. Create patch as large as the bounding box of the polygon.
2. Find the subset V of vertices that are in the patch and also inside the polygon.
3. Find the subset T of triangles that involve the vertices of V.
4. Use V and T to create a mesh.

Finding the subset of vertices and then also the corresponding subset of triangles that represent the polygon meant having to go through the list of triangles and comparing vertices with the polygon subset, an operation that takes $\mathcal{O}(n^2)$ time. Thus, this approach also proved to be insufficient.

5.1.3 Create a mesh

A mesh is defined by two things: a list of vertices and a list of triangles that connects these vertices, after triangulation a mesh will have these two things. A `Mesh` in Unity is mainly handled by two components, the `MeshFilter` and the `MeshRenderer`. The MeshFilter stores information about the mesh to be rendered, while the MeshRenderer stores the information concerning the rendering of the mesh, such as lighting and material, as well as being responsible for the rendering during runtime.

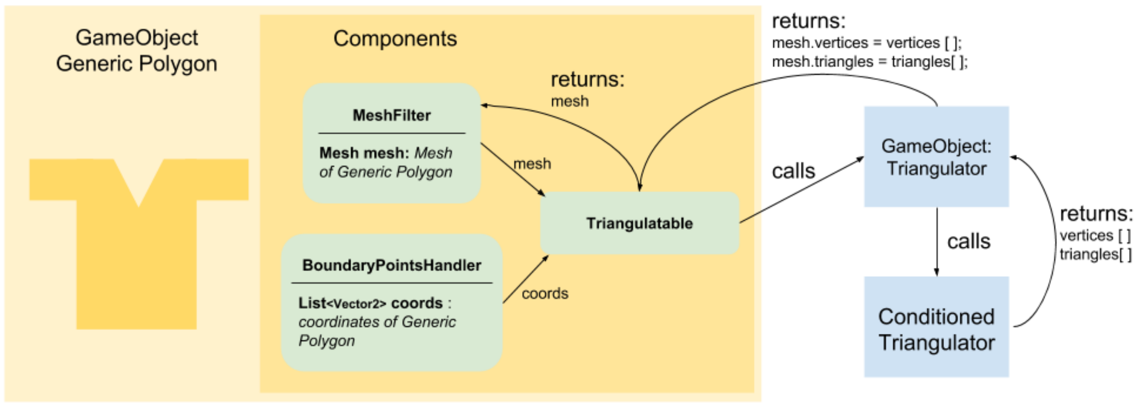

Figure 5.3: The process of creating a mesh. Coordinates representing the polygon shape, and the initially empty mesh is sent to `Triangulatable`, which calls `Triangulator`, which calls `ConditionedTriangulator`. ConditionedTriangulator then sends back a list of vertices and a list of triangles calculated from the coordinates. These lists are put in the mesh which lastly, is sent back to the MeshFilter.

In Vibbi, a polygon that has been triangulated according to CDT will then update

```
[DllImport("deform_plugin")] private static extern int
   CreateDeformableObject(Vector3[] vertices, Vector2[] uvs,
   uint numVertices, int[] indices, uint numIndices, Vector3
   location, Vector4 rotation, Vector3 scale, float
   distanceStiffness, float bendingStiffness);
```

Listing 5.1: CreateDeformableObject: the function used to pass data to Vivace.

the Mesh in MeshFilter with the outputted vertices and triangles, see Figure 5.3.

5.1.4 Give the mesh to Vivace

The Deform Plugin is used to give the mesh to Vivace. A new GameObject is created with the DeformObject component attached. The DeformObject is initialized with the mesh stored in `EditableModel` which in turn initialize three boolean arrays: fixedVertices, attachedVertices and friction. These values determines which vertices should be immobile, attached to a body or affected by friction during the simulation. As the simulation is started the information stored in each initialized DeformObject is passed to Vivace with a function called `CreateDeformableObject`, see Listing 5.1.

5.2 Phase 2: A 2D interface for modeling cloth

The second step in the process was to create a standalone 2D interface which allowed for modeling cloth. This required a few different things, such as:

1. A cloth model in order to make the computer able to aid the designer. A model which the computer can interpret allows for more advanced edits such as unfolding.

2. A user interface which allow the user to utilize the novel technology that Vivace offers and to understand how to use Vibbi, see Figure 5.4

3. Operations for changing the cloth model to allow the user to create different types of patterns.

4. A seam model that defines for Vibbi what a seam is, in order to let the user define seams.

5. Operations for sewing cloth models together so that the user can define *how* the patterns form a garment.

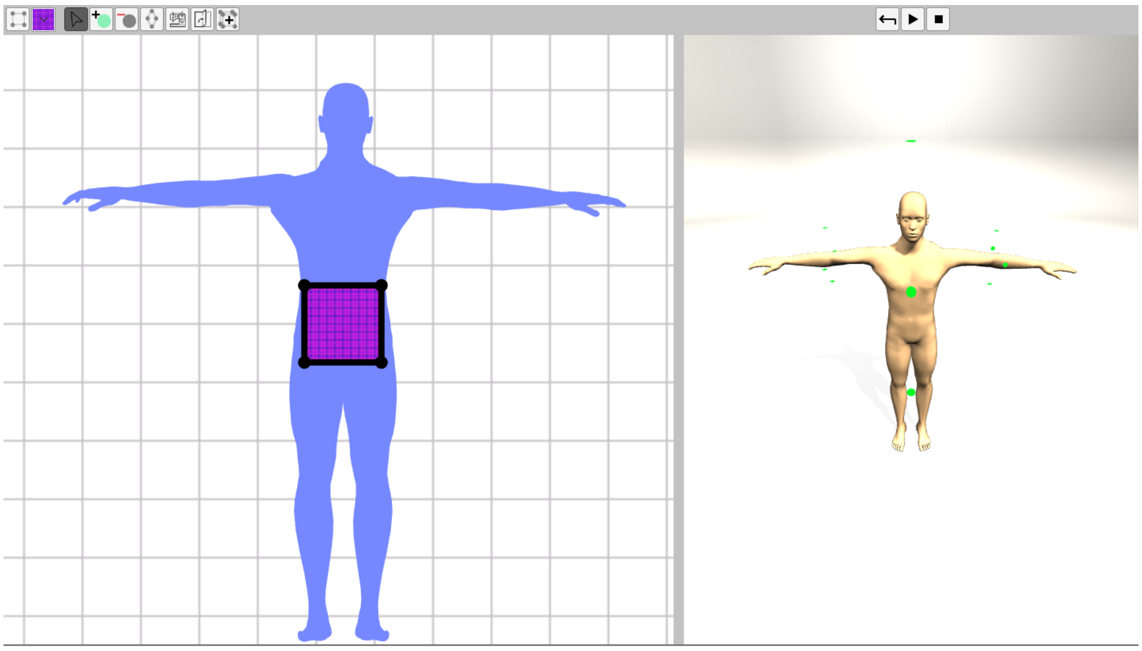

Figure 5.4: Vibbi's user interface with a 2D modeling view to the left and a 3D simulation view to the right.

5.2.1 Cloth model

Vibbi models cloth as a polygon shape. A polygon shape is defined by its boundary and hence a model for creating and editing a closed boundary was made by designing a few different GameObjects in Unity. The idea was to have *boundary points* which the user can move freely in two dimensions and *boundary lines* which visually connects the points. In Vibbi, EditableModel is the GameObject which represents the cloth model, and it consists of a list of `BoundaryLines` and `BoundaryPoints`, see Figure 5.5.

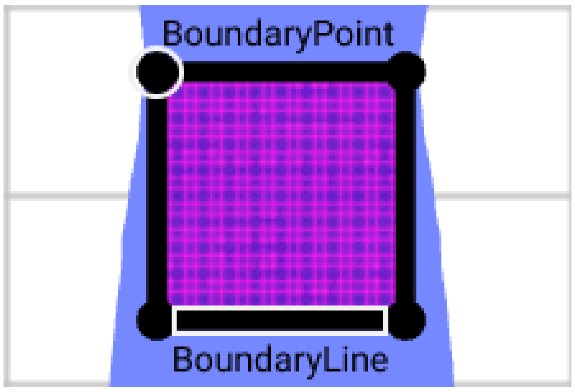

Figure 5.5: The EditableModel in Vibbi, highlighted is a BoundaryPoint and a BoundaryLine.

When adding a cloth model to Vibbi, it always start out as a simple square consisting of four points and four lines. By adding additional points to the existing lines, and then moving them, the user is able to create more complex polygons.

5.2.2 User interface

Vibbi required a space in which the modeling could take place, a typical canvas. Unity's documentation describes a UI system which allows for creating user interfaces fast and intuitively [61]. This UI system came in handy when creating the UI for Vibbi.

Vibbi's UI can be divided into three parts: screen overlay, model view and simulation view, see Figure 5.4. The screen overlay is the UI which will always stay the same, regardless of screen size, and which contains the toolbar. The toolbar, which is illustrated in Figure 5.6, allows the user to: add new cloth models to the screen, change material of a cloth model, and to switch between the different *interaction states*. Initially, the tools in Vibbi could only be accessed through keyboard shortcuts which are still left in the software allowing for faster interaction.

The model view is the UI in which the modeling of patterns is conducted. Unity provides different tools for this type of view and interaction. With Orthographic Projection it is possible to look at a 3D world without perspective so that it appears 2D, and together with ray tracing as well as other mouse to world position translation techniques it is possible to track what the user is doing with the mouse.

Simulation view is where the garments are displayed and simulated. In this view a camera with Perspective Projection is used. The two cameras are initiated to take up 3/5 and 2/5 of the width of the screen respectively, which creates the division in the UI.

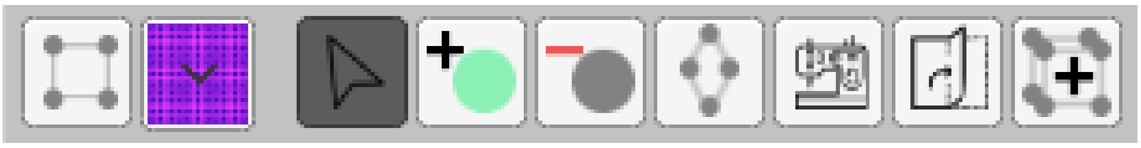

Figure 5.6: Vibbi's toolbar from the left: Add cloth model, Change material, Select, Add points, Remove points, Add darts, Add seams, Unfold and Duplicate.

An avatar silhouette was added in the 2D view which later was size correlated with the avatar in the 3D view.

5.2.3 Interaction states

Vibbi uses interaction states to let the user do different actions with the same kind of interaction. E.g., the interaction state determines what should happen when a user performs certain interactions, such as clicking on a BoundaryLine. The user has seven different tools available to them: Select, Add Point, Remove Point, Dart, Sew, Unfold, and Duplicate, see Figure 5.6. These tools are toggled so that only one of them can be active at a time, thus moving between the different interaction states. For example, the difference between being in the interaction state 'Select' and 'Add Point' is that when the user clicks on a line during the former, the line will be selected, and during the latter, a point will be added on the line.

5.2.4 Move a point

The BoundaryPoints of an EditableModel make use of Unity's predefined `Sphere`. Sphere is a GameObject which simply has a sphere mesh in its MeshFilter component. By also adding a `SphereCollider` it is possible to track if the mouse pointer is currently pointing at the object and if it is being dragged or not. By dragging a point the user can move it with two degrees of freedom, see Figure 5.7.

This is implemented by adding yet another Component to the points, called `Movable`. Movable is a script which makes use of the above mentioned functionality and translates how the mouse is moving over the screen into how the point should move through the world. The same Component can be used to move another GameObject similarly as long as it has a Collider, e.g., the whole EditableModel itself.

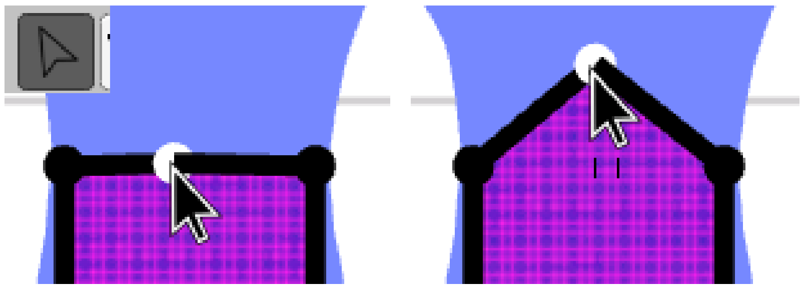

Figure 5.7: Illustrating how the user moves a boundary point.

5.2.5 Move a line

The BoundaryLines of an EditableModel is composed similarly to the BoundaryPoints. A big difference is that the lines can change their size depending on where the points are moved. This behaviour is scripted in a Component called `SimpleLineBehaviour`. SimpleLineBehaviour holds a reference to two points and by utilizing Unity's game loop framework, the line can continuously track the points and update its position and orientation according to theirs.

By dragging a line the user can move it with one degree of freedom, see Figure 5.8. This is implemented with a Component named `MovableLine` which translates the distance the mouse has moved over the screen into how far the line should be moved in the direction of its normal vector. This is done by actually moving the two points that the line is continuously tracking the positions of.

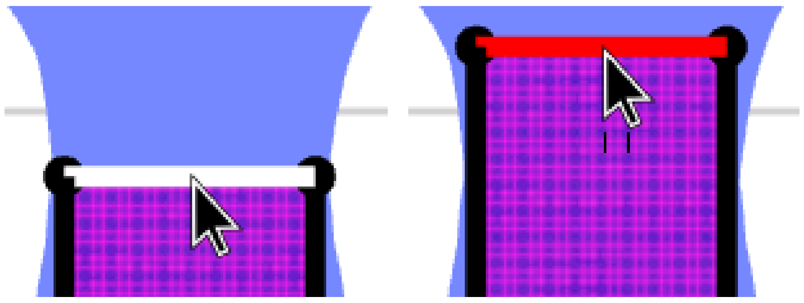
Figure 5.8: Illustrating how the user moves a boundary line.

5.2.6 Add & remove a point

The EditableModel has a component called `BoundaryPointsHandler` which keeps track of the model's boundaries. To add a new point the user simply has to click somewhere on one of the boundary lines, see Figure 5.9. When a point is added, the existing line is shortened and a new line is created so that the model always stays connected.

Similarly, it is possible to remove a point by clicking on it while in the correct interaction state (i.e., 'Remove Point'), see Figure 5.10. This action will also remove the line which were connected to the point being removed and connect the two neighbouring points with the line that is left.

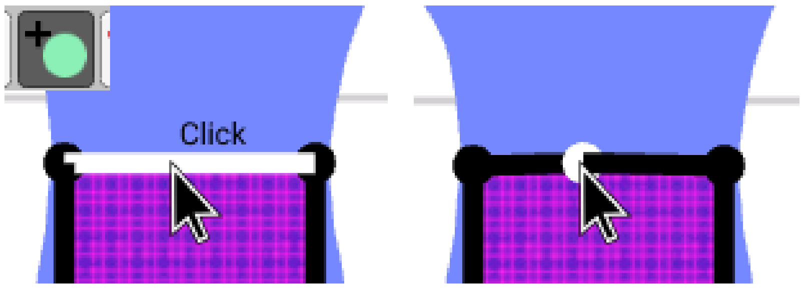

Figure 5.9: Illustrating how the user adds a new boundary point to the cloth model.

