

# **POLITECNICO DI TORINO**

Corso di Laurea Magistrale

in Ingegneria del Cinema e dei Mezzi di Comunicazione



Tesi di Laurea Magistrale

Cloth and hair simulation workflow analysis for an  
independent animation product.

Relatore: Prof. Antonino Riccardo Antonio Silvio

Candidato:

Nicola Figari Barberis



## Abstract

The thesis aims to illustrate the work done within Robin Studio, a company that works in the multimedia production field. In particular, the collaboration carried out with the company focuses on the serial animation project *Reverie Dawnfall*. The given task was to study and implement a workflow for cloth and hair simulations with the ultimate goal of producing a one minute teaser trailer for the series. The thesis starts by giving an introduction to the project, describing the pre-production phases, concentrating on references and summarizing the objectives and requirements given by the company. A description and explanation of the two main software tools Blender and Marvelous Designer follows with particular attention aimed at the functionalities that they offer for simulating cloth and hair. Subsequently, the proposed and implemented solutions for tackling the simulation problem are described and compared. Among these, the thesis focus primarily on fabric simulations carried out with the Marvelous Designer software and on the design and development of the Hair Curves plugin for Blender. The plugin automates the steps that are required to breathe life into the hair of the virtual characters and can be used for future work on the project allowing for time to be saved. Finally, the completed product is looked at and comments are made as to where improvements are still possible or whether the work is satisfactory.

<b>ABSTRACT.....</b>	<b>3</b>
<b>INDEX.....</b>	<b>4</b>
<b>CHAPTER 1: INTRODUCTION.....</b>	<b>7</b>
1.1 REVERIE DAWNFALL	7
1.2 ORGANIZATION AND INITIAL DEVELOPMENT	8
1.2.1 <i>Pre-production</i>	8
1.2.2 <i>References</i>	9
1.2.3 <i>Production team</i>	13
1.3 GOALS AND OBJECTIVES	14
<b>CHAPTER 2: SOFTWARE AND TOOLS.....</b>	<b>15</b>
2.1 BLENDER	15
2.2 BLENDER 2.8	16
2.3 BLENDER API	17
2.3.1 <i>The BMesh module</i>	17
2.3.2 <i>The BVHTree Submodule</i>	20
2.4 MARVELOUS DESIGNER	22
2.5 PRODUCTION PIPELINE	23
PRE-PRODUCTION	24
ASSETS MODELING	24
MOTION CAPTURE	24
RETARGETING & ANIMATION CLEAN UP	24
SIMULATIONS	24

SHADING & TEXTURING	24
RIGGING	24
LIGHTING	24
FACIAL PERFORMANCE CAPTURE	24
RENDERING	24
POST-PRODUCTION	24
<i>2.5.1 File management</i>	25
<b>CHAPTER 3: CLOTH SIMULATIONS .....</b>	<b>26</b>
3.1 HISTORY OF TEXTILES IN CG	26
3.2 BLENDER SOFT BODIES & CLOTH	28
<i>3.2.1 Soft Bodies</i>	28
<i>3.2.2 Blender Cloth</i>	28
3.3 BLENDER CLOTH ADDON	33
<i>3.3.1 Modeling Cloth</i>	33
<i>3.3.2 Cloth Weaver</i>	36
3.4 MARVELOUS DESIGNER'S SOFTWARE	37
<i>3.4.1 Marvelous UI</i>	37
3.5 REVERIE WORKFLOW	39
<i>3.5.1 Pre-production design and references</i>	39
<i>3.5.2 Designing the Virtual Garments</i>	42
<i>3.5.3 Blender test simulations</i>	50
<i>3.5.4 Marvelous Designer simulations</i>	51
<b>CHAPTER 4: HAIR SIMULATIONS .....</b>	<b>54</b>
4.1 HAIR AND FUR SIMULATIONS IN HISTORY	55
4.2 HAIR SCULPTING AND SIMULATION TOOLS	56
<i>4.2.1 Hair and fur particle systems</i>	56
<i>4.2.2 Modeling hair with curves</i>	57
<i>4.2.3 Animating hair, the Spring example</i>	59
4.3 HAIR CURVES PLUGIN	63
<i>4.3.1 Hair curves structure</i>	63

<b>CHAPTER 5: CONCLUSION.....</b>	<b>76</b>
5.1 FINAL PRODUCT	77
<b>BIBLIOGRAPHY.....</b>	<b>80</b>

# CHAPTER 1

## INTRODUCTION

### **1.1 Reverie Dawnfall**

Reverie Dawnfall is a CG animated series set in a dystopian future in which humans are on the verge of self-destruction due to a period of severe worsening of the weather conditions and due to an energy crisis that gave rise to a worldwide conflict.

We follow the story of Nadya, a girl that, like the majority of the other inhabitants of Dome City, has some form of handicap which she copes with by improving herself with biomechanical technologies. Thanks to a device that injects a healing serum into her brain, Nadya keeps her debilitating synesthesia at bay, but she doesn't know that, during some of her synesthetic events, she is able to travel between different realities.

## 1.2 Organization and initial development

This section will focus on the aspects of development and pre-production that went into the creation of the product.

The look development phase will be analyzed and some examples of reference work will be given. Comparisons will be made between these and the final product.

Then, I will describe the structure of the production team and the roles assigned to each member.

### 1.2.1 Pre-production

The project went through a long pre-production phase during which attention was paid to what would be the final look of the product. In particular, the factors that most contributed to the decisions were:

- The low-cost nature of the production.
- The desire to have a style that would resemble the drawings of a comic book.

The limited resources and budget significantly affected choices in later stages of production. Computationally light solutions were required as the content creation was done on student laptops that varied greatly in specifications.

Furthermore, the limited size of the team, made up of 5 members, led to a separation of roles that was not always well defined and to a production pipeline that often had overlaps.

It was necessary to face problems with techniques that could easily go from hand to hand and could be developed simultaneously along with the other production phases as there was not much time before the deadline for finishing the trailer.

The desire for a visual style that moved as far away from 3D as possible also influenced the choices made. Except for particular cases that will be analyzed, in recent years the animation industry has been engaged in a search for realism in

regards to the behavior of light, particles, fabrics and, more generally, of all the elements that have to do with the physical world. This push towards photorealism meant that the software tools available often did not adapt to the style desired for the product and it was necessary to develop our own techniques that could overcome these limitations.

### **1.2.2 References**

Various references were provided from which to draw inspiration. Of these, some are high budget productions, others are from smaller companies. The main models were:

- Spider-Man: Into the Spider-Verse.
- Gatta Cenerentola.
- Love, Death & Robots.
- Spring.

Produced by Columbia Pictures and Sony Pictures Animation, Spider-Man: Into the Spider-Verse is the most recent animated film from the Marvel universe starring Spiderman.

Although animated entirely in 3D, the film is characterized by a style that resembles two-dimensional comic books. This distinctive look is achieved through several means including:

- The addition of line work to accentuate the expressions and movements of the characters.
- The superimposition of balloon text, typical of the comic medium, to describe sounds or for narrative purposes.
- The use of animation at 12 frames per second (on 2's animation) for the characters in some scenes.



Fig. 1.1 Example of balloon text and linework from Spiderman: Into the Spiderverse

Gatta Cenerentola was born from MAD Entertainment, a studio located in Naples. It is a low budget production that has chosen the open source Blender software as its work tool. Similarly, Blender was chosen for the realization of Reverie Dawnfall. From the animation style of Gatta Cenerentola, it was decided to take inspiration from the way in which the characters are outlined by clear, irregular and not very precise lines. This is to recall the stroke of a hand drawing.



Fig. 1.2 Frame from Gatta Cenerentola



Fig. 1.3 Nadya's room from Reverie Dawnfall

Love, Death & Robots is a series of animated short films produced by Netflix and characterized by a different style and look for each episode.

For the Reverie project, we focused on the episode The Witness. Inspiration was drawn for the appearance of the streets of Dome City, the background of Nadya's events, and for the realistic physics of the fabrics as opposed to the stylized look of the other elements.



Fig. 1.4 The Witness

Produced by the Blender Animation studio, the last given reference was Spring. Created with Blender 2.80, the same release utilized by us, the short film has the intent of showing the public all the latest and greatest that the software can deliver.

The Blender Animation Studio in fact, distributes a new open project during each major release of the software to showcase all the new additions to their product. Although Spring's look is very different from the other provided references, the reason why it was considered is because of the Blender Cloud. The Blender Cloud is a payed web service that allows users to access tutorials and data from the Blender Animation Studios open projects. Because Blender Cloud was already available to the company, it was possible for the team to access tutorials and behind the scenes of how Spring, technically the best that can be done with Blender, was made. The Blender Cloud has proven to be a good source of information and some production choices in the end were inspired by videos from it.



Fig. 1.5 Spring

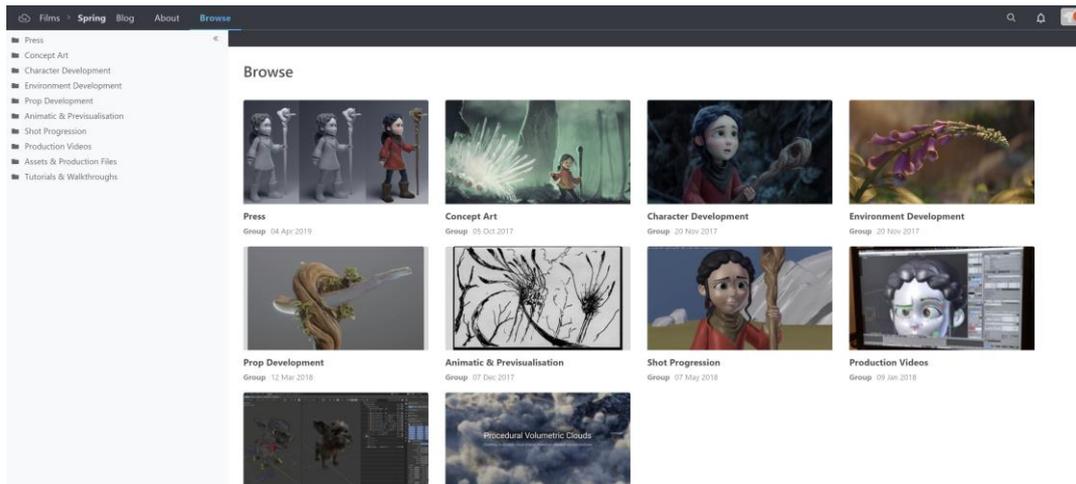


Fig. 1.6 The Blender Cloud. Tutorial and material from the Spring open project

### 1.2.3 Production team

The team was divided as follows:

- Modeler.
- Rigger.
- Lighting Technical Director.
- Texture / Material Artist and TD Supervisor.
- Cloth and Hair TD.

As mentioned above, despite the division of jobs, due to the limited number of people, it was necessary to deal with tasks that were outside of the specific role. An example is the animation which was managed equally by all team members.

The team was given creative freedom within some limits that were placed during the pre-production phase. Periodic meetings have been organized with the company supervisor to monitor progress and make changes where necessary.