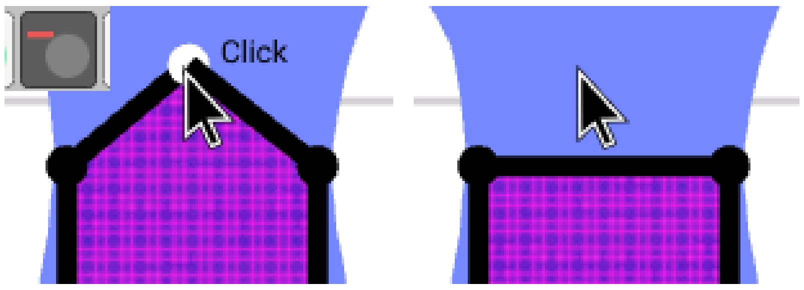

Figure 5.10: Illustrating how the user removes a boundary point from the cloth model.

5.2.7 Unfolding the model

Unfolding is done by choosing a line that the whole model should be unfolded through. Each point, except the ones connected to the line, is copied and their positions are mirrored through the line, see Figure 5.11. To keep the same connectivity as well as the polygon shape, lines are added and updated to be part of the new boundary. This is all scripted in BoundaryPointsHandler.

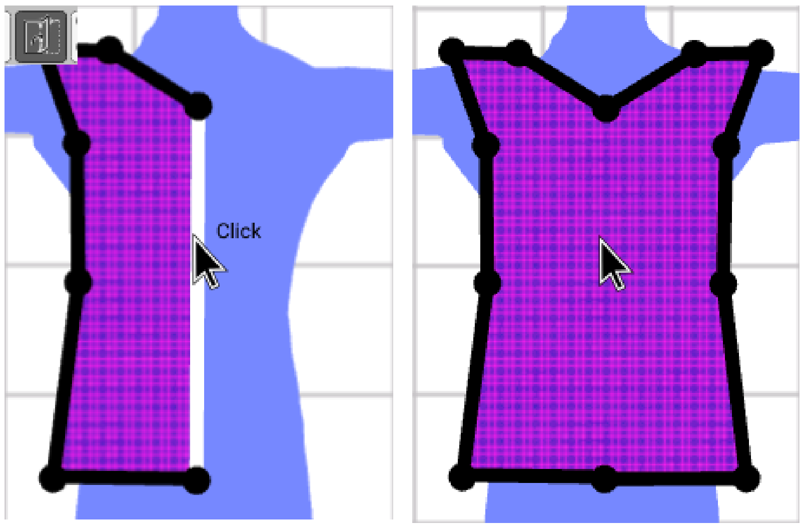

Figure 5.11: Illustrating how the user unfolds a cloth model.

5.2.8 Darts

Darts are created by dragging the mouse across an editable model while using the 'dart tool', see Figure 5.12. The initial shape of the dart is a diamond consisting of four boundary points and lines. The dart serves as a hole in the mesh which is created by giving the boundary positions of the dart to the triangulation. The seam of the dart has not been implemented, but can be added manually with the 'sewing tool'.

5.2.9 Duplicating the model

It is possible to duplicate an existing EditableModel. This is mainly done with Unity's built-in functionality for copying GameObjects. By adding the boundary objects as children to the EditableModel certain functionality was gained inherently. Among other things, child objects tracks the parent and stays relatively positioned in the case the parent is moved. Also if the parent object is copied, all children are copied as well. Vibbi used this to its advantage but for it to work properly it was necessary to reinitialize referenced variables in the new BoundaryPointsHandler so that there would not be a connection between the new model and the old boundaries.

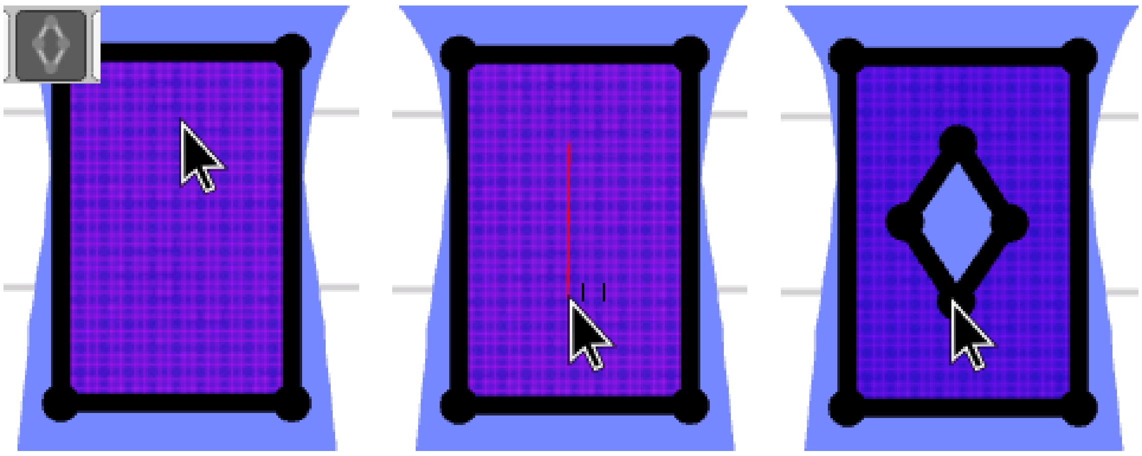

Figure 5.12: Illustrating how the user creates a dart on top of a cloth model.

5.2.10 Undo & redo

A undo and redo framework was added to Vibbi, and some of the initially implemented operations such as adding, removing and moving a point was added to it. To be able to undo and redo operations, inverse operations were implemented and Vibbi had to track the history of operations.

5.2.11 Seams & sewing

The user can select two boundary lines, either from the same cloth model or two different ones, to sew them together, see Figure 5.13. A seam is hence defined by two lines and for convenience also the EditableModels that the lines belong to are involved.

To avoid the seam stitches from crossing it was important that the user could define where the seam starts and ends on both of the lines. To solve this, Vibbi tracks where on the BoundaryLines the user clicks as the sewing is being conducted and uses the closest edge as the start position. This is visualized with small notches which is placed closer to the start of both lines. A seam is further visualized by coloring the lines with matching color as well as connecting the start and end positions of the lines with two smaller lines.

5.3 Phase 3: A 3D environment for garments

The third phase in the implementation was to create a 3D environment in which the patterns and seams could be modeled as garments. To achieve this there were five features that had to be implemented:

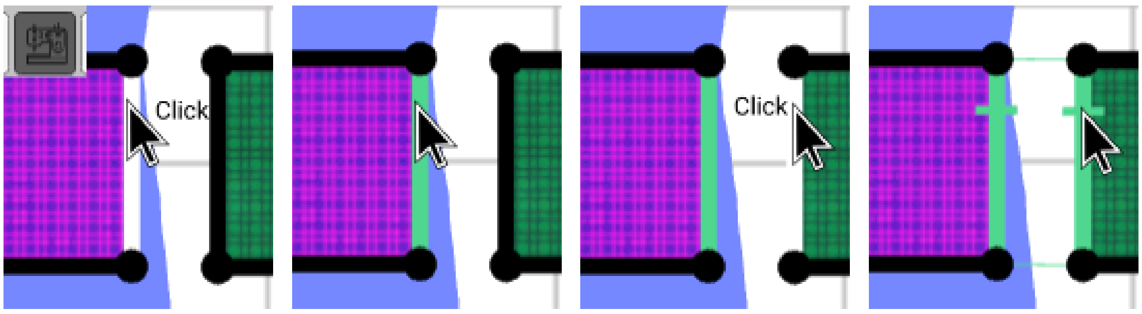

Figure 5.13: Illustrating how the user can sew two cloth models together.

1. Loading the cloth model to 3D in order to put the garment in a space where it can eventually be simulated.

2. A 3D avatar to hang the garment upon, in order to allow the user to gain an understanding for how the garment looks on a body.

3. Operations for dressing the avatar so that the user can freely choose how the avatar should be dressed in their designed garment.

4. A 3D seam model to bring the information from the 2D seam into the 3D environment.

5. Make the simulation work between Vibbi and the Deform Plugin.

5.3.1 Loading the cloth model

When loading an EditableModel to the 3D view, the EditableModel works as a template for creating a copy in the 3D window. This copy is a new GameObject called `ClothPiece`. The ID, mesh, and material of EditableModel is copied over to the ClothPiece. The shared ID allows for bidirectional editing, so that edits made on the EditableModel will immediately be seen on the ClothPiece as well.

When the ClothPiece has been created, the EditableModel that is currently being loaded will also try to load any seams that are connected to it. Thus, as long as both ClothPieces that have the same ID as the relevant EditableModels are loaded to the 3D view, the seam will be loaded as well.

5.3.2 Avatar

There are plenty of meshes that can be found online which models differently shaped human characters. First, Vibbi used a robot-like avatar from Mixamo [62]. The reason for this was mainly how simple it was to import it to Unity but also the potential to later be able to animate the avatar with clothes on, something which is possible with Vivace but not yet implemented in Vibbi.

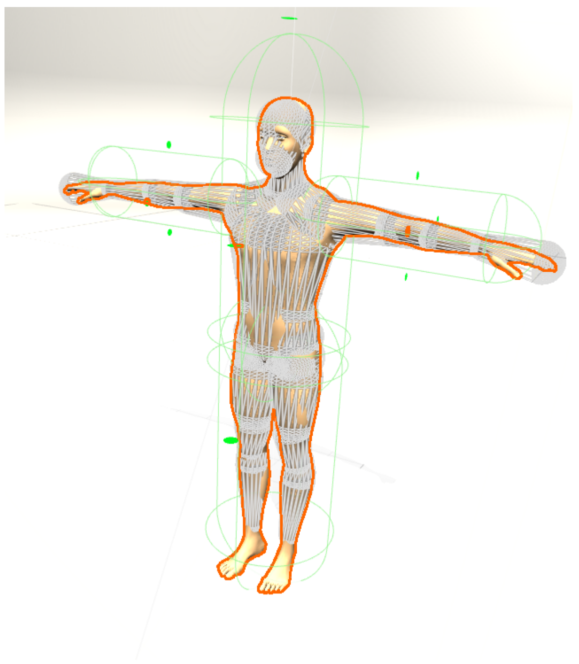

Figure 5.14: Illustrating how the avatar is surrounded by two groups of cylinders: white for collision detection and green for adding attachment points.

At the moment Vibbi uses a human-like avatar. This was done mainly to emphasize that the tool can be used for creating patterns for real garments and not just virtual ones.

Vivace can, as of yet, not handle collision with a complex mesh such as a human body in real time. Hence, a set of cleverly placed cylindrical `CapsuleColliders` are used to model the surface of the avatar, see Figure 5.14.

5.3.3 Placement of cloth pieces

Vibbi uses a set of *attachment points* to let the user place cloth pieces on top of the body of the avatar in the 3D environment. In addition to a few standard, previously placed attachment points it is possible for the user to add more to the environment by clicking on the avatar. Currently the attachment points align themselves to a set of conveniently placed CapsuleColliders and not to the avatar itself, see Figure 5.14.

To place and load a cloth piece to the environment the user simply selects the cloth and clicks on a desired attachment point. The cloth piece and in particular its mesh will align to the attachment point's position and orientation, which is automatically directed away from the colliders. This way the cloth pieces are always front faced regardless of which side of the avatar it is placed, see Figure 5.15. The orientation

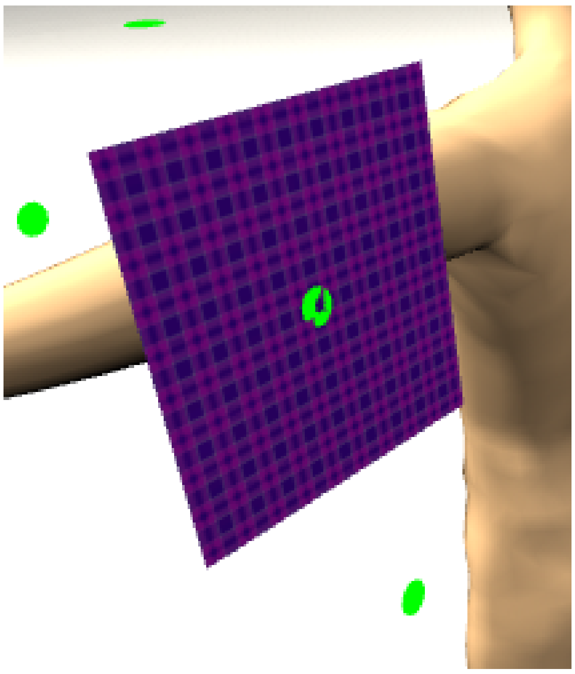

Figure 5.15: Illustrating how a cloth piece is oriented according to a selected attachment point which is facing away from the avatar.

matters, especially to avoid seam stitches from crossing. If a cloth piece is rotated differently from how the user thinks, the seam is likely to be rotated differently as well.

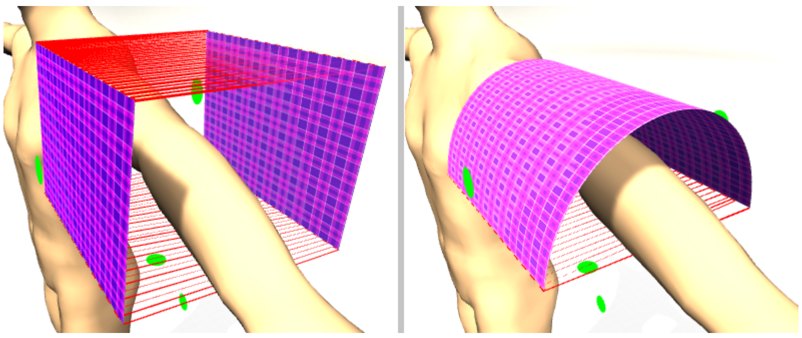

Figure 5.16: Illustrating two possible ways of preparing a sleeve for simulation using either two pieces or one piece of cloth.

The flat pieces of cloth and the way seams constrain these together during simulation introduces an issue: it is not possible to make a one piece sleeve. In reality, a sleeve usually only requires one pattern piece, which sewn together with itself creates a cylindrical shape. Hence, a bending algorithm was created to circumvent this issue. Loading a piece of cloth to an attachment point positioned on top of one of the arms will automatically bend the piece 180 degrees around the direction axis of the arm, see Figure 5.16.

5.3.4 Camera orientation

The 3D environment uses a Camera which the user can move around due to a component called `Orbit`. All the movement is done by manipulating the camera's position and rotation. It is possible to pan the camera both horizontally and vertically. It is also possible to move the camera forward and backwards which creates a zooming effect. Furthermore, it is possible to rotate the camera around a fixed point which is a certain distance in front of the camera. The fixed point is set to be the avatar's position in the 3D environment, which make it seem as if the user is orbiting the avatar as the mouse is dragged around the screen.

5.3.5 Seams

Similarly to how an EditableModel is converted into a ClothPiece, a seam in 2D is converted into a seam in 3D, which is a GameObject called `GarmentSeam`. A GarmentSeam is defined by the ClothPieces it belongs to as well as a list of vertices. This list of vertices contains each pair of vertices which are to be constrained during the simulation, see Figure 5.17.

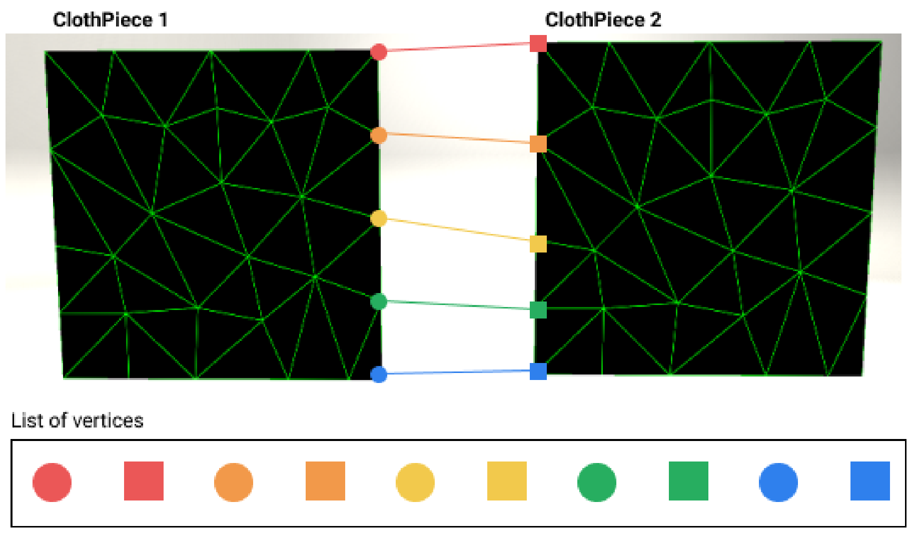

Figure 5.17: Representation of a typical GarmentSeam. Illustrating how the vertices are ordered in pairs in one list that is later sent to Vivace.

In order to define a seam one first has to find every vertex present on the BoundaryLines that are to be sewn. This was done by first defining a linear function between the two BoundaryPoints on the edges of a BoundaryLine, and then going through the vertices in the mesh to find which vertices fit in the function. When all the vertices of both lines have been found they can be paired up.

Since two BoundaryLines are not required to be of equal length, and more importantly to not contain equally many vertices, a strategy was used in order to achieve evenly distributed vertex pairs. The vertices present on the start and end positions of the BoundaryLine are always paired up first, i.e., start with start and end with end (therefore, in order to not get crossed seams it is important to correctly define where the start position is). Next, the remaining vertices are paired up using a divide and conquer algorithm, see Figure 5.18.

The divide and conquer algorithm initially takes the two separate lists of line-vertices and connects the vertices in the middle of the lists. Then it divides each list into two new lists, e.g., four lists in total, connects the middle vertices of these lists, and so on. This continues until there are only two elements or less left in one of the lists.

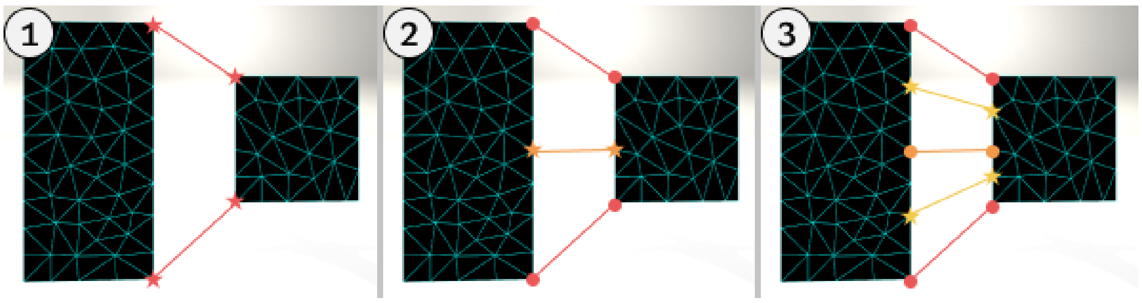

Figure 5.18: How the vertices of a seam are paired up. 1) Start and end vertices are paired. 2) Divide and conquer algorithm connects the middle vertex in each list. 3) Only one element left each in the lists on the right. They get paired up with the middle vertices on the right.

Vibbi visualizes loaded seams as a bunch of lines between two loaded ClothPieces, see Figure 5.19. This is done with the `LineRenderer` component, which draws a line between a list of positions. These lines are drawn between the actual vertex positions that are going to be constrained during the simulation.

5.3.6 Simulation

The simulation view offers a toolbar to the user with the options to Unload all ClothPieces, Start simulation, Stop simulation, and Fix the camera, see Figure 5.20.

Starting the simulation sets the following chain of events in motion. First, the DeformObject component is attached to all of the ClothPieces in the 3D window. Next, the DeformManager is reset, meaning it will shutdown the Vivace engine, and then find all of the DeformBodies - i.e., all the ClothPieces with the DeformObject component attached. All the DeformBodies are sent to Vivace, and then all the lists of seam vertices in Vibbi are fetched and sent to Vivace as well. After that the DeformManager tells Vivace to start the simulation.

Stopping the simulation does not involve shutting down Vivace because that will make the engine crash. Instead, DeformManager simply stops requesting updates

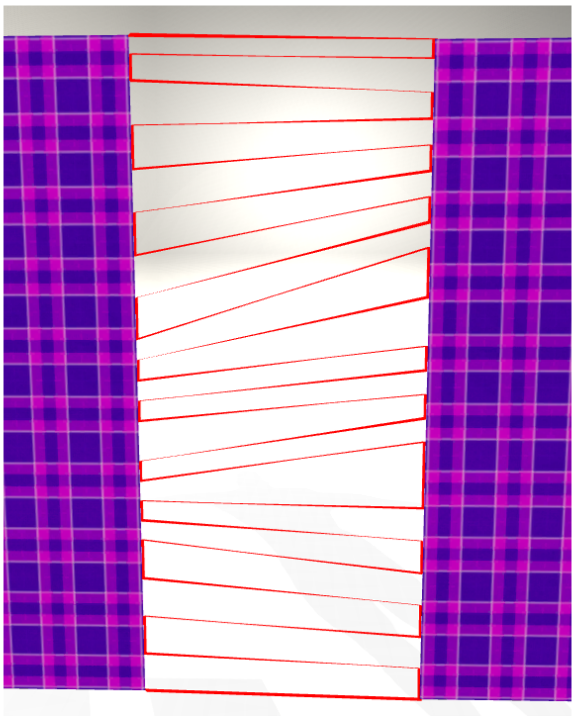

Figure 5.19: What a seam looks like in Vibbi (3D window).

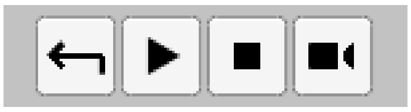

Figure 5.20: The simulation view toolbar. From left to right: Unload all ClothPieces, Start simulation, Stop simulation, and Fix camera.

from Vivace. After that all of the original positions and rotations of every ClothPiece are saved before all of the ClothPieces and GarmentSeams are unloaded (i.e., deleted) from the 3D window. Next, new copies of the previously deleted ClothPieces are created using their original positions and rotations. The new GarmentSeams are automatically loaded together with the ClothPieces.

Fixing the camera simply positions the camera at a user defined position and rotation. Allowing for viewing the designed garments in a consistent manner.

5.3.7 Interaction

It is possible to pinch the cloth and drag it, in order to move it around. This interaction is done by right-clicking on a piece of fabric, the vertex that is hit with the mouse will then follow it around, see Figure 5.21. This interaction is handled by the DeformManager and Vivace.

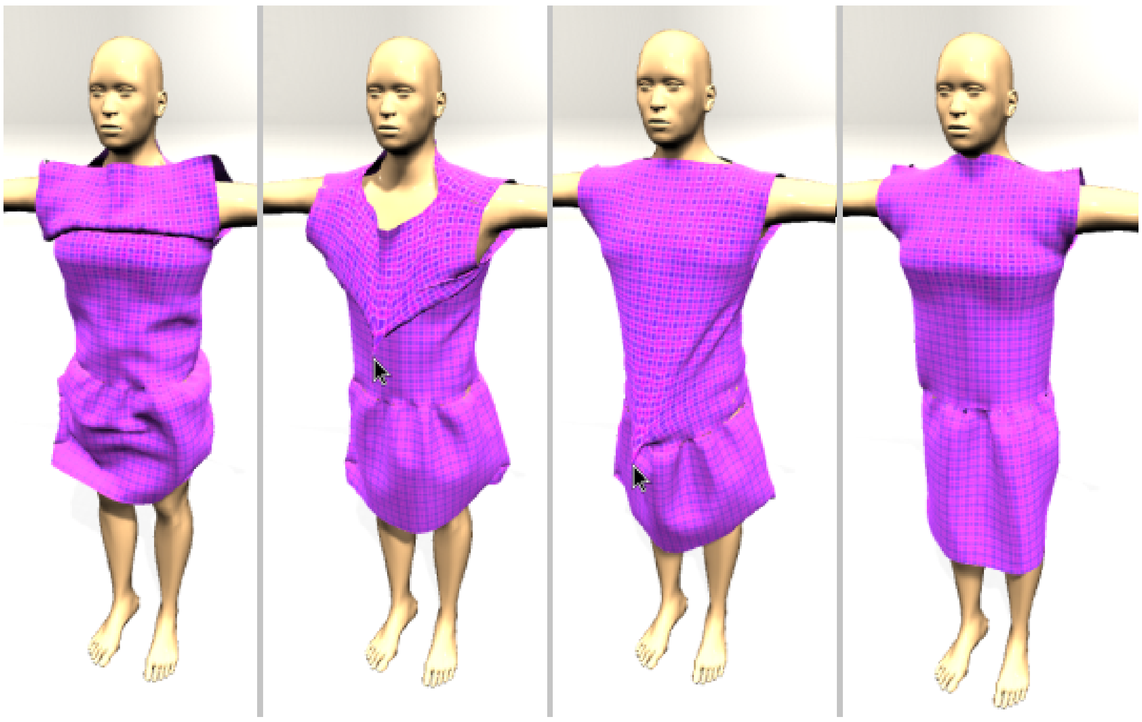
Figure 5.21: The simulated dress is undesirably positioned due to friction and stiffness. The user can tug the garment to position the garment differently.

5.4 Phase 4: Simulation while modeling

The fourth phase of the project was to implement functionality for simulation while modeling. To do this Vibbi had to gain access to each timestep of the simulation loop of Vivace, in order to be able to update the mesh. While this functionality partly exists, as one can interact with the simulated garments (see Section 5.3.7), the full functionality needed to update a mesh did not.

However, a cloth model that has been copied and loaded to the 3D environment can still be edited. Changes made to the mesh and material will happen in both 2D and 3D but not during simulation. This is because EditableModel and ClothPiece shares the same instance of Mesh. It is possible to iterate the design by stopping the simulation, making a few changes, and starting it once more. Seams are updated each time a triangulation is made, hence the loaded seams are also updated when changes are made to the model.

Some strategies were discussed, but not implemented, to be able change the mesh while the simulation was running. It was decided that changing the length of cloth should not trigger a complete triangulation of the new shape - since that takes a lot of time - but rather stretch the mesh by moving the vertices along the boundary being moved, see Figure 5.22. This stretch can potentially be progressively refined for better results.

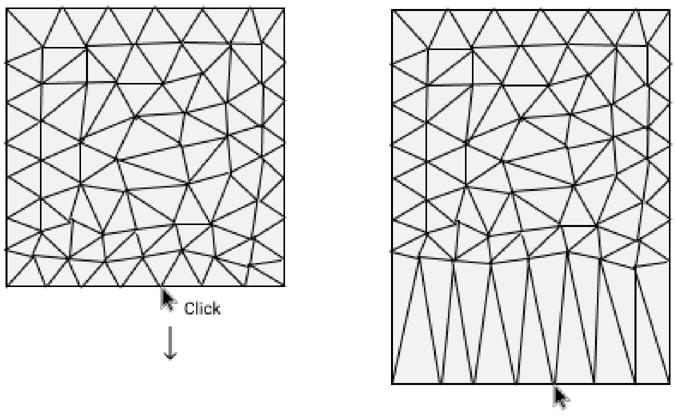

Figure 5.22: Illustrating how a mesh can be stretched by simply moving the positions of the boundary line vertices.

6

Results

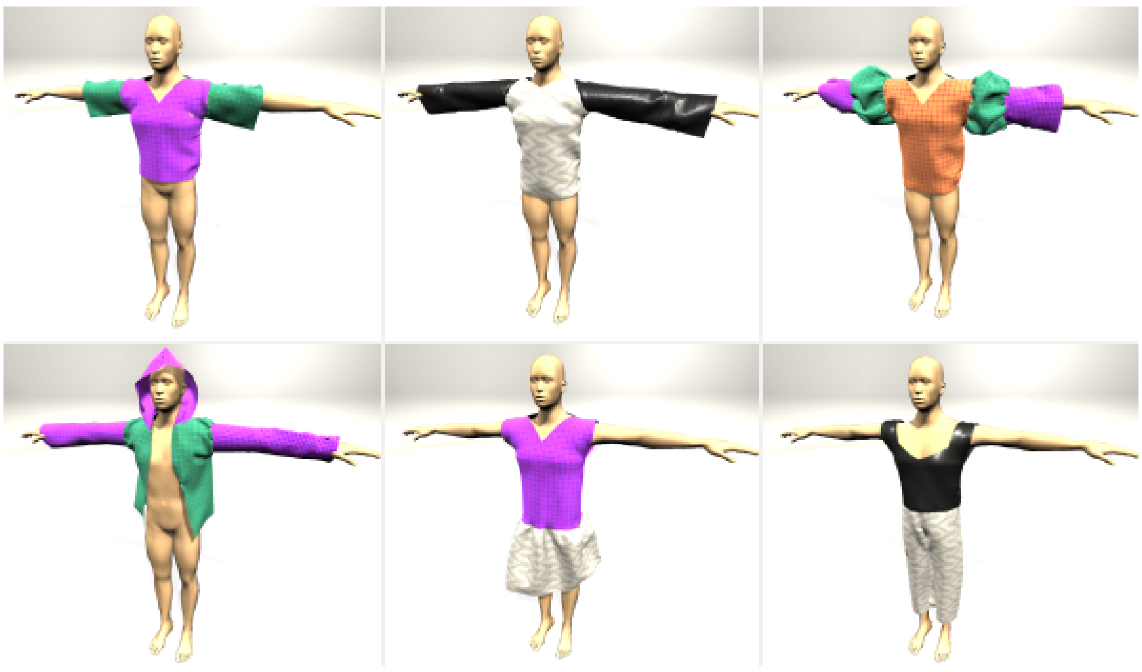

Figure 6.1: Six different garments designed by using Vibbi. In the figure the garments are simulated in the 3D view of Vibbi.

Figure 6.1 illustrates how one can, with seamless effort, design different types of virtual garments using Vibbi. In steps, the user can model differently shaped patterns, choose how these patterns should be sewn together, visualize the patterns together in a 3D environment, and finally simulate the patterns as sewn garments.

Vibbi enables the user to see the result in 3D even though the modeling is being done in 2D. The garments are also simulated using physical parameters such as gravity, friction, and stiffness which allow for a more real world accurate representation of the result. Taking this complexity into consideration, the simulation computes in the order of milliseconds which allows for immediate interaction.

6.1 The Vibbi design process

In order to design a garment in Vibbi one typically goes through five steps, and these are:

1. Create pattern pieces.
2. Sew pieces together.
3. Put the pieces in the 3D view.
4. Simulate.
5. Make edits as you wish.

In order to gain a full understanding for this process, each step will be thoroughly explained.

6.1.1 Create pattern pieces

Every piece of cloth that is to be part of the garment needs to be created in the model view of Vibbi, see Figure 6.2. A new polygon is created either by pressing the square polygon icon in the toolbar (furthest to the left) or by pressing 'C' on the keyboard.