### **1.3 Goals and objectives**

The goal of the project was to produce a 1 minute teaser trailer of the series. This teaser would then be used to look for further funding to devote to production of an entire season.

Given the storyboard and reference material, production choices had to be taken to optimize cost, time and reusability.

In order to do this, some of the measures that were taken are:

- The use of Blender, an open source 3d software, as the main work tool.
- A flexible and non-linear pipeline was developed in order to allow multiple steps to be worked upon at the same time.
- The new solutions that were proposed had to be packaged in the form of plugins or snippets of code to allow for further use.

## CHAPTER 2

# SOFTWARE AND TOOLS

This chapter will go over the tools and software utilized during production. Blender will be analyzed first. Particular attention will be given to its API and to the plugin development tools. Secondly, Marvelous Designer, a cloth simulation and garment design software will be looked at. Comparisons will be made between Marvelous Designer's cloth simulations and Blender's.

### **2.1 Blender**

This 3D, cross platform, open source computer graphics software was first released to the public in 1995. It was developed by the Dutch animation studio NeoGeo.

Although lacking in many aspects compared to industry standard software like Maya, Houdini and Cinema4D, over the years Blender was able to garner attention from a large online community because of its free open source nature.

Many of its shortcomings have been solved thanks to plugins and addons developed by users around the world. Some of these have been integrated in the main release and, with time, Blender has become a solid option for low budget productions.

## 2.2 Blender 2.8

Although Blender has solidified itself as the main 3D creation tool outside of medium and big studios, up to the 2.79 release it has been plagued by a user interface that has been criticized for being unintuitive and lagging behind other options.

This changed in summer 2019 when the Blender Foundation released the long awaited 2.80 update.

2.80 brought a slew of new features that drastically improved the interface and general usability.

Some of the new added components include:

- Revamped user interface.
- New dependency graph.
- Eevee rendering engine.
- New scene organization methods.

The innovations brought by Blender 2.8 together with it being free made it the best option for the production of the teaser trailer.



Fig. 2.1 UI overhaul between blender 2.79 and 2.8

## 2.3 Blender API

Blender uses Python for its development and scripting API and is based on python 3.

When started, the software automatically loads its integrated python interpreter that is used to draw the user interface and also in some internal processes.

The Blender Foundation provides access to all of its software API offering users the power to create ad hoc tools for productions. A number of scripts and addons have been written during the making of the trailer, in particular, I developed the Hair Curves plugin.

Through Python it's possible to access all of blender data, meaning that it's possible to access functions that are not directly available through the UI. One example of this is the possibility of editing parameters from linked files.

The Python modules that have been used for the Hair Curves plugin are:

- bpy, the module used by Blender for accessing data, classes, and functions
- bmesh, used to access mesh and geometry data
- mathutils, for math operations, in particular the BVHTree submodule was used

### 2.3.1 The BMesh module

BMesh is a non-manifold boundary representation that represents geometry with the following structure:

- Verts:  
Contain coordinates and link to an edge in the vertex disk cycle.
- Edges:  
Connection between two vertices. Link to loop in radial cycle of the edge.
- Loops:  
They define the boundaries of a face. Each loop logically corresponds to an edge, but an edge can correspond to more than one loop.

- Faces:

Faces link to a loop in the loop cycle, the circular linked list of loops defining the boundary of the face.

The BMesh structure is based on connectivity cycles around structure elements. To achieve this, the BMesh structure is characterized by:

- Persistent adjacent information.
- Locally modifiable geometry.
- Arbitrary length faces.
- Represents non manifold information trivially.

The connectivity cycles can be of three kinds: Disk Cycles, Radial Cycles and Loop Cycles.

### ***Disk cycles***

Disk cycles represent the connectivity of edges around a vertex. Each vertex links to an edge which contains two disk link structures. These structures point to the other disk link structures of the edges going around the vertex.

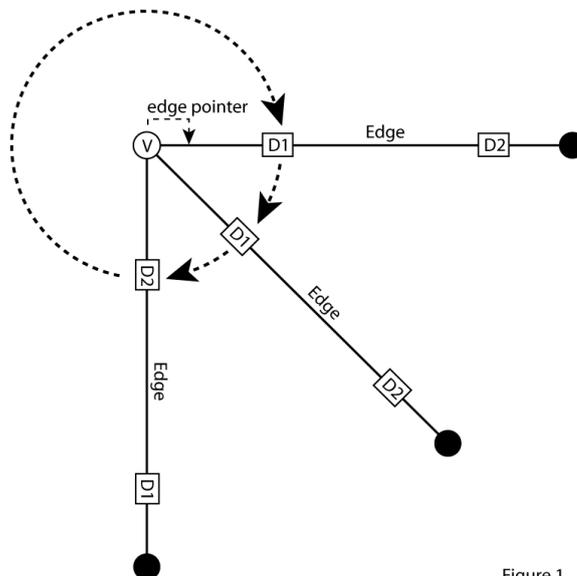


Figure 1.

Fig. 2.2 Disk Cycle. D1 and D2 represent the disk link structures of an edge.

Each vertex has a pointer to an edge in its Disk Cycle.

## Loop Cycles

Loop Cycles represent the edges around a face. Each face is described by 2 structures. The first contains a pointer to the first loop of the loop cycle that goes around it. The second is the loop structure that points to a vertex, an edge with which it corresponds and the face to which it belongs.

```
typedef struct BMSFace {
    BMHeader head;
    struct BMFlagLayer *oflags; /* an array of flags, mostly used by the operator stack */

    int len; /*includes all boundary Loops*/

    BMLoop *l_first;

    float no[3];
    short mat_nr;
} BMSFace;

typedef struct BMLoop {
    BMHeader head;
    /* notice no fLags Layer */

    struct BMVert *v;
    struct BMSEdge *e; /* edge, using verts (v, next->v) */
    struct BMSFace *f;
    /* --- snip --- */
} BMLoop;
```

Structures representing a face

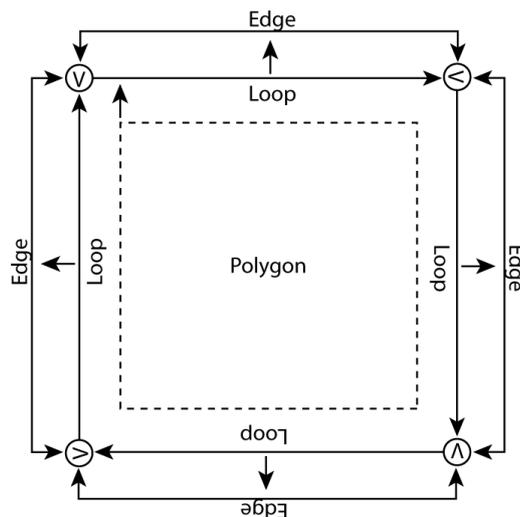


Fig. 2.3 Loop Cycle around a Face

## **Radial Cycle**

A Radial Cycle is a loop of faces connected to an edge. Each edge stores a pointer to a radial loop structure that points to the face to which it belongs and to the next loop structure in the edge's Radial Cycle.

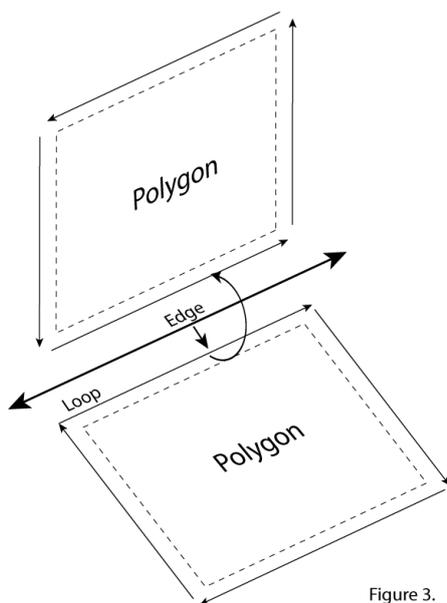


Figure 3.

Fig. 2.4 Radial Cycle

The BMesh module contains low-level API that allow to travers the geometry and make local adjustments, mid-level API that provide iterators and higher level functions/top-layer API that contain editing tools and operators.

### **2.3.2 The BVHTree Submodule**

BVH trees are structures used for ray casting and proximity detections. They are based on primitive subdivisions, which are wrapped into bounding volumes, that are partitioned into a hierarchy of sets enclosed in larger bounds. Recursively, these are again enclosed into bigger bounds to create a tree like structure with a single bounding volume.

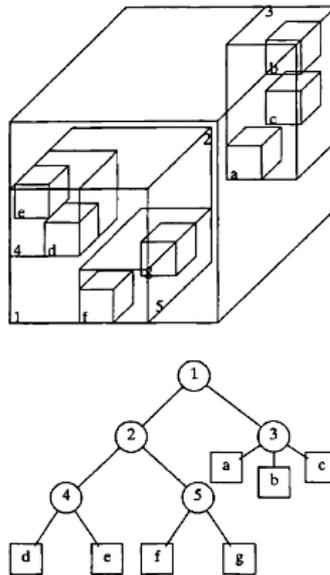


Fig. 2.5 Representation of BVH tree structure

BVH trees allow for quicker collision detections as it is possible to exclude the analysis between candidates whose parents do not collide.

The Blender mathutils module contains utilities that were useful during the Hair Curves plugin development. Some of these were:

```
class mathutils.bvhtree.BVHTree
    classmethod From BMesh(bmesh, epsilon = 0.0)
```

Creates a BVH tree based on the BMesh data. Epsilon is the value of the threshold for detecting ray hits and overlaps

```
overlap(other_tree)
```

Finds overlapping indices between two BVH trees.

## 2.4 Marvelous Designer

Marvelous Designer is a design and simulation 3D virtual clothing software. It allows users to replicate almost any type of clothing, material and textures.

Unlike Blender, Marvelous Designer isn't a free software, but has a price of 50\$ a month. Because of the low budget nature of the production, the decision to rely on a pay to use software was taken after careful evaluation.

The main reason for the choice was that Marvelous offered simulations that are much more stable and realistic compared to Blender's. Being that one of the main references for the teaser was Love, Death & Robots episode 3: The Witness which is characterized by hyper realistic cloth simulations in an otherwise very stylized environment, the possibility to replicate this look very easily was appealing and useful.

Another reason why the software was adopted is the easy integration with Blender. It's possible to import animations from Blender, simulate clothing and export the result as an animated obj.

Finally, Marvelous Designer is a popular software with an active online community that offers many tutorials, trailers and advice.

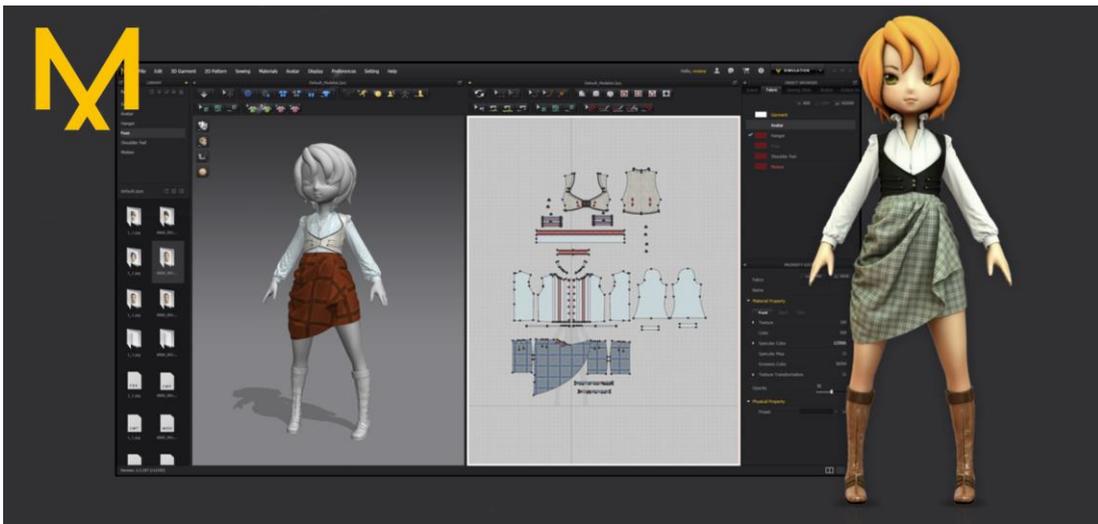


Fig. 2.6 Marvelous Designer

## 2.5 Production Pipeline

After going over the software choices that were made, it's important to look at the pipeline that was implemented for production.

The pipeline that was designed, offers versatility and has the advantage of not being strictly sequential.

These are the steps that were required for production:

- Assets modeling and rigging.
- Performance capture for main character.
- Performance retargeting.
- Cloth simulation.
- Texturing.
- Hair simulation.
- Lighting.
- Postproduction.

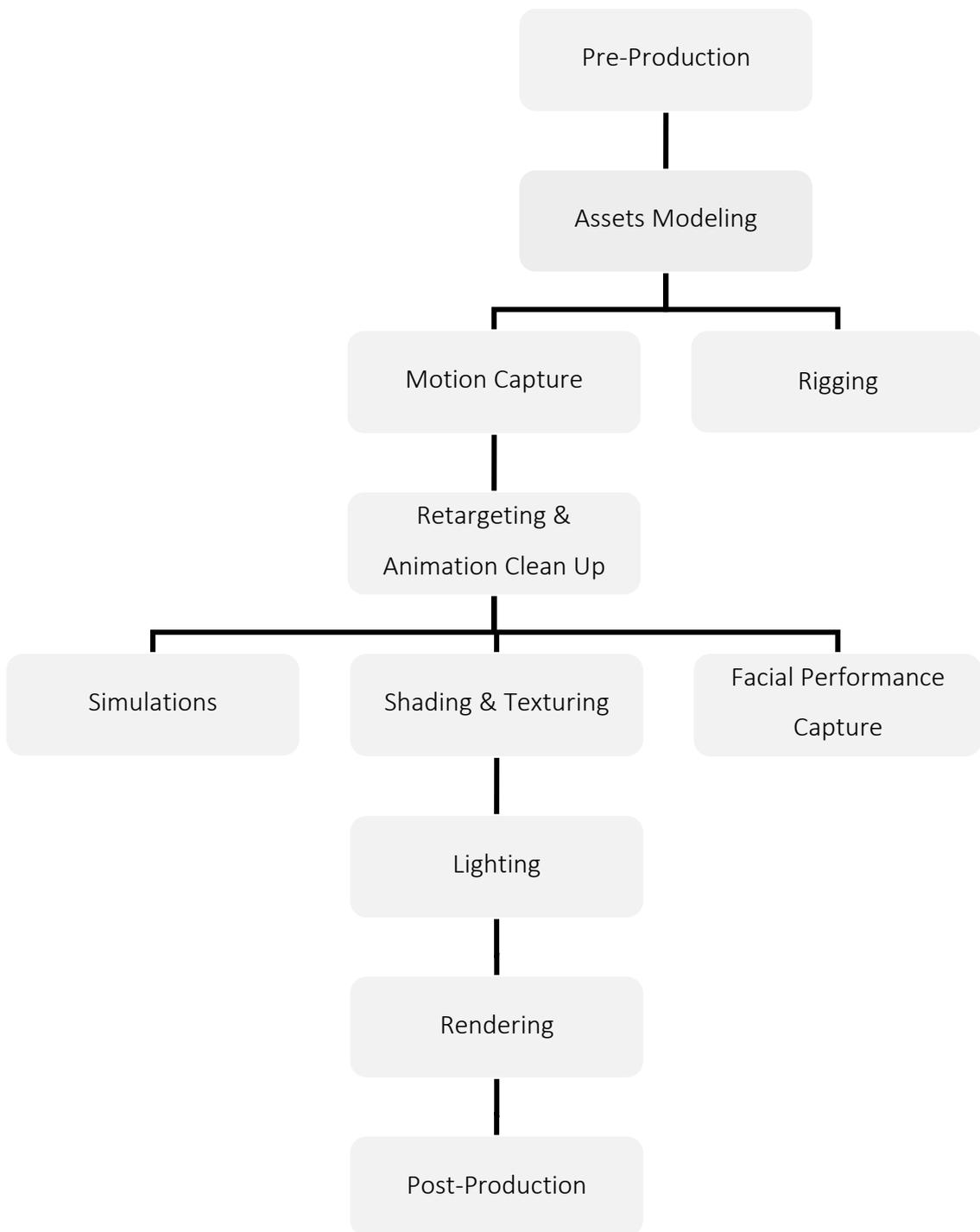
The models for the main characters were already provided but the models for the backgrounds were still being worked on.

This allowed for rigging and performance capture to begin at the same time. Both of these were divided into body and face. Animation and rigging of the body was tackled first as the facial expressions could be added later.

The Rokoko suit was used for motion capture and the Rokoko IOS app for facial capture.

After retargeting the performance, cloth simulations were tackled. Animations weren't final yet, but they were at a stage that was sufficient to see how the garments designs would react to movement.

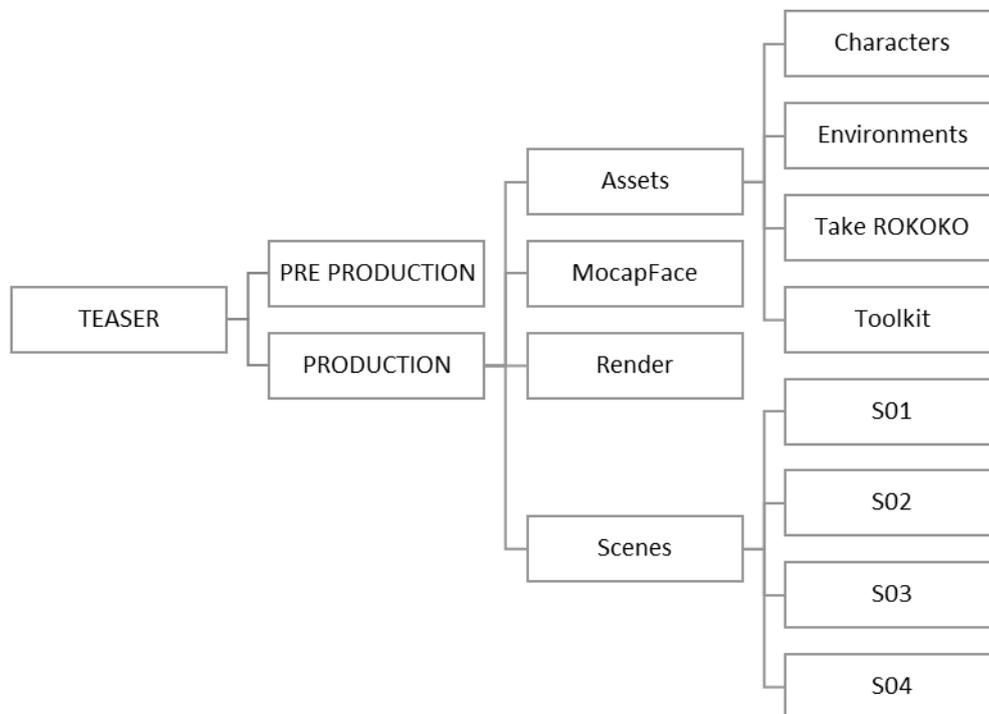
Hair rigging was finalized as soon as animation was, simulations could be added in when the product was almost finalized. Lighting was the last step of the process.



## 2.5.1 File management

Care was given to file management and directory structure. What allowed the parallelization of the different steps in the pipeline was the linked file system that was implemented.

Scene files were set up and all assets were linked to them. As the linked files were edited and updated, the scene files would automatically update making production speedy. Furthermore, linked files were much lighter in weight compared to appended ones and, although they can not be edited directly through the interface from the scene file, they can easily be modified through Python code.



# CHAPTER 3

## CLOTH SIMULATIONS

This chapter will go over the first of the main working points that this thesis focalizes on, cloth simulations.

Firstly, a brief history of cloth simulations in animation will be given, from its origins to the state of the art. After, the various software offerings for simulating will be analyzed. Reasons will be given as to why Marvelous Designer was chosen to work on the teaser trailer.

Finally, the process with which the garments were realized for the teaser trailer will be illustrated.

### **3.1 History of Textiles in CG**

Throughout the history of CG, many improvements have been made to the way cloth is treated. It started with hand animating but rapidly evolved into dynamic simulations. These can be broadly subdivided into two categories: Sheet-based and Yarn-based models. Sheet-based simulations approximate textiles with elastic materials while Yarn-based ones model the way yarn is interwoven into the fabric and should result in a more realistic look but slower calculations.

Here are reported some of the important steps in the way textile material was approached in history.

Flag and Waves (1986) is the first example of cloth simulations in animation history.

The animation was screened at the 1986 SIGGRAPH and was developed for Pixar.

In Geri's Game (1997), Pixar's objective was to drastically improve the animation for humans and clothing. In particular, there was the desire to find a way to automatize cloth dynamics to avoid having hand animated folds and creases into the material. One of the main discoveries that were made during the development of the short film was the importance of tailoring clothes. Not only did the dynamics had to be considered, but the shape and stitching of the garments had to be realistic otherwise the result wouldn't look convincing.

Stable but Responsive Cloth (2002) is a paper published by Choi and Ko for the Seoul National University. The paper tackles the issue of fabric buckling. Up to that point research avoided the problem because of its instable and non-linear nature, but Choi and Ko were able to apply a particle based model to obtain a stable and realistic simulation.

Presented at SIGGRAPH 2007, Efficient Simulation of Inextensible Cloth is a paper that, as the name suggests, looked at the fact that textiles do not stretch under their own weight. Until then, many solvers disregarded this to improve on stability and simulation speed. Cloth was seen, with a Sheet-based approach, as an elastic material with very high elastic moduli to reduce visible stretching. The proposed method treated the inextensibility as a constraint and as a result gained speed and simplicity.

2010 saw the improvement of collision detection in Yarn-based approaches with the publication of Efficient Yarn-based Cloth with Adaptive Contact Linearization.