When a new polygon has been created they are edited into the desired shape by adding points and moving points and lines around.

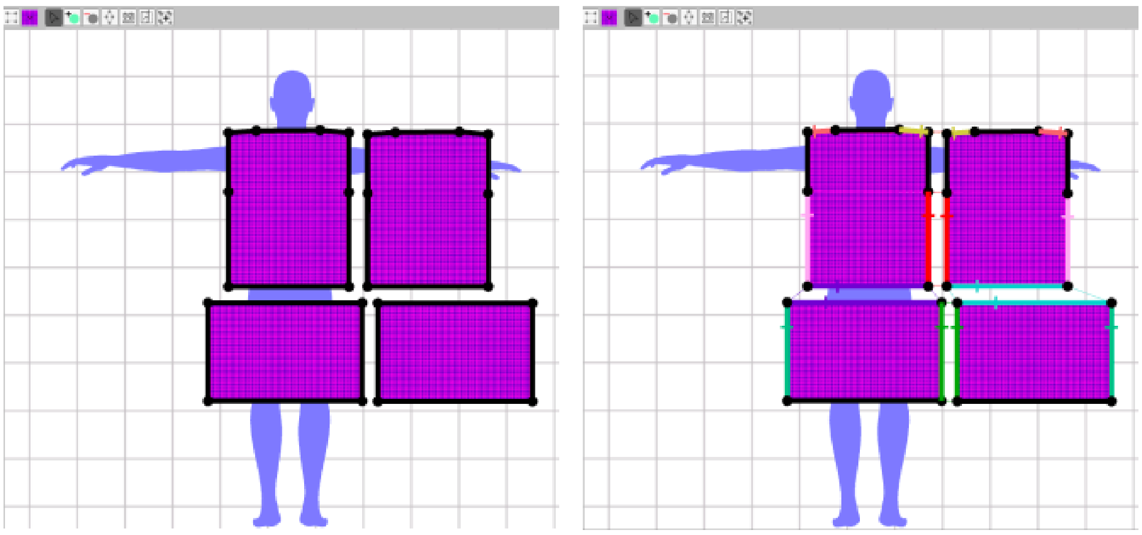

Figure 6.2: On the left, four different pattern pieces have been created in the model view of Vibbi. On the right, seams have been added to make it into a dress.

6.1.2 Sew pieces together

When the desired pattern pieces have been created the user can sew the pattern pieces together, see Figure 6.2. To take into consideration is the future orientation of the pieces in the 3D view, as well as how the start points of the seam will affect the outcome. Seams can be done after the pieces have been loaded into the 3D view as well.

6.1.3 Put the pieces in the 3D view

After the previous steps the pieces are put in the 3D view, see Figure 6.3. As mentioned before, the sewing can take place either before or after this step. As it is now possible to inspect the seams to see if anything is amiss, it is possible to redo a seam if it is not as desired.

The user adds their pieces to the 3D window by selecting an attachment point and a piece of cloth (in any order), and then pressing 'L'. If the user does not chose an attachment point the piece of cloth will simply be loaded above the character instead. After the piece has been loaded to the 3D view the user can move the piece around with two degrees of freedom in order to ensure that the avatar, seams and cloth pieces are not crossing each other.

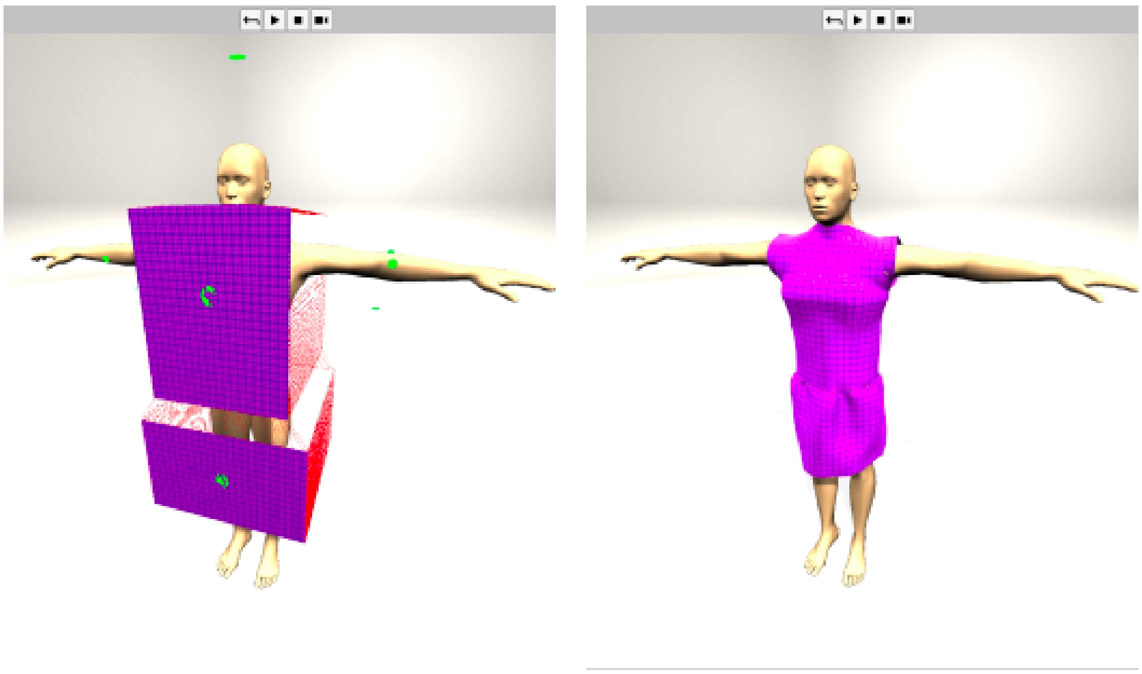

Figure 6.3: On the left, the four cloth pieces have been loaded to the 3D view where the seams are visualized. To the right, the simulation is running with the same pieces.

6.1.4 Simulate

When the user is satisfied with the orientation of the pattern pieces and the seams in the 3D view it is time to simulate, see Figure 6.3. The simulation is started either by pressing the 'Play' button above the simulation view or by pressing the space bar. During simulation the user can right click and drag to move the cloth around, so it is possible to adjust the garment on the character.

6.1.5 Make edits

Since the simulation of the garment provides new insights for a more desired look, the user can retroactively make edits to the pattern pieces, see Figure 6.4. While the simulation is running it is possible to change the material of the fabric. However, to change the mesh of the cloth pieces it is required that the user first stops the simulation. This does not require pieces of cloth to be reloaded to the simulation view, it is handled automatically by Vibbi.

6.2 What should be considered when making an interactive modeling tool for designing virtual clothes?

A CAD tool for designing virtual garments can be done in several different ways depending on the target user and where in the design process of clothes one feels a need to simplify it. In chapter 2: Technical Background several different approaches are presented such as Tangible Modeling, Sketch-based Modeling, and Interactive Modeling - as well as different CAD tools already present on the market. These different approaches all come with various types of problems that needs to be solved. However, only considering an interactive modeling tool the problems become more distinct.

More specifically, by creating Vibbi - an interactive garment modeling tool - the following points had to be considered:

- What external tools to use
- How to work
- How to model patterns
- How to sew patterns together

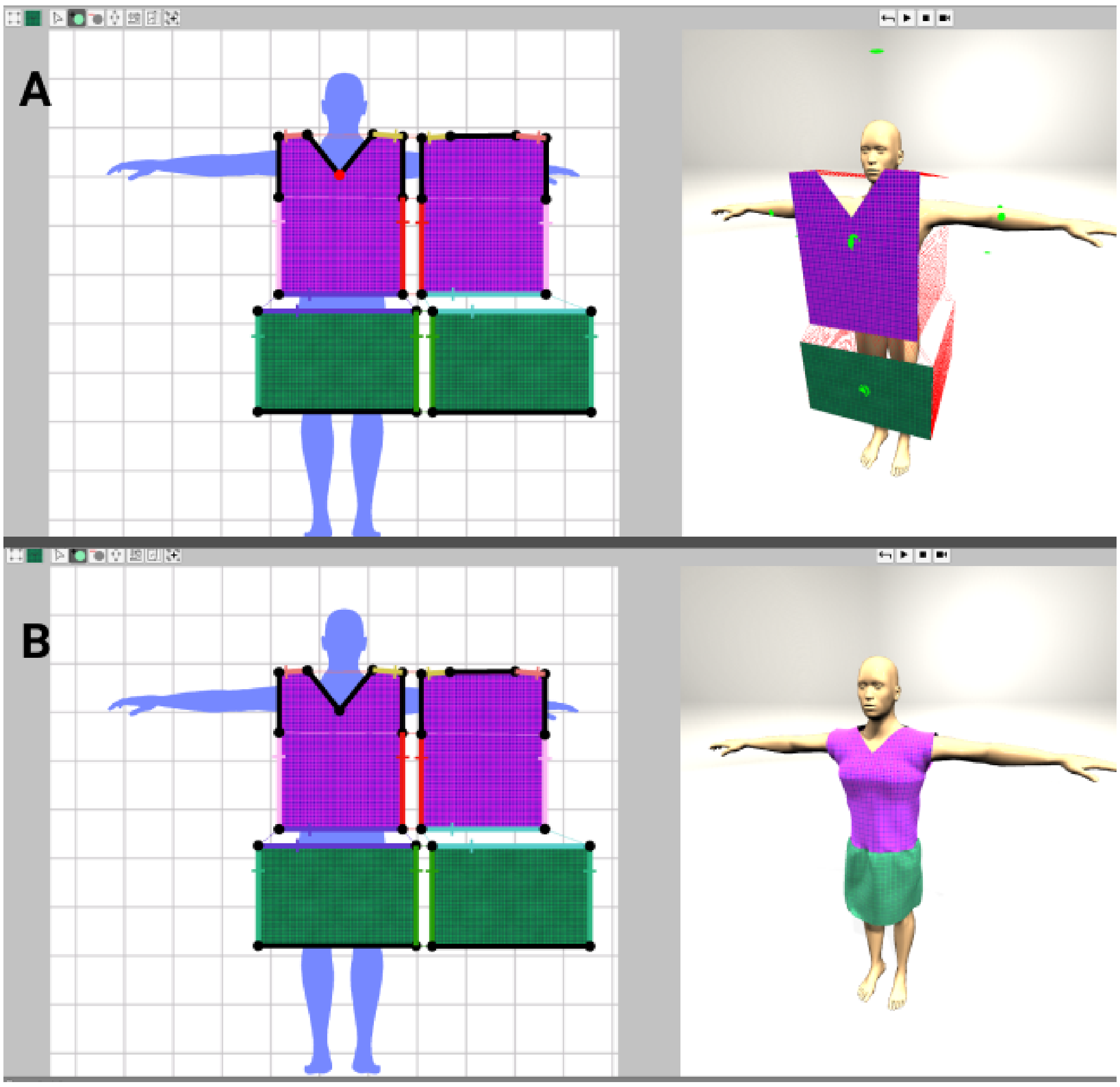

Figure 6.4: Making edits to the previous dress.
In A) the simulation is stopped and thus the new mesh of the front piece can be loaded to the simulation view.
B) Shows the edited dress while it is being simulated.

- How to dress the avatar
- How to simulate the clothes

6.2.1 What external tools to use

The first point to consider when creating an interactive garment modeling tool is what other external tools to use. This depends heavily on two things, namely: how much time there is for development and where the focus of the work is. There is always a need for a physics engine capable of simulating cloth at interactive rates, which is a rare commodity.

If, as with Vibbi, time is short and focus lies on exploring new interactive techniques such as simulation while modeling and bidirectional editing one should consider using a third party software or RDL to aid the implementation. For example, there are tons of different free to use game development libraries and platforms, like Unity, available [63]. These tools provide a lot of things for free, such as collision handling and rendering - there is no need to reinvent the wheel.

6.2.2 How to work

Using ASD is good for basically any type of software development project [49]. Working feature by feature, rapidly creating prototypes to try out the software is a good way to ensure that something of value is being created. How to work when considering an ASD method is very dependent on the size of the team and length of the project.

6.2.3 How to model patterns

Modeling patterns can be done in various different ways, for example freehand drawing or shape-editing. Depending on the target users, to be considered is what types of interactions are necessary for them. Also to be considered is if the users should be able to create actual, real clothes or if the software should be used as a way to quickly prototype new designs. As making real garments typically requires a high level of accuracy for every measurement and to ensure symmetry.

6.2.4 How to sew patterns together

For the patterns to become a garment they need to be sewn together. What one needs to consider foremost is the actual interaction to define the seam, to somehow let the user select two different lines, either lines defined as edges of a pattern or lines defined in the middle of a pattern.

The next thing to consider is how to visually state to the user both how, and that the seam is defined. With multiple seams present, how the user can differentiate between them, and at the same time know which lines are sewn to which. Color is a convenient way of doing this, but has the drawback of not being applicable for colorblind users. Using color there is also a need to put work into building up a 'color bank' of strong colors that are not too similar to each other.

The third thing to consider is how the user selects the starting point of the seam on each line, as well as how this is visually represented.

6.2.5 How to dress the avatar

One needs to consider how the patterns goes from the 2D modeling view into a state where they are placed as a garment on an avatar. Dressing the avatar on the computer comes with both possibilities as well as drawbacks. There are opportunities here to be explored such as changing the size of the avatar to see how the clothes will fit differently. Most drawbacks comes from the fact that the user is doing 3D interactions on a 2D screen, thus losing their sense of depth.

One can provide the user with the possibility of moving each pattern piece freely in the 3D world, in order to provide maximum freedom. While considering that this can provide a lot of problems for them as well. One can also assume that since the patterns are to become garments, they should be placed on the avatar, and thus provide a way for the user to place their garments on top of it, e.g., the attachment points found in Vibbi. In reality when creating patterns one defines each piece as to where they belong, for example ’Front’ or ’Back’. This requires either some sort of classification system which can become confusing and restricting for the user - especially if creating a program that can handle multiple layers of cloth - or some sort of natural language interpreter which is technically complex to make it work flawlessly.

6.2.6 How to simulate the clothes

Representing the actual result digitally can be done in different ways, e.g., with 3D garment models or simulation. In reality, the result can be tried out on a person who can effortlessly walk around and give visual and subjective feedback. Interactive simulation is required in order to achieve equal amount of feedback digitally. Simulating clothes in a realistic manner is a complex task which constrains what models can be used for both patterns, seams, and garments. Inherently it can also limit the amount of interactions possible for the designer. E.g., the models decides whether or not bidirectional editing is possible. Hence, it is important to carefully consider which type of simulation to use.

7 Discussion

In this section the work is discussed in several parts. First off: Vibbi is discussed as a tool for designing garments. After that, the process, tools used as well as the considerations are discussed and evaluated. Then, creating a computer aided design tool for garments using novel technology as well as its possibilities and requirements are discussed. Finally, potential ethical issues and future work is mentioned.

7.1 Vibbi

In this section, Vibbi is discussed as a tool for designing garments. This is divided into discussing some of its features, what it can produce, and what it can be used for.

Vibbi is intended to showcase and explore the potential of integrating quick simulation in the process of designing garments. The tool allows the user to model differently shaped patterns, choose how these patterns should be sewn together, visualize the patterns together in a 3D environment, and finally simulate the patterns as sewn garments. This variety of features is rare on the market and what exist is sometimes considered inaccessible or insufficient. For example, to gain access to VStitcher one has to contact one of its distributors and Marvelous Designer is available as a $ 50.00 monthly subscription.