As for the present, Yarn-based approaches have generally been abandoned in consumer available software and most tools use a mass-spring model.

## 3.2 Blender Soft Bodies & Cloth

The first option that was considered was to use Blenders internal tools for cloth and soft body simulations. They are available through the physics tab and can be applied to objects through modifiers.

### 3.2.1 Soft Bodies

Used for the simulation of deformable objects, soft bodies can be used for cloth. Soft bodies interact with external forces and can collide both with other objects and themselves. Blenders manual suggests that this modifier is well suited to:

- Add secondary motion (jiggle) to an already animated object.
- Simulate the behavior of deformable and elastic objects like rubber.
- Flags, ropes, cushions and simple fabric objects that react with forces.

Although soft bodies and cloth are treated in a similar way in Blender, the manual suggests the use of the latter for more complex meshes as the solver used with cloth simulations is faster.

In soft bodies the vertices within a mesh are treated like single particles having mass. These particles interact with exterior forces as well as interior ones.

Exterior forces are applied to vertices or to edges. Forces like gravity are applied to vertices, while aerodynamics are applied to edges.

The interior forces are modelled with 2 elements, the first being a spring that keeps particles together at a certain distance and the second being a damping force that resists movement.

### 3.2.2 Blender Cloth

Blenders cloth modifier is a powerful tool that was explored as an option for realizing Nadya's clothing.

One of the strengths of the tool is the ability to mix the modifier together with other modifiers that Blender provides. One example of this is the possibility of adding a subdivision surface modifier to the cloth object in order to have a simulation with a small number of vertices that is smoothed by a high polycount afterwards.

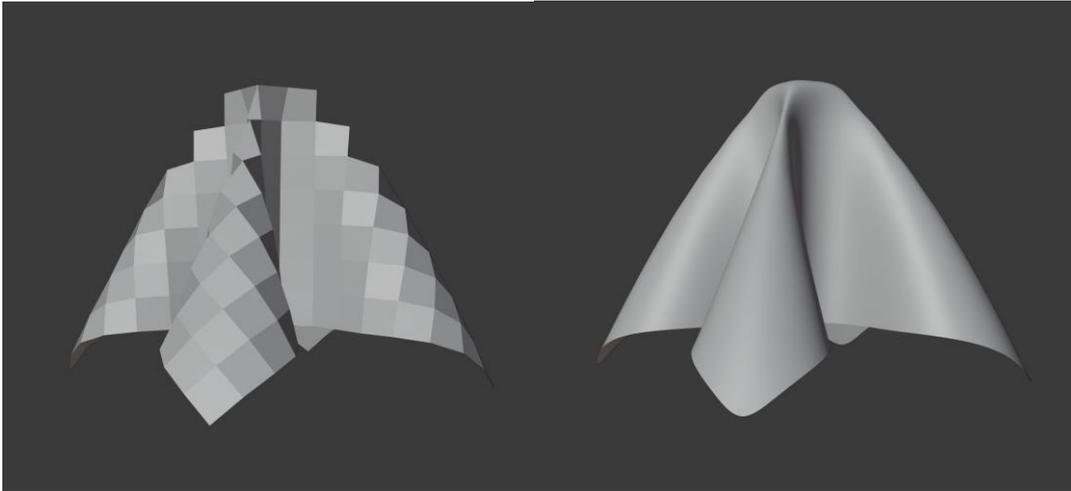


Fig. 3.1 Example of subdivision surface in cloth simulations

### ***Cloth Physics***

What are the dynamics behind the cloth simulations?

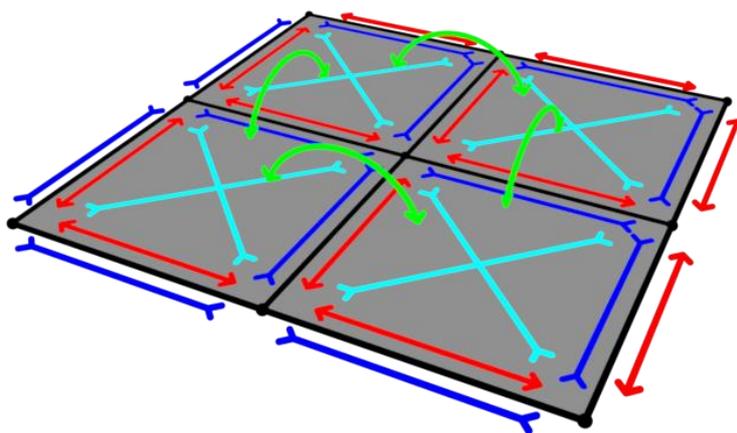


Fig. 3.2 Tension springs (blue), compression springs (red), shear springs (cyan), and angular bending springs (green).

Like with soft bodies, cloth simulations approximate the behavior of the material through the use of virtual springs that stretch and compress the object.

These springs are of four types:

- Tension springs  
Control the stiffness of the cloth.
- Compression springs  
Control the amount of force required to collapse the cloth.
- Shear springs  
Control angular deformation.
- Angular Bending Springs  
Control how susceptible the cloth is to folding and crumpling.

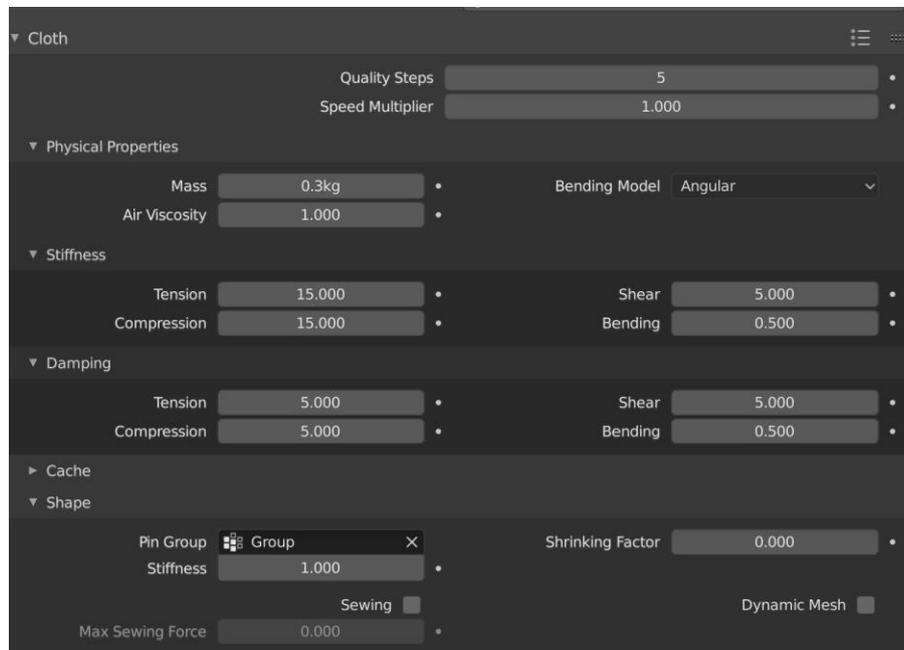
The parameters for controlling the strength of the virtual springs are found in the Blender Physics tab under Stiffness.

Along with springs, textiles have an internal damping force that calm the movement down. These forces resist motion and changes in the object, they can be controlled with the Damping parameters in the physics tab.

External forces are also considered in the cloth modifier. These forces apply to either vertices or to edges. In the case of vertices, they are calculated using Newton's Laws of Physics, in the case of edges they are calculated through the projection of the direction of the force on the edge. Vertex forces can be gravity and charge, while examples of edge forces are wind and drag.

## Blender's Parameters

Blender uses the following parameters to control the simulation properties.



Some of the parameters have already been explained in the previous paragraph. The remaining parameters follow.

- Mass: Mass of cloth material.
- Air Viscosity: Thickness of air that slows object down.
- Bending Model: Linear or Angular bending springs used in model.

The modifier allows for some Shape parameters.

- Pin Group: Pin groups are controlled through the choice of a vertex group. They allow to choose which part of the object remains still (pinned) and is not affected by the simulation.
- Stiffness: Pin group stiffness.
- Sewing: Sewing springs is a method for restraining cloth. The springs pull together two vertices of the object just like the virtual springs covered earlier, but sewing springs are external and correspond to edges that do not belong to faces.

- Shrinking Factor: Factor that shrinks or expands the cloth based on positive or negative value.
- Dynamic Mesh: Allows for the simulated mesh to be further animated with the use of shape keys or modifiers.

### ***Cloth collisions***

Blender offers external and internal cloth collisions. Collisions are calculated using primarily vertices although edges and faces can be included to improve collision detection.

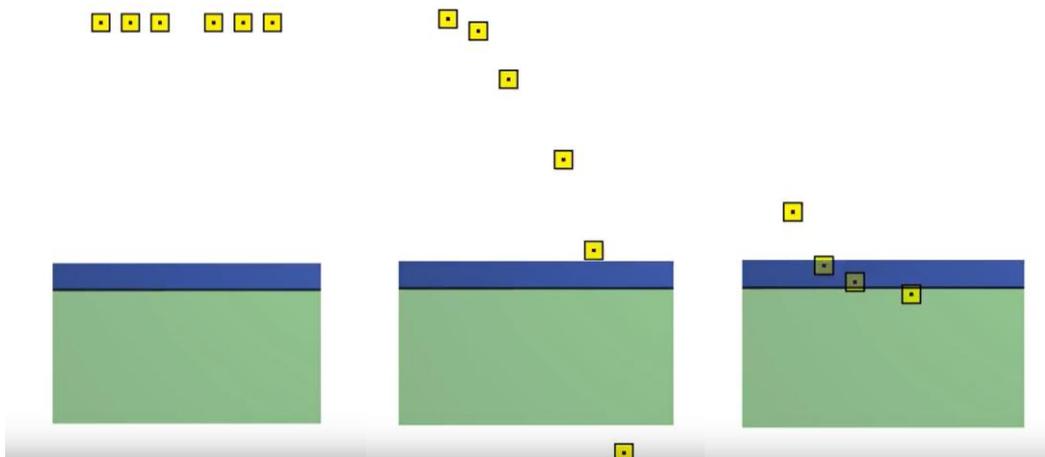
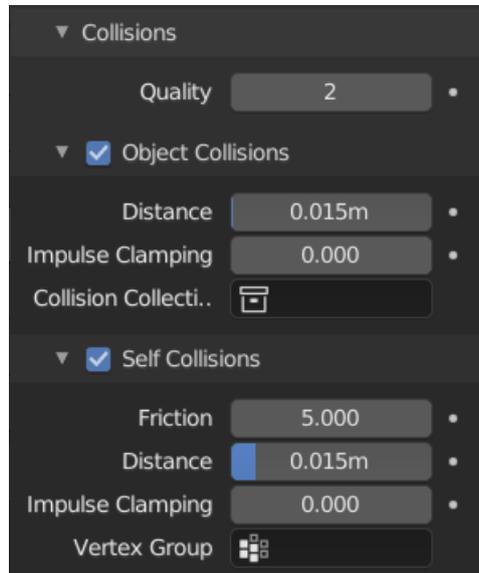


Fig. 3.3 Example of collision detection in Blender.

A collision object has 2 zones, an inner (Green) and an outer (Blue). The yellow objects start at a certain height and have different initial speeds (image 1). If the speed is too high the object will go directly through the collider (image 2), otherwise it will end up between the inner and outer zones of the collider (image 3) and collide. The repulsion forces will depend on the sum of the different forces acting on the objects.

Collisions have different parameters in blender as illustrated:



- Quality: General parameter that influences how fine the simulation will be.
- Distance: Distance an object must get to in order to be repelled from the collider.
- Impulse clamping: Restricts the amount of movement after collision.
- Collision collection: Collection to be considered for collision detection.

Self-collisions are possible and they have the same parameters as external collisions with an added Friction value that controls how slippery the cloth is.

### 3.3 Blender Cloth addon

Because of its open nature, Blender offers a slew of addons that change the way cloth simulations and garment design is tackled. Some offerings have been considered and they are reported here.

#### 3.3.1 Modeling Cloth

Modeling Cloth is a Blender addon developed by Rich Colburn. It's entirely reliant on Python code, doesn't rely on Blender's integrated solver and takes

inspiration from the Marvelous Designer software. The final objective for the addon is to implement all of the Marvelous feature directly into Blender.

The main feature of the addon is that it allows the user to edit the cloth while the program is simulating in real time. This is not possible internally in Blender.

For its editing, Modeling Cloth uses shape keys. In particular, when activated, it creates 3 shape keys on the object:

- Base key: the original shape of the cloth when Modeling Cloth was activated.
- Modeling Cloth source key: key that allows to edit the shape of the mesh as the starting point for the simulation.
- Modeling Cloth key: simulation key.

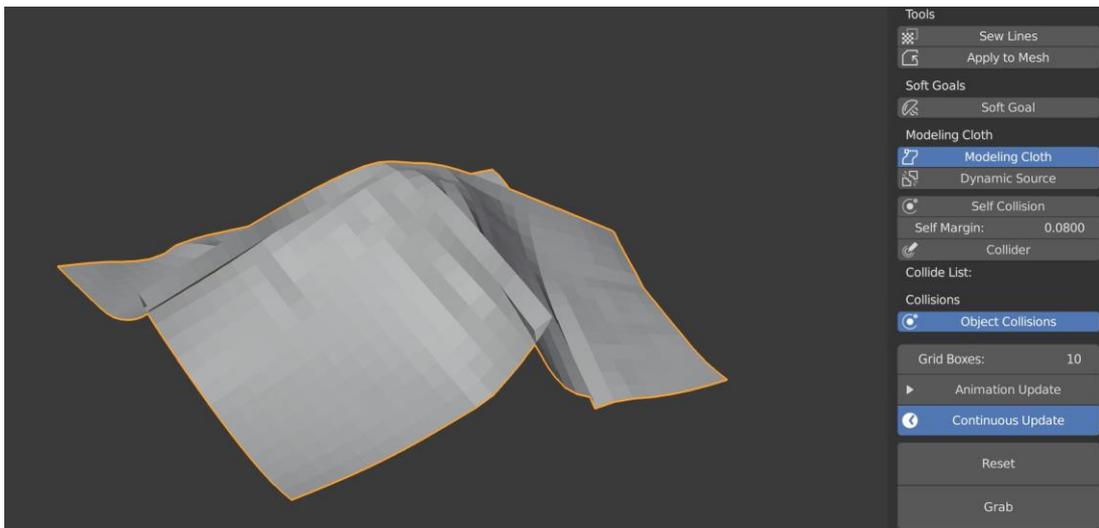


Fig. 3.4 Modeling Cloth addon continuous update

Some of the interesting features that Modeling Cloth offers are:

- Possibility to live update and interact with the cloth simulation.
- Possibility to sew pieces of garments together automatically.
- More stable and easier to set up compared to Blender's internal engine.
- Cloth pinning and wind force.

The addon has a price tag of 35 USD but, being that Blender 2.8 has recently released and that the addon has only recently been ported, it is available for free.

Modeling Cloth offers some different parameters to play with compared to the standard cloth modifiers.

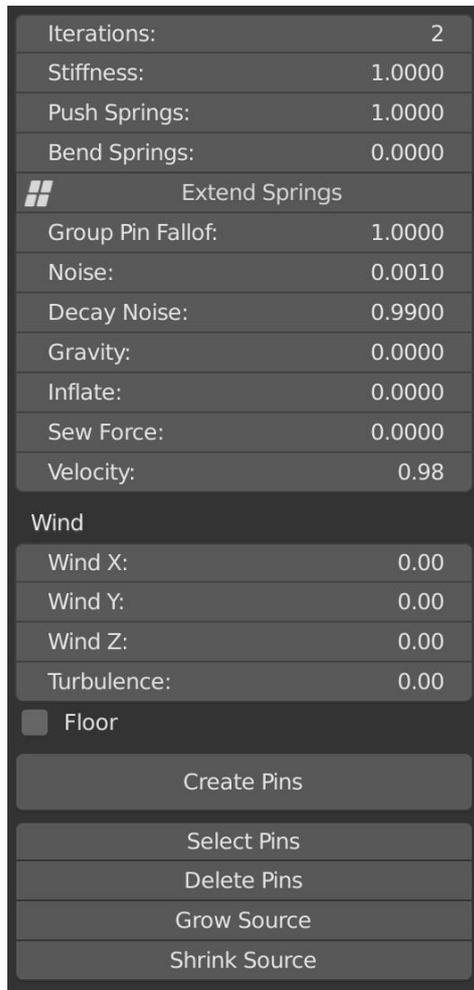


Fig. 3.5 Modeling Cloth parameters

To control the way the material behaves the Stiffness, Push and Bend Springs sliders are available.

The noise values indicate the tendency for the garment to bunch up instead of being flat.

One of the disadvantages of using the addon is the fact that the only forces with which the object can interact are gravity and the internal wind force value indicated in the addon tab.

Although modeling cloth is a very powerful and simple tool to use, because of the fact that Marvelous Designer offers more freedom in designing clothes, Modeling Cloth was left behind in the preproduction phase.

### 3.3.2 Cloth Weaver

Cloth Weaver, developed by Xanderak LLC, is another add-on that aids its users in the creation of clothes for characters. Unlike Modeling Cloth, Cloth Weaver doesn't use its own solver, but relies on Blender's internal modifier. This allows it to take advantage of all the built-in features of Blender but lacks some of the ones that are brought with Modeling Cloth.

Cloth weaver in fact is more of a design tool rather than a simulation tool.

The main offerings included in Cloth Weaver are:

- Library of templates.
- Ability to design custom clothing.
- Fabrics and accessories.
- Materials, patterns and textures

The template library offers a wide variety of choices, from tops to swimwear to accessories. These garments come in 2D patterns that are sewn together when the simulation is started. This offers the advantage of already having the UV maps necessary to apply textures afterwards.

The software also offers a selection of textures and patterns to apply to clothing. Although the software offers many options for stock pieces of clothing, there are better options in terms of simulations and there are also better options for stock garments.



Fig. 3.6 Cloth Weaver UI

### 3.4 Marvelous Designer's software

A brief introduction to Marvelous Designer was given earlier. Here I will go over some more details on how the software works and what is the workflow to follow to get from an idea to a final design.

The software is tried and tested as It has been used in multiple Hollywood movies and AAA game titles. Some of these include Ted (2012), Far Cry 5 (2014), Game cinematics for Assassin's Creed, The Witcher 2 and many other products.

The instruments offered are some of the best virtual clothing design tools available on the market.

Designing virtual clothing with Marvelous very much resembles designing in real life. It starts with 2D patterns, then moves on to fitting and finally to adding details like stitching and textures

#### 3.4.1 Marvelous UI

Marvelous designer's user interface is divided into different environments. The environments can be chosen with the drop-down panel on the top right of the UI. First the Simulation section will be looked at.

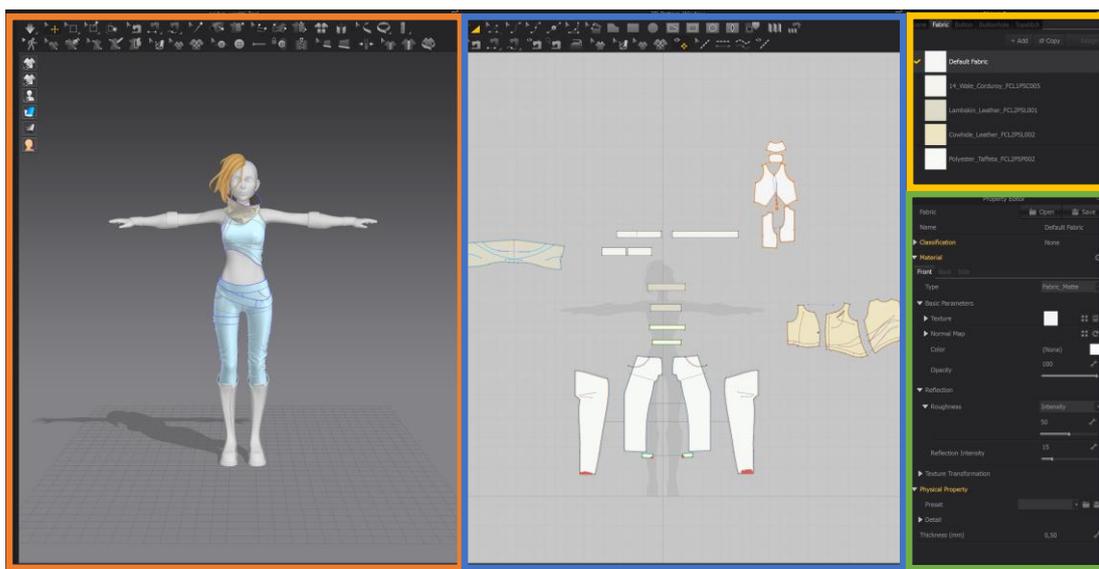


Fig. 3.7 Marvelous Designer simulation environment UI

- In orange the 3D garment window is indicated. This is where the 3D clothes and avatars will be displayed. In this window users can start the simulation and find tools to dynamically interact with the creation.
- In blue is the 2D pattern window. In the pattern window users tailor the 2D garments using drawing, sewing and edit tools.
- The object browser window is in yellow. Here the various objects, garments, materials are collected and ordered.
- Finally, in green is the property editor window. This is where the pattern and fabric property values are found and can be modified. It's also where the simulation parameters are.

The second important environment is the Animation one.



Fig. 3.8 Marvelous Designer Animation UI

- In green is the 3D viewer.
- In yellow in the object browser window and property editor.
- In orange is the animation timeline. When animation data is available It will show up here. When a simulation begins, the cache will build and be show in

the timeline. It's possible to export the baked animations in different formats.

The last environment to look at is the sculpting one. Here the sculpting tools can be found. The image of the sculpting environment is not reported as it's just a 3D view with the sculpting tools overlaid.