The decisions made for Vibbi's features are based on observations of related work as well as literature studies of the design process of garments. A decisive factor for every decision was whether or not a feature was required in order to showcase simulation while modeling. These opinions are our own, a user evaluation of Vibbi has not been conducted but would definitely be a great way of continuing the work. It was deemed unnecessary to user evaluate Vibbi in its current and earlier states mainly because usability was not the focus. Testing Vibbi when some level of interactive, bidirectional modeling is achieved is highly considered. User tests could answer what type of interactions are desired by the designer in the 3D view and how are they implemented in an intuitive way.

7.1.1 Starting with a polygon shape

Starting the design process with a polygon only marginally increases the speed in which new patterns can be created. This solution is merely beneficial for implementation, since it has less complexity than for example free drawing. Most desired would be a mix of using pre-made pattern templates and being able to draw freely line by line. Creating and saving template patterns for later use should also be considered, especially since pattern creation is typically initiated by editing an already existing 'base' pattern.

7.1.2 Modeling patterns

Creating different simple patterns works well but Vibbi is missing functionality for making detailed, more advanced patterns. The greatest reason for this is its inability of freely drawing curved boundaries. Drawing curves is an important functionality when designing patterns but it was given low priority since simple patterns were enough for demonstrating interactive modeling.

7.1.3 Unfolding

Unfolding works fine for the typical use case in which the user unfold a pattern once to create symmetry but it has currently no restrictions which can put the user in a hassle. No further work was conducted in order to prevent this or to figure out how unfolding should work more in detail. We endorse freedom while designing with the possibility to regret one's choices. Having that said, implementation of a framework for undoing and redoing actions was instantiated but rather quickly realized to be a too time consuming task to complete.

7.1.4 Avatar

Having an avatar silhouette proved to be a intuitive way of figuring out general size for patterns but sometimes exact measurements are believed to be required, e.g., to make sure that two patterns have the same metric length or to match it with standardized measurements.

Furthermore, it would be interesting to change the avatar to whichever one chooses. It is relatively simple to change the avatar manually using Unity, and it can be implemented in Vibbi relatively simply as well, it was just not prioritized for the project. Changing the avatar would let the user both explore how garments change for different types of bodies - expanding their intuition of the design space - as well as being able to have their own body with exact measurements to try clothes on.

7.1.5 Sewing

Selecting the sewing machine and then selecting the cloth edges to be sewn together is an interaction which simply works, it is closely related to the mental model of sewing. Not as intuitive, is having to indicate where the seam should begin. This becomes an issue since automatically sewing selected edges correctly together is a nontrivial problem to solve. The interaction is not intuitive but luckily not complicated, it becomes simple after it has been taught.

Even harder than having to select where to start sewing is sewing while taking into consideration how the cloth is going to orientate in 3D. The solution we found was to orientate the cloth pieces consistently, namely that the front of the cloth (the side seen while modeling) should always point out from the avatar. Something which potentially can be more intuitive and hence worth consideration, is the interaction to sew directly in 3D once the cloth has already been oriented.

Visualization of seams in 2D and 3D proved to be very useful since sewing tend to be complicated to get right at the first attempt. It increases the speed in which the user understands how it works. It also proved useful during implementation, especially by showing which vertices are paired. This particular technicality is something which we are uncertain whether it should be shown to the user or not. It is good in the way that it explains why and how certain edges get folded. Best would be to have seams which are not as restricted to the technical model, potentially connecting vertices along one edge with points picked freely along the other.

7.1.6 Placing cloth pieces in 3D

Attachment points proved to be a nice and quick way to initiate patterns in 3D. It is intuitive and allows the user to skip plenty of excise in translating and rotating the cloth.

The interactions which Vibbi allows for in 3D are still limited though. To be able to move the pieces with two degrees of freedom was much better than being completely bound by the attachment points. Since it is hard to predict how a user would want to place pieces we emphasize freedom. Aids and shortcuts, such as the attachment points, should be provided where possible but it should still be possible to rotate and translate the pieces with three degrees of freedom.

Another attempt in creating freedom was the algorithm for bending the mesh initially in the 3D environment. It is convenient that the bending is made automatically but it is also a restriction. Knowing how much the user want to bend the mesh is impossible to know for each specific case. The approach has potential but intuitive interactions for letting the user bend the mesh, either in relation to the avatar or freely, are required.

7.1.7 Future directions

No edits made to the 2D patterns while the simulation is running will be transferred to the 3D garments. The Deform Plugin does not allow for making changes to the meshes during the simulation. Even if it did, a strategy for translating potential new vertices from 2D to an already 'curved' 3D mesh is required.

No edits can be made directly on the garments in 3D. Since simulation while modeling was not reached, no work was put into interactions that would alternate the garments directly. The plan was to allow the user to tug the hemlines of sleeves, an interaction not so different from moving a line, except that it is in 3D and typically includes moving more than one line. This would also require a model for what a hemline actually is. Potentially the user can define it similarly to how a seam is defined but preferably Vibbi should figure it out with the existing information. On typical clothing, all the lines which are not sewn are part of a hemline but it is a bit harder to define which lines are part of the same hemline.

With this said we would like to point out that it is not impossible to add interactions that edits the cloth pieces in the 3D environment before they are simulated. This can be enough to achieve a bit of bidirectionally to the design process.

7.1.8 What Vibbi produces

Vibbi produces patterns, garments and intuition about the design space in a rather flexible, fast, and easy manner. Since a curved line can be modeled as a number of finite lines there is almost no limitation to what can be modeled with Vibbi. The issue is how tediously long time it would take to design certain patterns in comparison to analogue pattern making or other available digital tools. Similarly to other tools on the market, Vibbi needs to exploit the flexibility of computers to become faster, more flexible, and easy to use. This includes providing plenty of shortcuts and good design decisions for the pattern making process, such as the ones mentioned previously in this chapter.

Vibbi allows its user to design a variety of clothes, as can be seen in chapter 6: Results. All of these clothes can be considered simple clothing which uniformly consists of one type of soft fabric. They are all designed in a couple of minutes by non experts. The only step in which the user has to wait for the software is during triangulation. Triangulating several large pieces with proper resolution can take up about a minute. If all the patterns for a garment, such as the ones shown in chapter 6, would be triangulated at the same time, it would take about 20 to 30 seconds. Triangulating the patterns one by one as they are being created feels more instant but even then it takes a few seconds.

The simulation is instant and interactive and only needs a few seconds of time to find a state of rest for the garments. It quickly provides useful information about the length and volume of the garment being made. The simulated garment gives a clear picture of how the patterns can be changed in order to gain a different and potentially more desired result. This lets inexperienced designers try their way to proper results, something which we believe is useful for experienced designers as well.

The users of Vibbi can not change which fabric is being used and in particular not change the amount of friction or how much the clothes are stretching. This render the user unable to make for example a belt, which result in it being hard, almost impossible, to design a pair of pants which actually stays on during simulation. Not having proper hips to collide with can potentially also be an contributing factor to this issue.

7.1.9 How Vibbi can be used

Vibbi is missing some key features to be properly used in the process of designing virtual or real clothes. For one, it is not possible to import or export patterns and even if it would there is no metric system implemented into the tool.

Still, Vibbi can be used to try out pattern designs and quickly see the result under physical conditions such as gravity and friction. This can be used to understand concepts such as stretching, wrinkling and folding of cloth as it is worn by a character. It is also a great tool for learning how changes affect the final result.

7.2 Process

As explained in chapter 4: Process the process included a literature study, a tool exploration, implementation planning, and finally the implementation itself. Doing the project in this manner worked quite naturally, as it is rather advantageous to gather information about a subject before you start working with it.

It was decided to make a basically fully implemented prototype in order to be able to point out requirements of novel technology which is still in an early phase of development. Among other things, the background research clarified that the functionality that Vibbi provides is desired by practitioners, which gave us the reason to make a high fidelity prototype. This is in contrast to focusing on for example just one aspect of CAD tools for garment design, such as simulation while modeling or the user experience. In those cases the work can be carried out differently by for example using several low fidelity prototypes and design methods like 'Wizard of Oz' to create the desired effect during a user evaluation.

It was assumed that interactive modeling was desired by the industry and practitioners. With low fidelity prototypes and user tests one can make sure that this is the case. Interactive modeling for garment design is not provided by any tools found on the market and it was assumed the reason for that was limitations in the technology. Hence, limitations in the technology was explored and not whether interactive modeling was desired or not.

7.2.1 Literature study

The literature study gave us important knowledge of the design process of clothes as well as what kind of digital tools there are which improves it already. Most prominently though, was learning about current limitations and the techniques being tried to circumvent them. It made us able to focus on one particular limitation instead of addressing several of them. We went with exploring interactive and bidirectional editing since it involved using novel technology for simulation, and because it is more about interaction than new technically complicated implementations compared with for example sketch-based modeling.

7.2.2 Tool exploration

Exploring already existing tools proved very useful. We tried exploring as many as possible, finding most tools by reading scientific articles that mentioned them. However, many are only available for the industry, being highly specialized and expensive. Some tools were simply a bit outdated, or not interesting enough to try out. Therefore most of the exploration had to be done through watching YouTube videos, following along with tutorials in how the software was used. With the exception of Marvelous Designer.

We were inspired by Marvelous Designer since it is the most available, commercial tool, and also basically the standard tool for creating virtual clothes. What we struggled with when using Marvelous Designer was how our own software should be different, in order to have a scientific contribution. What we eventually landed in was both the fact that Vibbi uses such novel technology that Vivace is, combined with the knowledge gained about the research area around CAD tools for garment design would suffice as the scientific contribution.

7.2.3 Implementation plan

It was decided that Scrum, since it is meant for big organizations with several teams of developers and not for a single two man team, would hinder the implementation more than it would aid it. Still, inspiration was drawn from the Scrum guide and different ASD methods were explored in order to ensure a structured implementation process. FDD, was particularly considered, but it still had to be adapted to the extent of this thesis.

The implementation process was divided into phases for a couple of different reasons. First of all, it is in accordance with ASD, and more specifically FDD. And in line with this, there are inevitably features that depend on other features to be completed before it is even possible to know how they should be defined and implemented. With some anticipation, a great extent of these dependencies can be dealt with in a structured manner. Furthermore, phases makes it easier to focus on the task at hand. Having too many tasks on the agenda at the same time would prove to be less productive; breaking down a huge project into smaller tasks is productivity 101. By defining phases it was possible to also make sure that the work would have some value even if there was not enough time to complete every phase. This is emphasized with ASD, which is good at dealing with unforeseen complications.

The background study may also be the reason why our plan proved really useful. The plan was to the point and the chronology of it made sense throughout the whole project. We believe the work we did the first few weeks of this project put us on a path towards good results, where each step on the way provided some new insights of the complications and requirements of CAD tools for garments.

We found a good balance between keeping the plan specific enough that we knew where we were headed but loose enough to allow for Vibbi to grow and develop as organically as possible. This saved us a lot of time by us not focusing on complex guesswork for each feature since it would have been impossible to know how to implement certain features in advance. Instead, it was good to stay agile; discussing and prioritizing each new feature as they came into attention.

7.2.4 Implementation

The implementation was conducted feature by feature. The features were derived from previous studies of literature and related work and not by human centered methods. It was decided that implementing features similar to how they have been implemented before was good enough for Vibbi. It was not inside the scope to question established interactions for designing patterns. Instead human centered methods was believed to be more useful for the potential new interactions made possible by interactive simulation, but to explore and evaluate such interactions required a certain level of interactive simulation to begin with.

Pair programming helped us in understanding how certain features were implemented without having made them ourselves. It also gave helpful perspective on technically hard implementations. Kanban helped us to break down and keep track of the phases. It particularly helped us in the process of moving from one feature to another since it gave a nice overview of features and their value. Sadly we still ran into some issues during implementation. Solutions that did not work and features that should not have gained as much attention as they did.

An issue we had throughout the project was the ability to realize when a feature, one was currently working on, no longer was worth the time put into implementing it. It was easy to get offtrack and implement something that was missing but not really planned for. At several occasions, at least once a week, we discussed what was most important to get closer to our goal.

Undo/Redo was a typical example of where we spent too much time. While Undo/Redo is something that will be really important for the end result of any design tool, in the end it was not so important for this particular project. While implementing the developer is so consumed by the task at hand that it can be really hard to put the work into perspective. Asking important questions, such as if a feature is really necessary, is the job of a product owner, a role that were missing. The developer want to solve the problems not keep track of the full picture.

7.3 Tools

A big part of this project was to use and integrate Unity and Vivace. Following are some advantages and disadvantages in using the two tools to create Vibbi.

7.3.1 Vivace

The Deform Plugin serves its purpose of exposing Vivace's functionality within the Unity framework, it was of great help when implementing Vibbi. The provided scenes and scripts are extra useful since they not only show what can be done with Vivace but also how to do it.

The plugin is written to let the user try out Vivace's functionality by utilizing the Unity editor. It makes it simple to turn a GameObject into a DeformableObject in the editor and then watch it continuously being simulated when the application is running. That is probably the most typical use case of how the Deform Plugin will be used considering that Unity is made for creating games. However, making a modeling tool, is not the typical use case for Unity, hence there were some issues using the Deform Plugin as well.

Some early confusions arose due to the fact that DeformBody, DeformObject, and DeformCloth are written as if they are GameObjects even though they are Components in Unity's framework. It is not only because of the names but also the extensive usage of hiding functionality and variables from the Unity editor. One Component can require other Components but hiding them in the editor was a bit disorienting at first.

Adding a DeformBody Component to a GameObject in the Unity editor is very intuitive but adding it during run time: not as much. It took a lot of time to figure out *what* happens *when* in the DeformManager and DeformBody. For example, Unity has a method called `OnValidate()` which runs automatically whenever the user changes a value using the Unity editor. This method is responsible for setting the mesh of the DeformBody, but it is not triggered properly when manipulating the DeformBody in the code.

The Collider Component made by Unity is extensively used when creating any type of application in Unity. Preferably, DeformCollider can extend Collider, since colliders are used any way. It also adds flexibility to what objects can be used as colliders. Alternatively, DeformManager can potentially handle Unity's Colliders as well as DeformColliders.

Once meshes are loaded to Vivace, translations and rotations done in Vibbi misalign the view from the model. It is a bit confusing to know what is allowed and when, which makes it easy to make changes which creates a gap between the model, and the view. It is possible to for example translate, rotate and scale objects with a DeformObject Component in the Unity editor without it affecting the object being simulated.

Vivace was experienced as a bit unstable at times. Trying to shutdown the plugin without any loaded models, simply using mac OS, or making edits in the code while Unity was running are all examples of what frequently made Unity crash without any type of feedback.

Since it is too demanding for the simulation to be able to update itself with new meshes it would make sense for Vivace to handle triangulation internally - at least for cloth simulation. Vibbi can provide the ID and desired boundaries of the cloth models and then Vivace updates or creates the mesh according to what will be technically feasible within an interactive time frame. This aligns with how the designer work mostly with the boundaries of the mesh and not the internal structure.

To make intuitive interactions easier to achieve for developers, quick algorithms for finding and moving vertices along the boundaries of the models would be useful. The possibility to add and remove information about the seams and cloth pieces during run time are also desired of the technology. This includes, we believe, changing the resting state of the meshes while the simulation is running. Even better is to be able to add new vertices to the existing meshes.