## **3.5 Reverie Workflow**

After looking at the various options available for tackling the cloth design and simulation problems, it's time to concentrate on what workflow was adopted for the Reverie teaser trailer. In particular, the thought processes behind the decisions made along the way will be illustrated. Nadya will be used as the primary example for this purpose.

### **3.5.1 Pre-production design and references**

During the look development phase of the project a series of images that served as reference for Nadya's clothing were collected. These showed a particular style with the following characteristics:

- Futuristic looking clothing.
- Presence of many belt, bag elements.
- Presence of a collar of some sort.
- Generally dark colors.

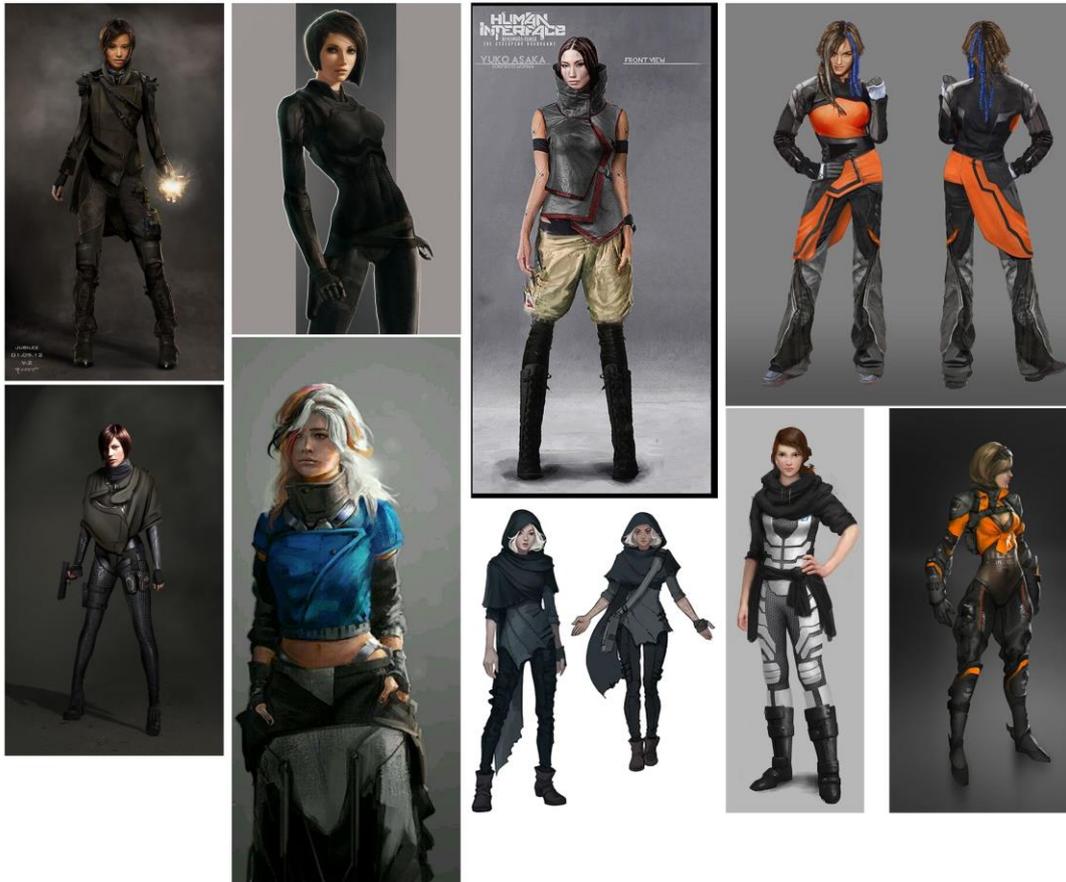


Fig. 3.9 Nadya clothing reference images

Given these references, it was decided, for simplicity purposes to avoid over complicated looks and garments, but there was still a desire to have a modern and futuristic look. Careful attention had to be made to the fact that Nadya lives in a dystopian future and that she couldn't look too sleek. It was necessary that, through her outfit, the character expressed who she was and what was her condition. Being that Nadya is the main character, it was also important that the audience could empathize with her, everyone needs to project themselves on her. Because of this it was decided to avoid looks that would be too characterizing. Finally, it was important that the character looked somewhat sexy and appealing to the eye.



Fig. 3.10 Nadya first drawing

In Fig. 3.10 the first sketches for Nadya's character are shown. It's possible to observe that the look is indeed futuristic, but not over-complicated. Although she underwent some other changes, the base combination of collar, short top and long pants always remained.

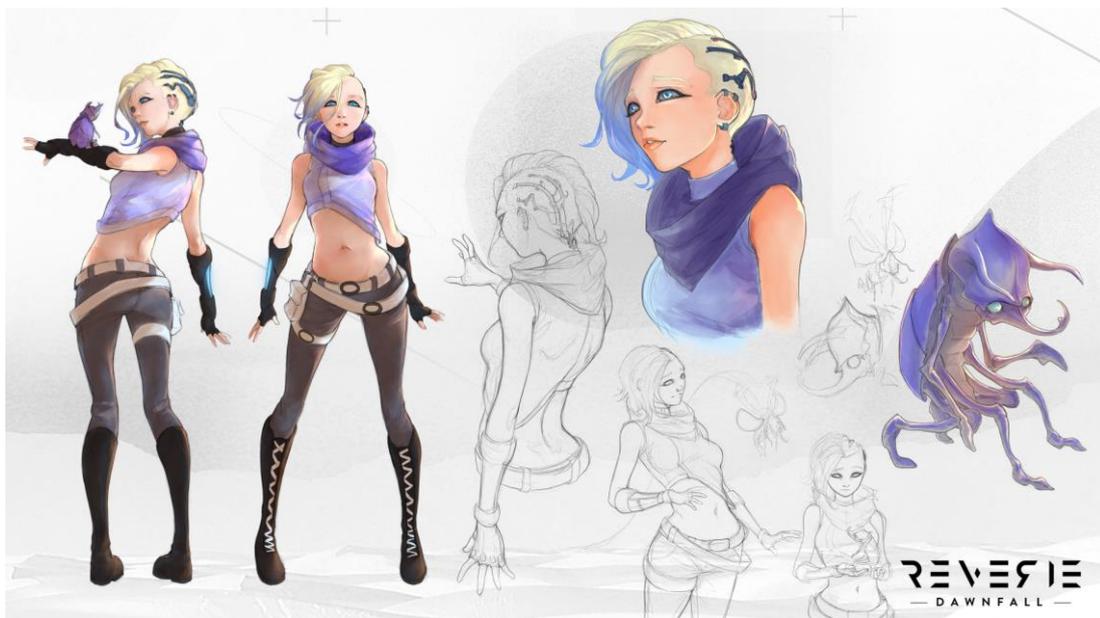


Fig. 3.11 Nadya final reference

In Fig. 3.11 Nadya's final look is presented. This was the main target to achieve when designing the virtual clothing on top of the modelled body.

The main bullet points that had to be fulfilled were:

- High boots.
- Skinny, low waisted pants.
- Presence of multiple belts.
- Short double-breasted crop top.
- Roomy and big collar scarf.
- Gloves with integrated display in one of the arms.

### 3.5.2 Designing the Virtual Garments

Once the final reference image was given, it was time to model the actual garments. It was decided that the best option to go with would be Marvelous Designer as it offered the most complete package.

The option of Cloth Weaver was also considered, but the stock garments given with the addon are modern and, although they could be readapted for Nadya, they would not fit the look of other characters and it was preferred to use one software only for the entire production.

#### ***Boots & Gloves***



Marvelous designer does have one limitation in the fact that it doesn't offer many options for creating rigid accessories. Because of this, and because it's not necessary to simulate the movement of rigid materials, Nadya's boots and gloves were directly modelled using Blender.

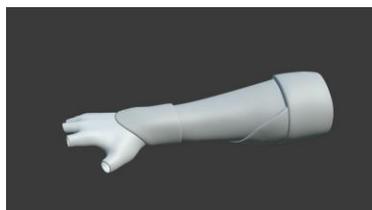


Fig. 3.12 Boots, gloves

## ***Undershirt***

Under her top, Nadya wears a tight undershirt with a high collar. The design of this piece of clothing started with finding the patterns for a blouse that would be similar to the desired one. This was done with a quick internet search.

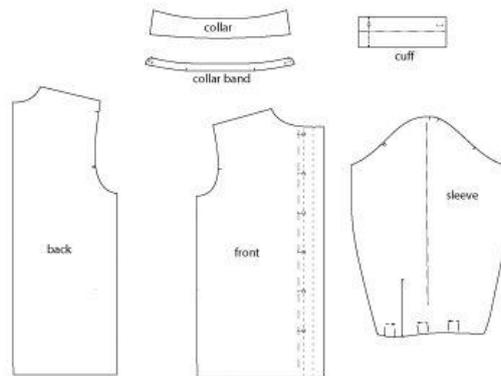


Fig. 3.13 Blouse pattern

Marvelous Designer offers the possibility to import an external image to use as background for the 2D design window.

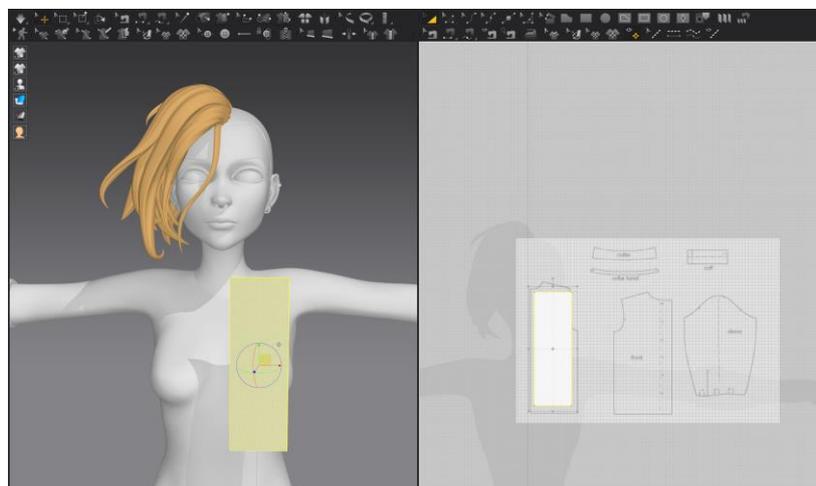
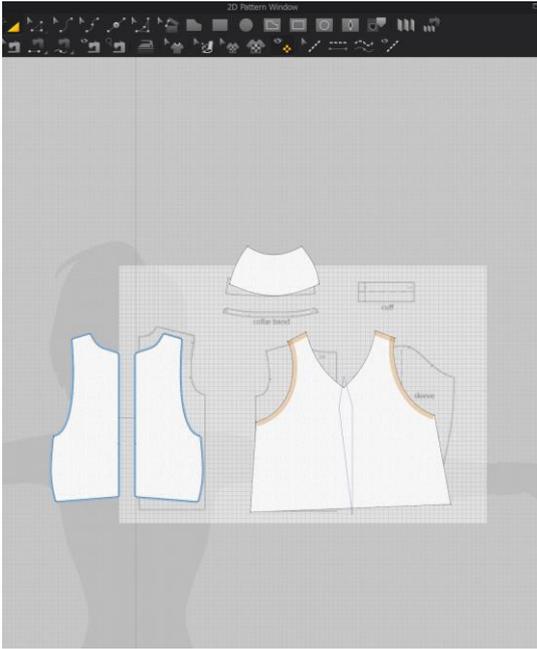


Fig. 3.14 Starting point for garment modeling

Once the pattern is imported, the next step is to create a rectangle of fabric as shown in Fig. 3.14. Using the tools on the top of the 2D editor it is possible to add and move around control points in order to shape the fabric to fit the sewing patterns.



After creating all of the necessary pieces and adapting the design to the character model, one can sew together the pieces of fabric with the sewing tool found in the 2D pattern window.

It is possible to sew together any two lines inside or on the border of fabric. Internal lines can be created and edited with the tools in the 2D pattern dropdown menu.

But how does the sewing tool work?

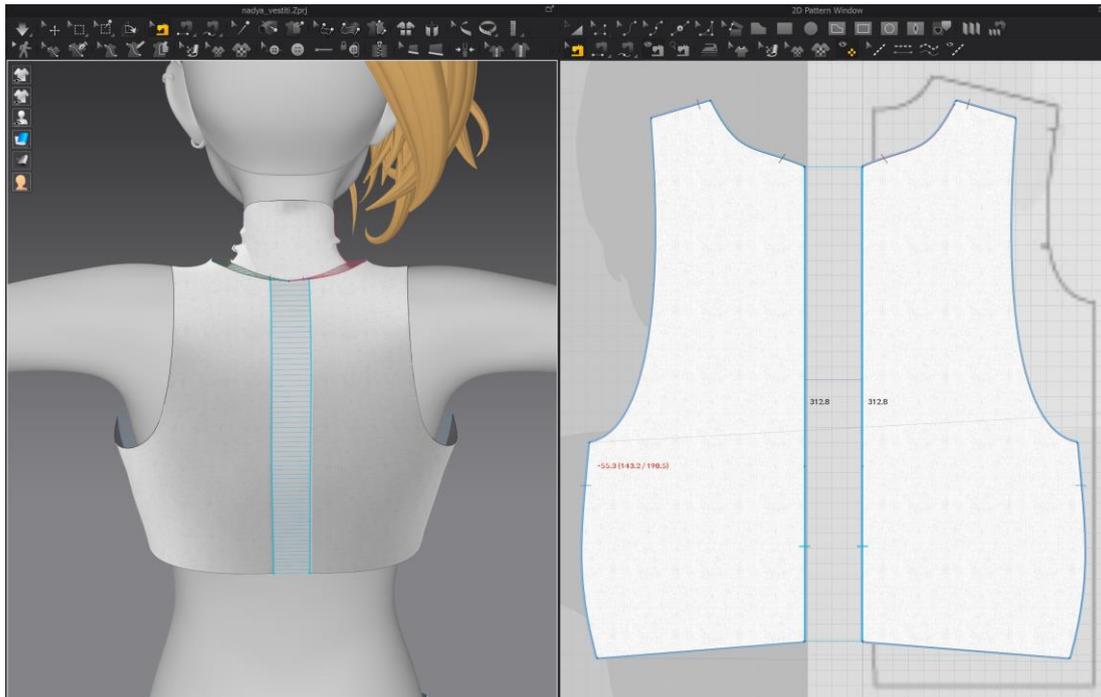


Fig. 3.15 Sewing tool

Using the sewing tool which is found both in the 2D Pattern Panel and in the 3D view panel, one can connect two lines together. The sewing will show in the 3D view so that users can understand which piece connect to which.

Sewing is directional, this means that attention needs to be paid to how two pieces are connected. It is also possible to indicate at what angle the two pieces are sewn together. In order to avoid crumpling of the fabric it is desirable for the length of the sewn lines to be of similar size.

Because the undershirt had to be tight fitting, the final adjustment was to cut out a piece of fabric to obtain a better fit. The final garment, once simulated and fitted to the avatar is shown in Fig. 3.16.

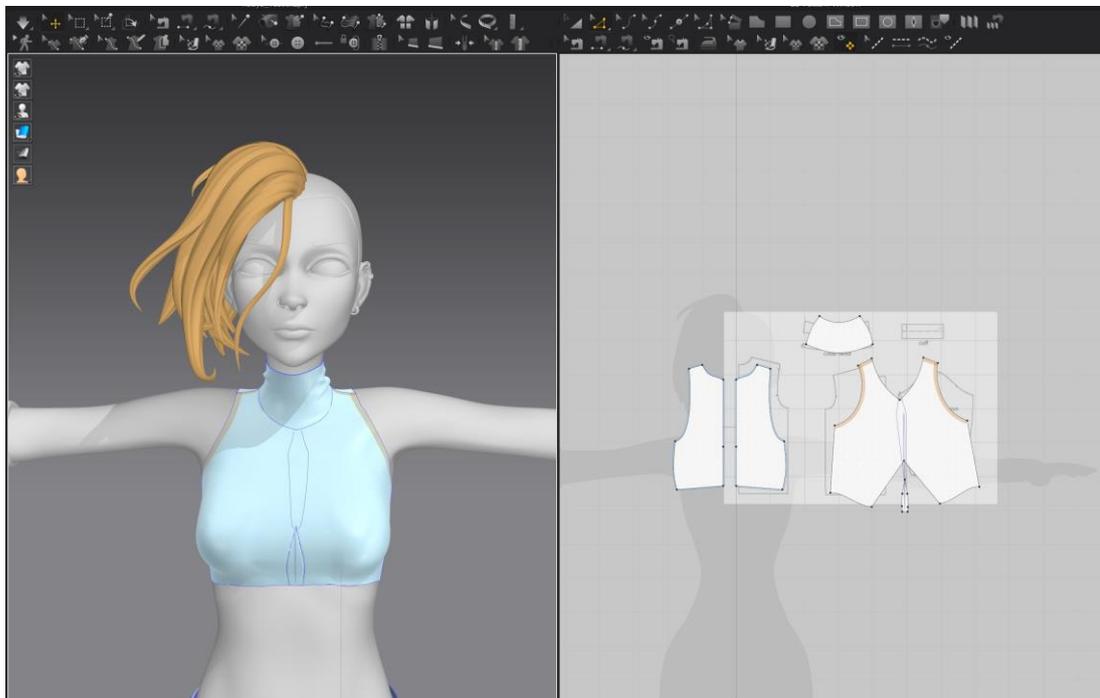


Fig. 3.16 Nadya's undershirt

### ***Pants***

In order to create the pants, instead of using the pattern first approach, a different workflow was followed. The flattening tool was used. To use flattening, one must draw directly on the avatar, using the avatar line tool, a closed loop. The flattening tool will directly create the 2D garment that fits the selected closed area. The tool is very useful in the absence of patterns to create form fitting clothing.

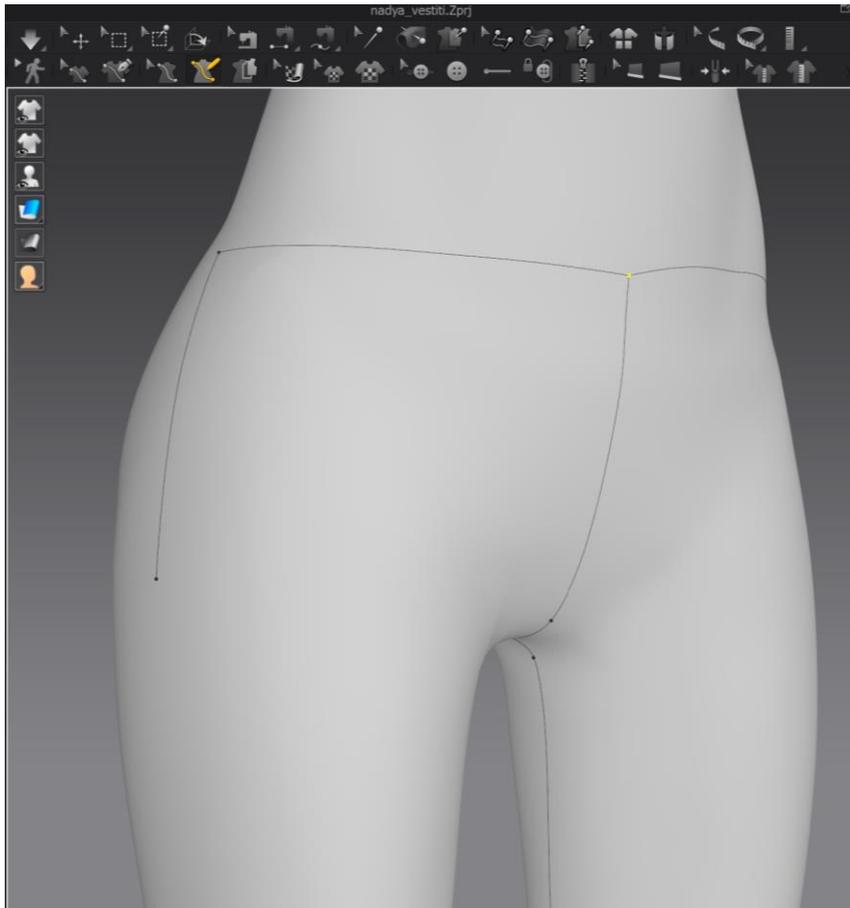


Fig. 3.17 Trace lines on avatar

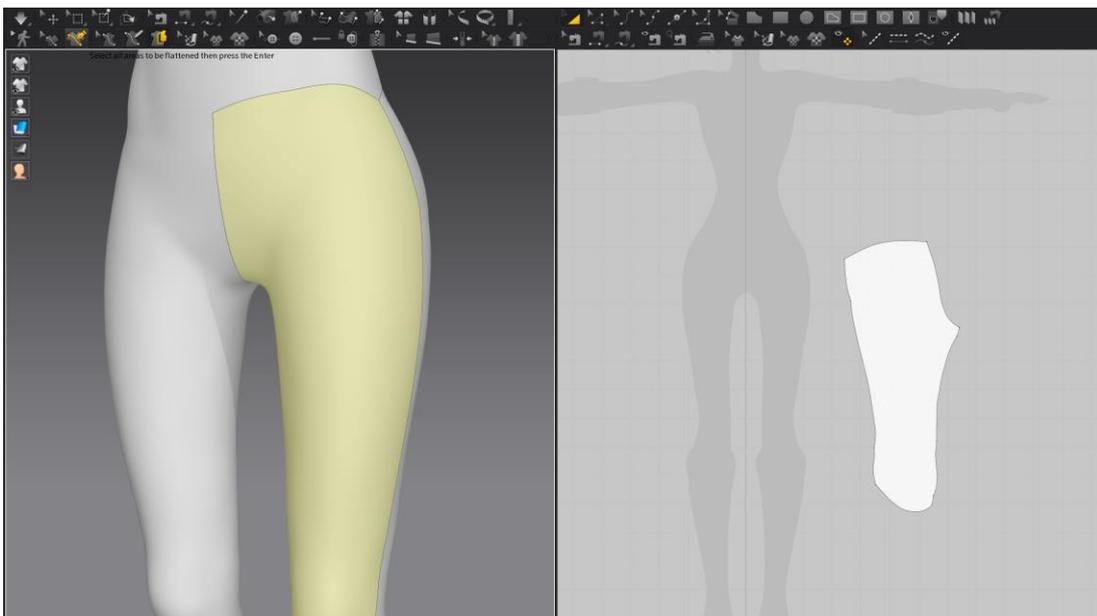


Fig. 3.18 Flattening tool result

Because Nadya's pants had to be skintight, it was decided that it wouldn't be very useful to simulate the cloth behavior. The simulations wouldn't add much in terms of visual flare and would only cost computationally.