Furthermore, the seam model has to be more closely related to seams in reality. Pairing vertices are a bit restricted and only intuitive for the one making the technology, a more intuitive model for sewing boundaries together, e.g., pairing arbitrary points along the boundaries, would be an improvement. Potentially, it can also be possible to visualize the tension in the cloth model, currently it can be hard to see if the garment is inhumanly tight.

7.3.2 Unity

Overall, Unity is intuitive to use and provides great documentation of all its features. There are more advantages than disadvantages in using Unity as a development platform.

Making GameObjects and Prefabs proved to be a nice way of defining the pieces of the different models without having to write much code at all. Using Unity's Mesh class was also easy and provided plenty of code which would have had to be implemented anyway.

Setting up the 3D environment in Unity was simple except for achieving the desired lighting. The environment tended to be too bright or too dark regardless of the amount, type and direction of light sources. Unity provides plenty of functionality for lighting 3D scenes which proved to be a bit too complicated for novice users who want to make a simple dressing room.

Making different looking materials and adding them to objects is also really easy in Unity. It is also impressive how simple it is to include functionality, images, materials, and meshes found elsewhere.

The Game Loop and 3D space was not required or particularly suited for making 2D modeling of patterns. Scripting objects to follow the mouse as it moves over the screen can be easier when the user has decided to make an application without depth.

There were some issues with version control and the scene files created by Unity. The scene files includes a lot of references to objects which are hard to interpret, i.e., they are basically large files with a lot of incomprehensible letters and numbers. The issue was avoided by copying scenes and working independently with different features on separate copies. This required scenes to be manually merged together, which usually left bugs which were hard to trace.

7.4 Considerations

Creating a hi-fidelity prototype of an interactive modeling tool for designing virtual clothes includes considering a number of different things. What should be considered has been brought up in chapter 6: Results and can be summarized as: what external tools to use, how to work, how to model patterns, how to sew patterns together, how to dress the avatar, and how to simulate the clothes. These considerations were gained from studying literature regarding CAD tools and simulation of cloth as well as the practical work of planning and implementing Vibbi.

However, as no user evaluations have been performed on Vibbi the reliability of these findings can be argued. At the same time, the considerations are just that, they are not requirements. These are simply issues that we had to overcome when developing Vibbi, and they are issues that we believe will persist when developing any interactive, garment modeling tool. There can also be a lot more things to consider, especially when getting to a point where users are more involved in the development process. We believe these considerations are a good starting guide for anyone who is planning to develop a tool like Vibbi.

7.5 Computer aided design tool for garments

Today, garment design are mainly done in an analogue way even though there exist computer aided design tools as digital alternatives. Still, there is great potential for future computer aided design tools since the required underlying technology is improving. In this section, some possibilities to improving the design process of garments through digital means are mentioned and their requirements are discussed.

7.5.1 Possibilities

In order to know what is required of a digital tool to improve the design process of garments, one first has to answer the question: what is an *improvement* of the design process? As brought up in chapter 2 Technical Background, there are many ways in which computer aided design tools already improves the process of designing garments. From our literature study and during the process of making Vibbi, we gathered some other potential areas of improvement which currently are being explored to different extent.

To improve the design process of garments, a computer aided design tool can:

1. Allow the draping process to be done in virtual space.
2. Allow for trying out patterns as garments on virtual bodies.
3. Provide the user with exploratory options.
4. Aid the user in gaining intuition about the design space.

Allowing the draping process to be fully conducted in virtual space is good in a couple of ways. First of all, draping a virtual body has no material requirement, the designer can do it over and over again without having to use extra fabric. Also, a virtual body has the potential of changing size after desire in almost no time, which means that a designer can try out different patterns and measurements for different body sizes without involving a single real body.

Garment design is complicated and will most certainly require a couple of iterations along the way. Trying out patterns on virtual mannequins and bodies would lessen the time consumed by shortening the length of required iteration steps.

A fashion designer typically explores different fabrics, patterns, and garment collections to gain inspiration for new creative designs. A CAD tool for garments has to take this in consideration and provide the user with exploratory options. Being a digital tool, these options can be available in a more accessible manner when compared to the analogue design process, and thereby improve the creative process.

Designing garments on the computer demands no requirement for material, physical tools, or physical space which makes the design process more accessible for all kinds of users. Even if the tool used does not produce actual garments it still aids designers in gaining intuition about the design space which has great pedagogical value.

7.5.2 Requirements

Achieving the above mentioned improvements of the design process comes with requirements, both for the design of the CAD tool as well as the underlying technology. Thus, we present four requirements of a digital garment design tool which we believe can improve the design process of clothes:

1. Intuitive interactions for manipulating cloth and garments in 3D space.
2. Reliable simulation of cloth and garments.
3. Compatibility with other digital and non digital tools.
4. Flexibility to aid and not hinder the designer's imagination.

Intuitive interactions
To allow the draping process to be done in virtual space the tool requires intuitive interactions, more specifically interactions such as: placing, pinning, rotating, and folding pieces of cloth in 3D on a suitable avatar. The user should not have to know how the pieces of cloth, the seams, or the folds are modeled, these objects should behave as similarly as possible to how they behave in reality. Thus giving the user the incentive to use the tool. On this topic, we will state and discuss whether and how Tangible Modeling, Sketch-Based Modeling, and Interactive Modeling (from section 2.2: Computer aided garment design) is involved.

What Interactive Modeling lack is the ability to translate the draping process of reality to the computer. Intuitive interactions of handling objects in a 3D space when using a mouse on a 2D computer screen are tricky to accomplish, see Figure 7.1. Particularly the sense of depth is lost during these interactions, which gives the user a loss of control.

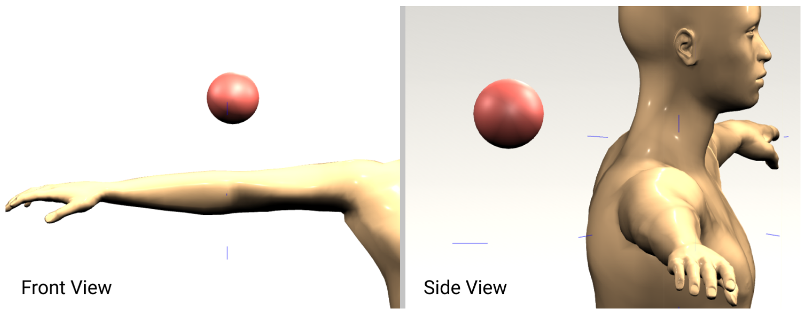

Figure 7.1: Illustrating the difficulties of placing objects in 3D when working in 2D.

With these circumstances it is reasonable to provide the user both the freedom of being able to move their cloth pieces with three degrees of freedom, as well as providing constrained options such as the attachment points in Vibbi. Especially useful is giving the user the freedom to add their own attachment points since this ensures that the piece of cloth will appear close to the avatar. Still, there is a gap between the intuitive interactions of the real world and the digital.

This issue is addressed with Tangible Modeling tools, and there is also great potential in using Virtual Reality. These two approaches to digital garment design have greater potential of offering the user with intuitive interactions, allowing them to stay and design in the 3D space. However, in order for these modeling tools to be beneficial in the long run they need to be exact and follow the movements of the user without fail. In conjunction with this statement, the question is if these tools are suitable for drawing the actual patterns, in Dress-Up for example they draw the conclusion that their method is not 100% accurate and is more suitable for initial prototyping [6]. Rather, these tools are more suitable for dressing an avatar with already finished pattern pieces.

In Sketch-based modeling this issue is ”addressed” by simply removing the interaction completely. Instead opting for an approach where the computer translates the user’s sketch directly into a 3D garment. Something which, if it works well, saves a lot of time. With that said, since this approach omits the draping process from the user, it omits a process which traditionally is where much of the creative work of a garment occurs.

Reliable simulation

In order to shorten the length of the iteration step of trying out patterns on an avatar, the tool is required to have fast and reliable simulation. Preferably, the user should be able to continuously make updates to the simulated garment, and be able to directly manipulate it. The immediate feedback that a physically correct simulation will provide the user with is essential to gaining a better intuition of the design space.

This requires models of cloth, seams, and garments which can be changed frequently by the user at the same time as equilibrium states can be found at interactive rates.

Compatibility
Providing the user with exploratory options such as different fabrics, patterns and garments requires compatibility. The design process of clothes will inevitable include several different digital and non-digital tools. If a designer has found an interesting fabric it should be possible to add it to the digital tool. The same goes for patterns that has been drawn with pen and paper.

It should also be easy to export and import patterns, textures, garments, avatars, etc., from one digital tool to another. Maybe the designer uses one tool for pattern making and another for draping, this should be possible and also designed for. Pattern making and draping can be seen as two different professions conducted by two different persons, potentially it would be better to make tools which address their needs separately instead of making one tool which can handle the whole process.

Flexibility
Aiding designers in gaining intuition about the design space requires the tool to be flexible. The design process looks different for different types of garments. In the real world, the imagination of the designer is figuratively the only limit to what can be designed, and therefore the tool has to at least provide an equal amount of freedom. The tool cannot remove anything that is possible to do in reality, but instead both provide the same utilities *and* offer digital tools that will improve the design process, such as undo/redo, copy/paste, and bidirectional editing. In virtual space, sleeves can become longer with a swipe of the mouse and fabrics can become stiffer with a simple click.

7.6 Ethical

Making it easier, more accessible and mainstream to create nicely designed clothes can reshape the clothing industry as it is today. With a tool that can correctly visualize the result in 3D, given a 2D pattern, a potential reduction of time on iterating the design is to be expected. It would also reduce the time for which it takes to get the experience required for designing desirable clothes. Even though it is made in order to improve the process of designing clothes, it can potentially have a negative impact on established clothing industries and experienced fashion designers, which have put a lot of time and effort into becoming professionals. In a similar manner to how recent platforms that provide content online has had an impact on the traditional publishing industry.

7.7 Future work

There are plenty of ways in which one can continue to explore what is required of a computer aided design tool in order to improve the process of designing garments.

We suggest five areas for future work:

1. Interactive, bidirectional editing
2. Intuitive virtual draping
3. Simulation while modeling
4. User evaluation of existing tools
5. Simulation of garments on a moving avatar

7.7.1 Interactive, bidirectional editing

In Vibbi, as well as other tools, it would be interesting to test 3D interactions such as: tugging hemlines, sewing, changing volumetric size of garments, moving darts along edges, cutting, etc. There are much to be explored in order to make this type of interaction even more intuitive, flexible and usable but it has potential of making users learn more about the design space. At least if they not only affect the garments directly but the patterns as well. It is complicated to simply look at patterns and understand how they would look like when draped on a body, but watching them change while editing the garments would eventually refine the ability to do so.

7.7.2 Intuitive virtual draping

Three dimensional interactions has worth even if there is no simulation involved. It is complicated to drape a body in virtual 3D space. Intuitive ways of placing, rotating, bending and pinning pieces of clothes freely on a virtual body would prove useful for CAD tools for creating garments. A virtual body has the potential of changing detailed size by desire which would require complicated mechanical inventions or several human models to achieve in reality.

7.7.3 Simulation while modeling

Having simulation while modeling would improve the learning outcome of Vibbi substantially. During simulation, the user has to drag the clothes in place in order to get a good representation of the actual result, something which has to be done all over again if the simulation restarts. A designer would like to add, remove and change stuff as the garment is being simulated. How to make such interactions both intuitive and technically feasible is a complex task. Hence, we still believe more work should be conducted towards the goal of achieving simulation while modeling.

7.7.4 User evaluation of existing tools

Very few user evaluation studies of existing tools were found during the background study. This made it complicated to motivate interactions which are complicated to implement. Collecting data regarding which tools are used, for what they were used and what their limitations are according to users would be useful. Data regarding the parts of the design process which are not done on the computer and the designers reasons for this would probably prove even more useful.

7.7.5 Simulation of garments on a moving avatar

Seeing designed garments in action on a moving avatar before making them in real would catch certain aesthetics which would be impossible to foresee with a static one. This can potentially result in more accessible knowledge about different fabrics as well as wrinkles and folds. Knowledge which are gained from years of practice and trials.

8 Conclusion

Designing garments is a skill that is hard to master. It is an iterative process which includes pattern creation, draping and tailoring. There is a gap between prototypes and the finished result which is believed can be shortened by visualizing both patterns and the corresponding garments in virtual space.

Simulating the garments as the patterns are iterated will potentially increase the designers intuition about the design space which will result in designing garments being easier to master. Quick, interactive editing of garments on a computer can also potentially shorten the time between prototype and the finished result.

With the purpose of answering the research question "*What should be considered when making an interactive modeling tool for designing virtual clothes?*" an interactive modeling tool, named Vibbi, was made. With Vibbi, a user can model differently shaped patterns, choose how these patterns should be sewn together, visualize the patterns together in a 3D environment, and finally simulate the patterns as sewn garments.

Vibbi was made using agile software development, more precisely a version of feature driven development in conjunction with structural tools such as Kanban and VCS, and software development techniques such as pair programming. Before beginning work on the implementation of Vibbi a literature study on the subject of CAD tools for garment design was performed, and an exploration of related tools was done.

From this work several considerations answering the initial research question were found, where the most apparent subjects can be summarized as: what external tools to use, how to work, how to model patterns, how to sew patterns together, how to dress the avatar, and how to simulate the clothes. While these considerations are not the only ones needed to be taken when creating this type of tool, as they are heavily based on the specific development of Vibbi, we consider them a good start and guide for anyone who is considering creating this type of tool.

Future work on Vibbi can be done to achieve interactive simulation of garments with bidirectional editing of patterns, which includes intuitive ways of editing garments in 3D space. Another approach will be to conduct user evaluation of existing tools, since it will steer research in the direction of making garment design more accessible.

Overall the user experience and interface of Vibbi can see improvement, and above all benefit from rigorous user evaluations.

Bibliography

[1] Jerry Weil. “The synthesis of cloth objects”. In: *ACM Siggraph Computer Graphics* 20.4 (1986), pp. 49–54.

[2] Melissa A Toups et al. “Origin of clothing lice indicates early clothing use by anatomically modern humans in Africa”. In: *Molecular biology and evolution* 28.1 (2010), pp. 29–32.

[3] Market Watch. *Annual Financials for H&M Hennes & Mauritz AB Series B.* URL: `https://www.marketwatch.com/investing/stock/hmb/financials?countrycode=se` (visited on 02/07/2018).

[4] Charlene Heezen. *Top 5 Richest Fashion Designers.* URL: `http://amayzine.com/en/2015/top-5-richest-fashion-designer` (visited on 02/07/2018).

[5] Nobuyuki Umetani et al. “Sensitive Couture for Interactive Garment Modeling and Editing”. In: *ACM Trans. Graph.* 30.4 (July 2011), 90:1–90:12. ISSN: 0730-0301. DOI: `10.1145/2010324.1964985`. URL: `http://doi.acm.org/10.1145/2010324.1964985`.

[6] Amy Wibowo et al. “DressUp: a 3D interface for clothing design with a physical mannequin”. In: *Proceedings of the Sixth International Conference on Tangible, Embedded and Embodied Interaction.* ACM. 2012, pp. 99–102.

[7] Marco Fratarcangeli, Valentina Tibaldo, and Fabio Pellacini. “Vivace: A practical gauss-seidel method for stable soft body dynamics”. In: *ACM Transactions on Graphics (TOG)* 35.6 (2016), p. 214.

[8] University of Fashion. *Draping: What is Draping?* URL: `https://www.universityoffashion.com/disciplines/draping/` (visited on 02/01/2018).

[9] Luke Olsen et al. “Sketch-based modeling: A survey”. In: *Computers & Graphics* 33.1 (2009), pp. 85–103.

[10] Emmanuel Turquin et al. “A sketch-based interface for clothing virtual characters”. In: *IEEE Computer graphics and applications* 27.1 (2007).

[11] Yuki Mori and Takeo Igarashi. “Plushie: an interactive design system for plush toys”. In: *ACM Transactions on Graphics (TOG).* Vol. 26. 3. ACM. 2007, p. 45.

[12] Gabriel A Wainer. *Discrete-event modeling and simulation: a practitioner’s approach.* CRC press, 2017.

[13] David M Gaba. “The future vision of simulation in health care”. In: *BMJ Quality & Safety* 13.suppl 1 (2004), pp. i2–i10.

[14] Hing N Ng and Richard L Grimsdale. “Computer graphics techniques for modeling cloth”. In: *IEEE Computer Graphics and Applications* 16.5 (1996), pp. 28–41.

[15] Pascal Volino, Frederic Cordier, and Nadia Magnenat-Thalmann. “From early virtual garment simulation to interactive fashion design”. In: *Computer-aided design* 37.6 (2005), pp. 593–608.

[16] Vajiha Mozafary and Pedram Payvandy. “Study and comparison techniques in fabric simulation using mass spring model”. In: *International Journal of Clothing Science and Technology* 28.5 (2016), pp. 634–689.

[17] T Agui, Y Nagao, and M Nakajma. “An expression method of cylindrical cloth objects-an expression of folds of a sleeve using computer graphics”. In: *Trans. of Soc. Of Electronics, Information and Communication, Col. J73-D-II* 7 (1990), pp. 1095–1097.

[18] BK Hinds and J McCartney. “Interactive garment design”. In: *The Visual Computer* 6.2 (1990), pp. 53–61.

[19] J McCartney and BK Hinds. “Computer aided design of garments using digitized three-dimensional surfaces”. In: *Proceedings of the Institution of Mechanical Engineers, Part B: Journal of Engineering Manufacture* 206.3 (1992), pp. 199–206.

[20] J Ascough, HE Bez, and AM Bricis. “A simple beam element, large displacement model for the finite element simulation of cloth drape”. In: *Journal of the Textile Institute* 87.1 (1996), pp. 152–165.

[21] L Gan, NG Ly, and GP Steven. “A study of fabric deformation using nonlinear finite elements”. In: *Textile Research Journal* 65.11 (1995), pp. 660–668.

[22] O. Etzmuss, M. Keckeisen, and W. Strasser. “A fast finite element solution for cloth modelling”. In: *11th Pacific Conference onComputer Graphics and Applications, 2003. Proceedings.* Oct. 2003, pp. 244–251. DOI: 10.1109/PCCGA.2003.1238266.

[23] Jinlian Hu, Shui-Fu Chen, and JG Teng. “Numerical drape behavior of circular fabric sheets over circular pedestals”. In: *Textile Research Journal* 70.7 (2000), pp. 593–603.

[24] Demetri Terzopoulos et al. “Elastically deformable models”. In: *ACM Siggraph Computer Graphics* 21.4 (1987), pp. 205–214.

[25] Carl Richard Feynman. “Modeling the appearance of cloth”. PhD thesis. Massachusetts Institute of Technology, 1986.

[26] Xiaoqun Dai, YI Li, and Xin Zhang. “Simulating anisotropic woven fabric deformation with a new particle model”. In: *Textile research journal* 73.12 (2003), pp. 1091–1099.

[27] Z Yueqi and W Shanyuan. “Cloth modeling based on particle system”. In: *Journal - Dong Hua University - English Edition* 18.2 (2001), pp. 41–44.

[28] Bernhard Eberhardt, Andreas Weber, and Wolfgang Strasser. “A fast, flexible, particle-system model for cloth draping”. In: *IEEE Computer Graphics and Applications* 16.5 (1996), pp. 52–59.

[29] Jian Dong Yang and Shu Yuan Shang. “Cloth modeling simulation based on mass spring model”. In: *Applied Mechanics and Materials*. Vol. 310. Trans Tech Publ. 2013, pp. 676–683.

[30] Liu Zhengdong and Shang Shuyuan. “Notice of Violation of IEEE Publication Principles A mass-spring model for real time cloth deformation”. In: *2011*

International Conference on Multimedia Technology. July 2011, pp. 2845–2848. DOI: 10.1109/ICMT.2011.6001862.

[31] Feng Ji, Ruqin Li, and Yiping Qiu. “Three-dimensional garment simulation based on a mass-spring system”. In: *Textile Research Journal* 76.1 (2006), pp. 12–17.

[32] Jing Hu et al. “Cloth Simulation with a Modified Implicit Method Based on a Simplified Mass-Spring Model”. In: *Applied Mechanics and Materials.* Vol. 373. Trans Tech Publ. 2013, pp. 1920–1926.

[33] Wenqing Huang et al. “Cloth Simulation Based on Simplified Mass-Spring Model”. In: *Indonesian Journal of Electrical Engineering and Computer Science* 12.5 (2014), pp. 3811–3817.

[34] Corey O’Connor and Keith Stevens. “Modeling cloth using mass spring systems”. In: *Appl. Soft. Comput* 12 (2003), pp. 266–273.

[35] Serkan Bayraktar. “Simulating cloth behavior by using mass-spring networks”. PhD thesis. bilKent university, 2002.

[36] Gerald Farin. *Curves and surfaces for computer-aided geometric design: a practical guide.* Elsevier, 2014.

[37] Mario Botsch et al. *Polygon mesh processing.* CRC press, 2010.

[38] Atul Narkhede and Dinesh Manocha. “Fast polygon triangulation based on seidel’s algorithm”. In: *Graphics Gems V* (1995), pp. 394–397.

[39] Raimund Seidel. “A simple and fast incremental randomized algorithm for computing trapezoidal decompositions and for triangulating polygons”. In: *Computational Geometry* 1.1 (1991), pp. 51–64.

[40] Matthew T Dickerson et al. “Fast greedy triangulation algorithms”. In: *Computational Geometry* 8.2 (1997), pp. 67–86.

[41] M. I. Shamos and D. Hoey. “Closest-point problems”. In: *16th Annual Symposium on Foundations of Computer Science (sfcs 1975).* Oct. 1975, pp. 151–162. DOI: 10.1109/SFCS.1975.8.

[42] Daniel Cohen-Or and Yishay Levanoni. “Temporal continuity of levels of detail in delaunay triangulated terrain”. In: *Visualization’96. Proceedings.* IEEE. 1996, pp. 37–42.

[43] Boris Delaunay. “Sur la sphere vide”. In: *Izv. Akad. Nauk SSSR, Otdelenie Matematicheskii i Estestvennyka Nauk* 7.793-800 (1934), pp. 1–2.

[44] Der-Tsai Lee and Bruce J Schachter. “Two algorithms for constructing a Delaunay triangulation”. In: *International Journal of Computer & Information Sciences* 9.3 (1980), pp. 219–242.

[45] Matthias Müller et al. “Position based dynamics”. In: *Journal of Visual Communication and Image Representation* 18.2 (2007), pp. 109–118.

[46] Yousef Saad. *Iterative methods for sparse linear systems.* Vol. 82. siam, 2003.

[47] Sofien Bouaziz et al. “Projective dynamics: fusing constraint projections for fast simulation”. In: *ACM Transactions on Graphics (TOG)* 33.4 (2014), p. 154.

[48] Jan Bender et al. “A Survey on Position-Based Simulation Methods in Computer Graphics”. In: *Comput. Graph. Forum* 33.6 (Sept. 2014), pp. 228–251. ISSN: 0167-7055. DOI: 10.1111/cgf.12346. URL: https://doi.org/10.1111/cgf.12346.

[49] Association of Modern Technologies Professionals. *Software Development Methodologies*. Visited on 2018-03-14. URL: http://www.itinfo.am/eng/software-development-methodologies/.

[50] Ken Schwaber and Jeff Sutherland. *The Scrum Guide*. Visited on 2018-03-14. URL: http://www.scrumguides.org/scrum-guide.html.

[51] Zahid Nawaz, Shabib Aftab, and Faiza Anwer. “Simplified FDD Process Model”. In: *International Journal of Modern Education and Computer Science* 9.9 (2017), p. 53.

[52] John Hunt. “Feature-driven development”. In: *Agile Software Construction* (2006), pp. 161–182.

[53] L. Williams. “Integrating pair programming into a software development process”. In: *Software Engineering Education and Training, 2001. Proceedings. 14th Conference on.* 2001, pp. 27–36. DOI: 10.1109/CSEE.2001.913816.

[54] Laurie Williams et al. “Strengthening the case for pair programming”. In: *IEEE software* 17.4 (2000), pp. 19–25.

[55] Rhode Code. *Version Control Systems Popularity in 2016*. Visited on 2018-03-27. URL: https://rhodecode.com/insights/version-control-systems-2016.

[56] *Kanban*. Visited on 2018-05-17. URL: https://www.atlassian.com/agile/kanban.

[57] *Case Studies*. Visited on 2018-04-23. URL: https://www.marvelousdesigner.com/cases/.

[58] *Unity Tutorials*. Visited on 2018-04-23. URL: https://unity3d.com/learn/tutorials.

[59] Jonathan Richard Shewchuk. “Triangle: Engineering a 2D Quality Mesh Generator and Delaunay Triangulator”. In: *Applied Computational Geometry: Towards Geometric Engineering*. Ed. by Ming C. Lin and Dinesh Manocha. Vol. 1148. Lecture Notes in Computer Science. From the First ACM Workshop on Applied Computational Geometry. Springer-Verlag, May 1996, pp. 203–222.

[60] David Vandevoorde. *The Maximal Rectangle Problem*. Visited on 2018-04-24. 1998. URL: http://www.drdobbs.com/database/the-maximal-rectangle-problem/184410529.

[61] *Unity UISystem*. Visited on 2018-05-04. URL: https://docs.unity3d.com/Manual/UISystem.html.

[62] *Mixamo*. Visited on 2018-05-04. URL: https://www.mixamo.com/#/.

[63] *What is the best alternative to Unity?* Visited on 2018-06-06. URL: https://www.slant.co/options/1047/alternatives/~unity-alternatives.

[64] Michael Keckeisen, Matthias Feurer, and Markus Wacker. “Tailor Tools for Interactive Design of Clothing in Virtual Environments”. In: *Proceedings of the ACM Symposium on Virtual Reality Software and Technology*. VRST ’04. Hong Kong: ACM, 2004, pp. 182–185. ISBN: 1-58113-907-1. DOI: 10.1145/1077534.1077572. URL: http://doi.acm.org/10.1145/1077534.1077572.

[65] Hugues Hoppe. “Progressive Meshes”. In: *Proceedings of the 23rd Annual Conference on Computer Graphics and Interactive Techniques*. SIGGRAPH ’96. New York, NY, USA: ACM, 1996, pp. 99–108. ISBN: 0-89791-746-4. DOI: 10.1145/237170.237216. URL: http://doi.acm.org/10.1145/237170.237216.

A

Implementation plan

This section describes our planned method to create and design Vibbi, an interactive, bidirectional modeling tool for designing virtual clothes. By designing Vibbi we intend to create a conceptual rich artifact in order to explore simulation while modeling.

A.1 The problem

There are several tasks and subtasks that has to be completed in order to create a modeling tool for virtual garments. We have decided to perform these in the following five phases, described shortly in Table A.1 below, with an additional sixth phase containing additional implementations if time proves superfluous.

An in depth explanation of each phase is in the following sections.

Phase	Description
1	The simulation of a generic virtual cloth with any shape using Vivace.
2	A 2D user interface in which the user can create pieces of cloth.
3	A 3D view with an avatar to which the pieces of cloth can be added.
4	It shall be possible to do edits whilst the simulation is running.
5	Edits shall be possible directly on the simulated 3D garments.
6	Additional: Scaling or moving avatar or multilayered garments.

Table A.1: Phases of the project.

A.2 Phase 1: Simulation of a Generic Cloth

The first implementation will be the simulation of a generic virtual cloth. For Vibbi to ultimately be able to simulate garments (e.g shirts or pants), it first of all has to be able to simulate differently shaped pieces of cloth. This requires a cloth model which can be represented in 3D and simulated using a physics engine - e.g., Vivace.

First of all, a cloth model is necessary, and it was chosen to use triangle meshes. The main reason was that Vivace takes indices and vertices as input for soft body simulation - just like many other physics engines. Another reason for choosing triangulated meshes is that it also can model 2D virtual patterns, which is preferred for real world manufacturing.

In order to create triangulated meshes, the plan is to use Unity and write a script for performing naive triangulation on polygon shapes created with the Unity editor. When this step is reached the script should be modified to support Delaunay triangulation as well. The next step is then to integrate the Vivace engine into Unity and then properly send the vertex data for the triangulated mesh to the engine so that it can simulate the initial 2D shape as a piece of cloth.

A.3 Phase 2: A 2D interface for modeling cloth

Secondly, someone using Vibbi should be provided with an interface for creating and editing 2D polygon shapes. These shapes will serve as patterns for the garment that is being made. This requires a few different computer-aided modeling operations. It also requires a seam model which allows for virtually sewing the pieces of cloth together.

As for how to make the model editable in an intuitive way, the user will be provided with a canvas where they can edit a shape. We chose primarily to let the user have control over the boundaries of a predefined polygon shape. This polygon shape is defined by boundary lines and boundary points connecting the lines. The user should then be able to edit the shape in a couple of different ways:

1. Click and move a point, see Figure A.1a
2. Click and move a line, see Figure A.1b
3. Add or remove a point, see Figure A.2

Moving a point can be done with two degrees of freedom. Moving a line is done in the perpendicular direction of the line. Adding a point is done through some kind of interaction with an existing boundary line, for example right clicking or double clicking. Removing a point is done by selecting a point and then pressing delete, the user cannot remove points so that the shape is not closed - i.e., there should be at least three points left. An avatar, which later will size-correlate with the 3D avatar,

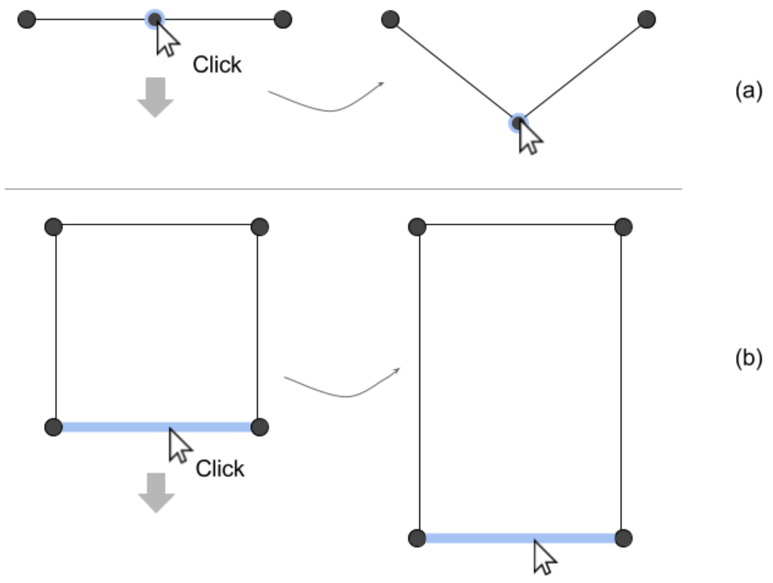

Figure A.1: Dragging points and lines to change the shape. Figure (a) shows the user moving a point downwards. In (b) the user moves a boundary line

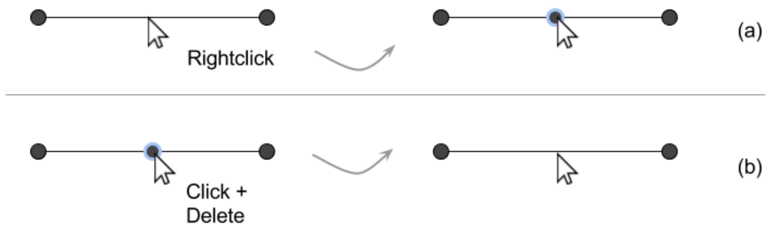

Figure A.2: Adding and removing points. Figure (a) shows the user right-clicking on a line in order to add a point. Figure (b) shows the user selecting a point and removing it

will be displayed on the canvas, making it easier for the user to make correctly sized shapes, see figure A.3.

By providing the user with a button to add a new polygon shape, i.e., a piece of cloth, the user can edit two shapes at once. The user should be able to move cloth with two degrees of freedom, in order to organize the work space. The user should be able to select and copy/paste a piece of cloth. This can be done by a so called unfolding where the cloth is mirrored symmetrically through a line.

Boundary lines of these pieces of cloth can be selected to be sewn together. These boundaries can belong to the same cloth or two different pieces of cloth, but the boundaries should not overlap. Two boundaries are sewn together by connecting and merging pairs of vertices [64]. A vertex can be subject to more than one seam at once. Boundaries that are of different lengths will be mapped together so that the longer boundary folds into a shorter one of equal length with the other boundary.

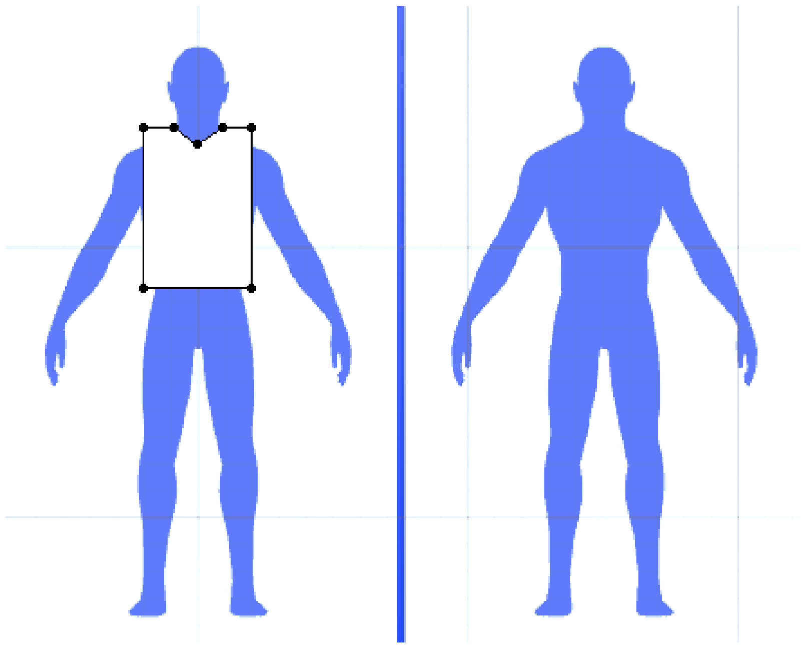

Figure A.3: A silhouette of the 3D avatar will be placed on the canvas to aid the designer in making correctly sized pieces of cloth.

This will create so called pleating, and will require some manual modification of vertices in the model to achieve nice folds.

Darts will be implemented as it is described for Sensitive Couture [5]. Darts are triangular folds that provide shape to a garment. They are a quick way of cutting and sewing at the same time and will be made by drawing a line over a piece of cloth. They will start off with a symmetric shape but should be editable similarly to the other boundaries. Furthermore, if the dart intersects a boundary line, it should be possible to move the dart along the line.

A.3.1 Additional features in 2D interface

The amount of tools and aid that can be provided to the user is tremendous, but implementing those are not in the scope of this project. There are some things that are more prominent to implement though, such as an aid in dragging straight when moving points and lines around. Usually in editing tools the user is provided with this aid by holding the SHIFT-key while dragging. Another thing is the ability to let the user draw their own closed shapes with a line-tool and providing the possibility to create curves. We have decided to mainly focus on straight lines as a proof of concept, but curves are ultimately an integral part of creating garments.

A.4 Phase 3: A 3D environment for garments

Thirdly, as the patterns and seams are created, Vibbi will generate garments as a result. These garments have to be visualized to the user, preferably while being simulated. This requires a virtual 3D environment containing an avatar, which will be wearing the garments. It also implies that the user can define how the garments are meant to be worn, since this is complicated for a computer to guess.

The user should be able to view the made garments in a 3D view. Depending on the complexity of implementation, the user should be able to view it with three degrees of freedom, see Figure A.4. Initially, a top down view at an angle will be sufficient. The user should be able to place the designed pieces of cloth on the body of a 3D avatar. To simplify this process, Front Top, Back Top, Front Bottom, Back Bottom, Left Arm, and Right Arm are potential attachment points. The user should be able to start, stop, and restart the simulation of clothes. Restarting implies that the clothes should start at their initial determined positions. The seams between pieces should be visualized.

A.5 Phase 4: Simulation while modeling

Fourthly, Vibbi shall be able to handle the edits from the 2D interface that are being made whilst the simulation is running. As the avatar is dressed, the user can choose to continually edit the patterns to alternate the result. This requires the implementation of a real time system which is able to quickly interpret the operations as well as recalculate the model and continue the simulation without seamless effort.

Using Unity to implement the system provides the real time element through its inbuilt game loop that continually updates and renders every object on the screen. In order to reach interactive rates while modeling it is most likely necessary to use strategies such as progressive refinement [65] and sensitive interaction [5]. Basically doing rough calculations while the user is editing and doing the finer calculations when they are not.

A.6 Phase 5: Editing of 3D garments

Fifthly, it will be more intuitive to make certain changes directly on the simulated garment. Instead of editing the 2D pattern, the user shall be able to make modifications of the 3D garment and when doing so the result shall be visualized in 2D as well.

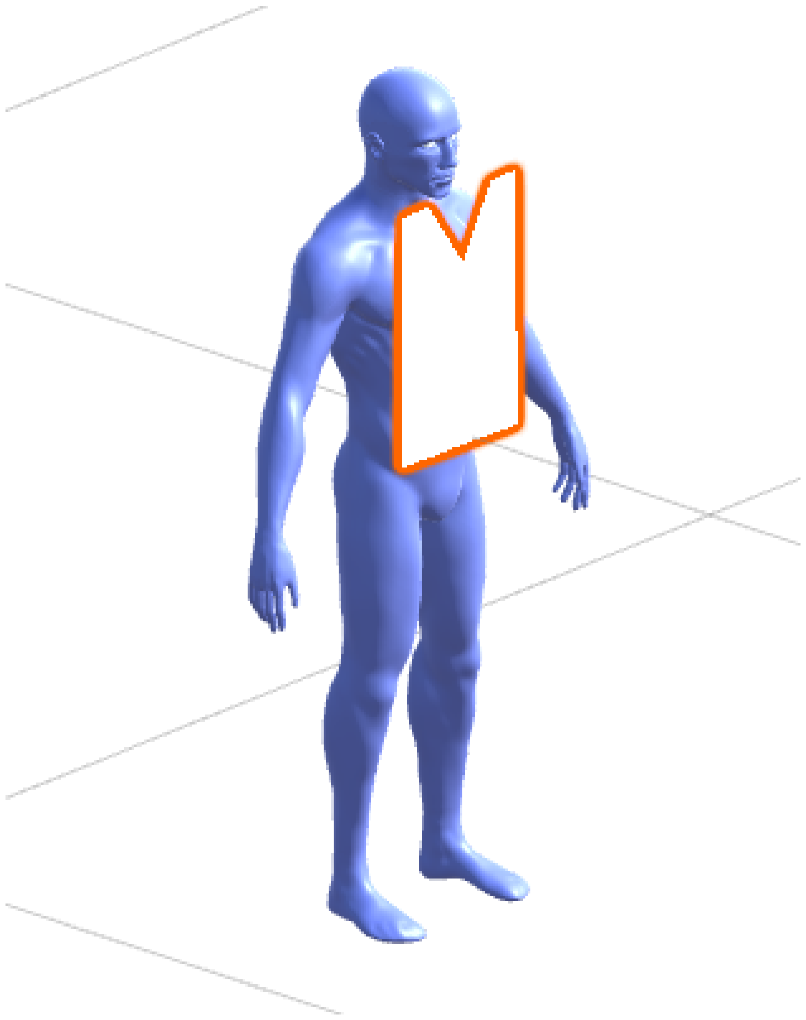

Figure A.4: Vibbi will have a 3D environment with an Avatar on which the patterns will be draped and later simulated as garments.

Vibbi is planned to have a representation of the design in both 2D and 3D and allow the designer to interact with both of them, see Figure A.5. The bidirectionality implies that affecting the design in 2D will change the representation in 3D and the other way around.

The interactive tools provided for the 2D interface shall, as much as possible, be usable directly on the 3D garment. In particular, the user shall be able to adjust for example sleeve length and waist width directly by dragging the corresponding hemlines of the garment. Another interesting operation is to change the volumetric of for example sleeves and chest pieces. Even so, since real world manufacturing is of interest, all 3D edits done on garments has to be translatable into 2D edits made on patterns. It shall also be clarified to the user, both in 2D and 3D, what edits are about to happen.

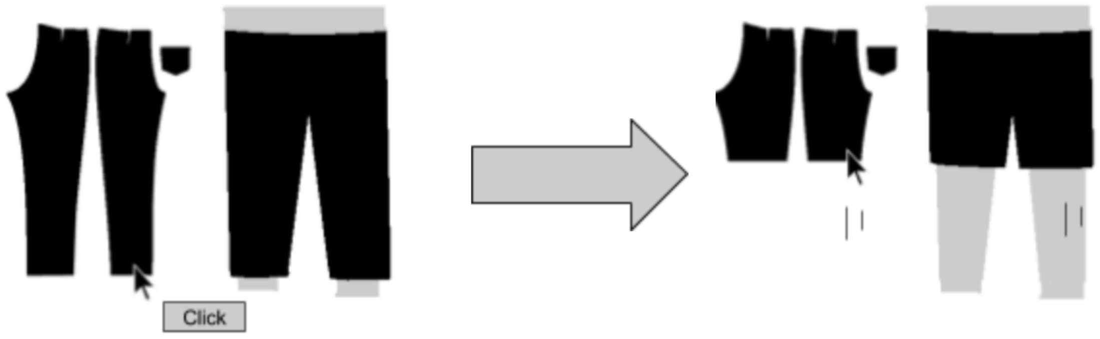

Figure A.5: Illustration of simulation while modeling. The user constantly see both the 2D pattern and the simulated result in 3D. When manipulating the 2D pattern, the 3D result will be changed accordingly and vice versa.

A.7 Phase 6: Additional implementations

Lastly, it would be interesting to let the user change the properties of the cloth as well as the avatar and visualize the result accordingly. Even more interesting would be an avatar in motion or multilayered garments.

Vivace already supports simulating pieces of cloth with different properties, such as stiffness and weight, which makes it possible to simulate different types of fabric such as silk and leather. Adding the possibility to interactively change these parameters during simulation can be of interest for the designer, especially if different pieces can represent different fabric.

Changing the properties of the avatar can be interesting since it can visualize how the garment looks like with different body measurements such as waist and chest size or arm and leg length. Of possible interest is the scaling of the garments according to the new measurements since it will allow for reuse of designs. The desire to design similar types of clothes for different sized bodies, or completely different characters, is rather common and should be possible to do by simply changing the character for which the clothes were designed originally.

An avatar in motion dressed in garments can visualize the effects of using a certain design or fabric. Simply moving the arms can be of interest since a typical issue in prototyping is to get the sleeve lengths correct. There is a lot of stretching and compression involved when raising and lowering an arm.

Designing garments with multiple layers is common but simulating them is considered nontrivial. Exploring how one would want to model cloth on top of cloth, directly in 3D, can be of interest. A typical use case is to sew a pocket onto an existing garment.

A.8 Time Plan

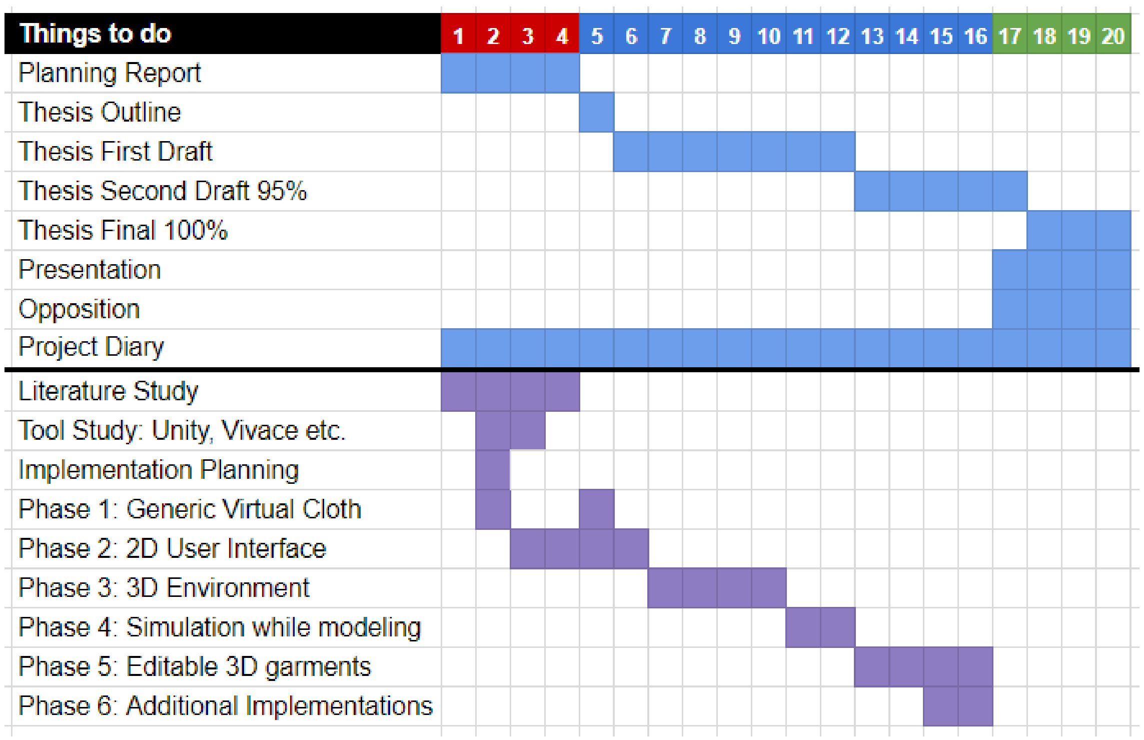

Figure A.6: Gantt chart of the time plan. The numbers representing the weeks.