To create a more organic look though, sculpting was done on the garment to create folds and creases on the knee area.



Fig. 3.19 Nadya's pants

### ***Accessories and collar***

Nadya's belts and collar were made with strips of fabric sewn at the ends. No patterns were used as the shapes are very basic and were adjusted while simulating. For Nadya's collar some folds were introduced in the fabric in order to create dimensionality and a better fit. Folds can be specified with an angle and a strength.

Because of the fact that Marvelous Designer is not able to introduce solid elements in garment designs except for buttons and zippers, in order to have buckles on the belts, it was necessary to model them separately in Blender.

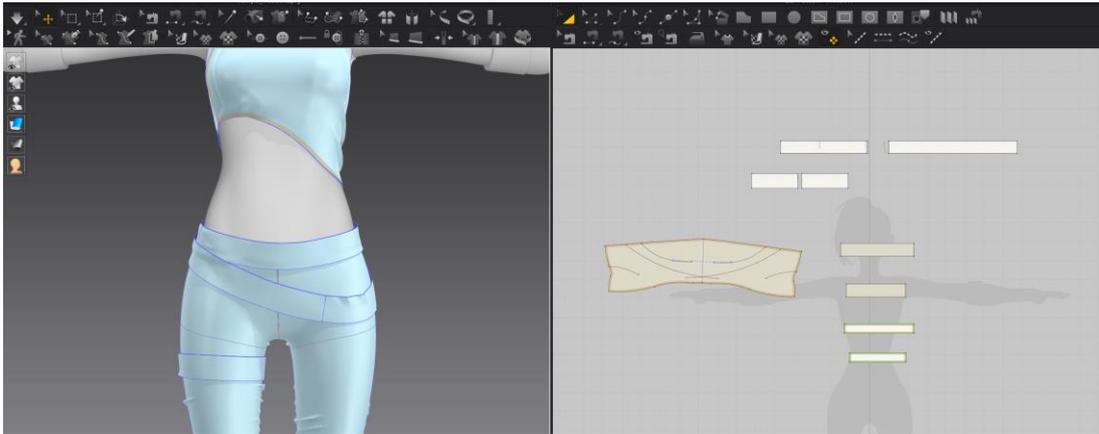


Fig. 3.20 Nadya's belts in Marvelous Designer



Fig. 3.21 Nadya's belts after modeling in Blender

## **Blouse**

The design of the shirt went through a couple of iterations. The first design was very accurate to the drawn references with a double breasted tight look. This worked well with a still character but didn't give the desired effect when simulating the garment. In fact, the tightness of the shirt didn't allow for much movement of the fabric and resulted in an animation that didn't look very organic.

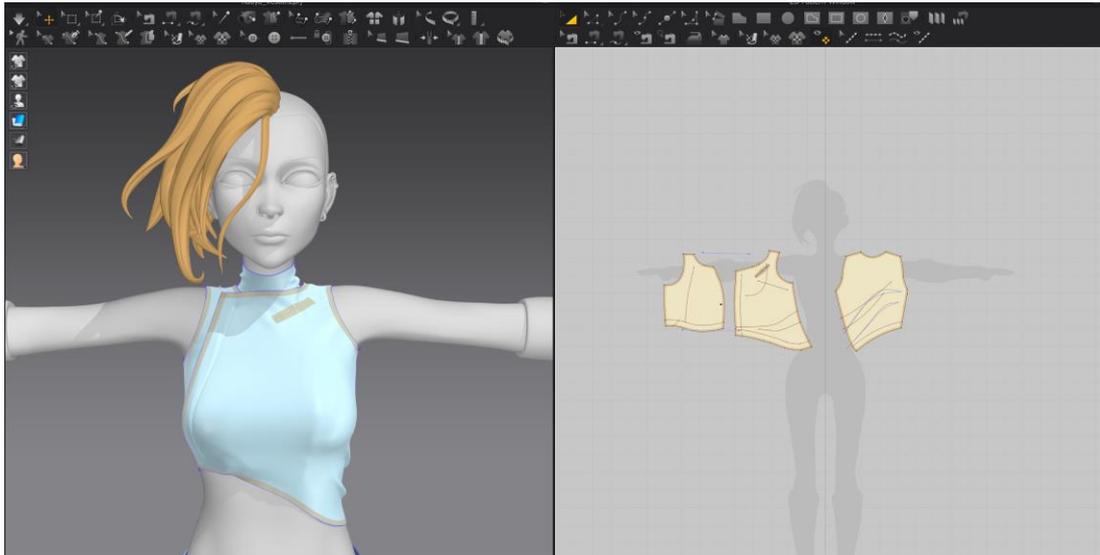


Fig. 3.22 Nadya's blouse iteration 1

In order to achieve a garment that would move and flow more, it was decided to open up the stitching that kept the two front pieces together and only stitch the top part of the shirt.

The results after simulation were very appreciated and it was decided to adopt the change as permanent.

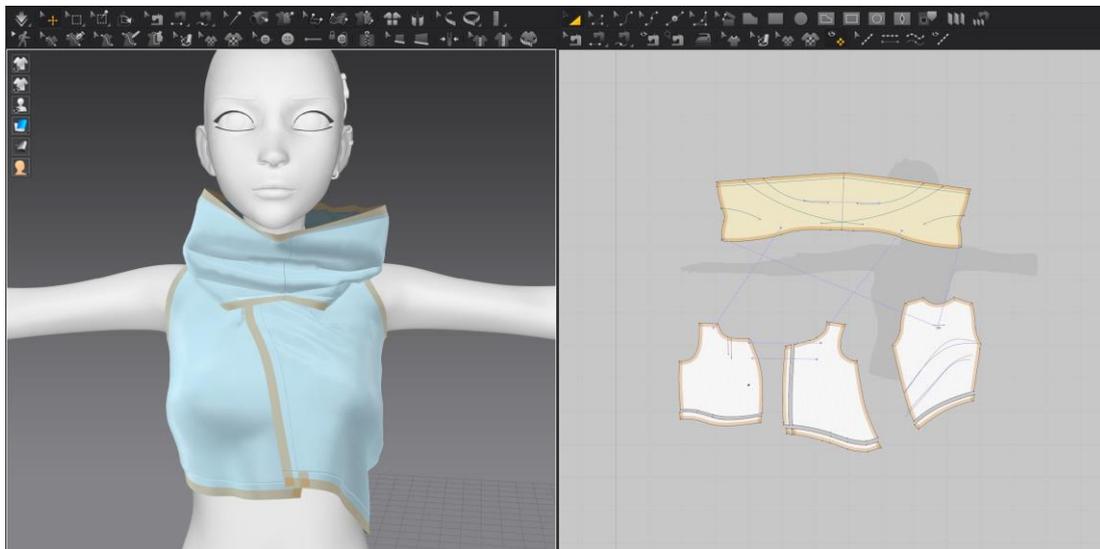


Fig. 3.23 Nadya's blouse iteration 2

### 3.5.3 Blender test simulations

After completing the design of the clothing, it was time to decide how to tackle the problem of simulating the clothing. The tailoring was done with Marvelous Designer but the possibility of using Blender’s internal cloth modifier was considered.

Inspired by Spring, produced by the Blender Animation Studio, a novel kind of workflow was tested. Instead of applying the modifier directly to the garment object, a simple cage that completely covers the clothing is modelled. The modifier is applied to the cage and the garments follow the cage through a Mesh Deform modifier.

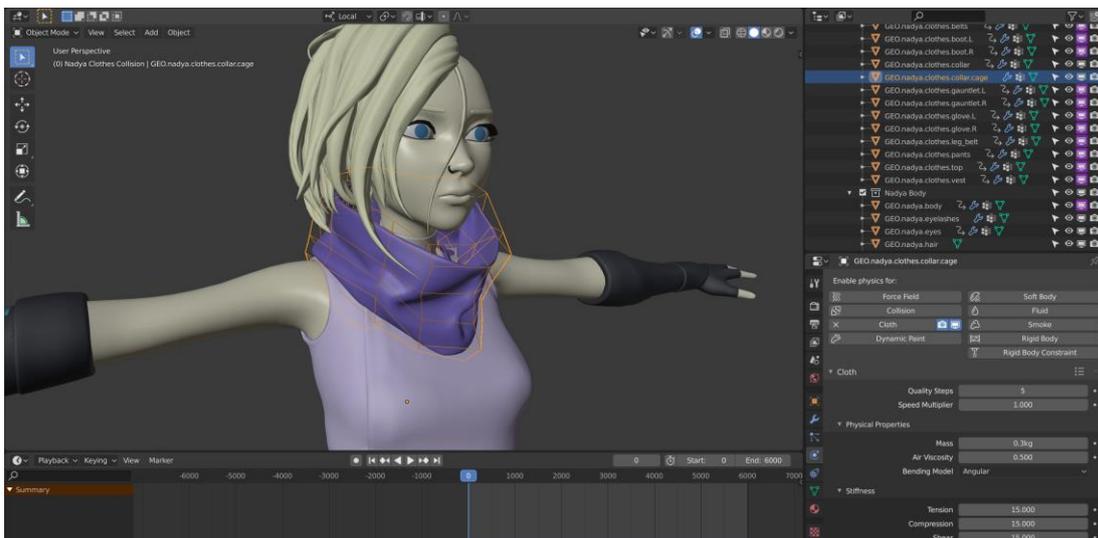


Fig. 3.24 Cloth animation test with Blender

This technique allows for object with a large number of faces and vertices to move without being simulated. The simulation is done on simpler objects to increase speed and decrease the chance of simulation errors or instabilities.

The results are satisfactory for simple garments and the movement of the clothing looks organic, but it was decided to abandon this method. The reason being that during the look development phase, Netflix’s “Love, Death & Robots” episode 3 was

given as reference. The episode features hyper realistic cloth dynamics on a stylized background and the result obtained with Blender didn't match that.

### 3.5.4 Marvelous Designer simulations

The final simulation tests, that would end up being the chosen solution, were done with Marvelous Designer.

The software offers the possibility of importing Alembic files that contain avatars with animation data. The garments inside Marvelous can be fitted to the avatars and, in the animation panel, can be simulated. Marvelous then saves the simulation to cache and it is possible to export the data in different formats.

As mentioned earlier, only the top parts of the outfit were simulated. The pants were too tight to add any significant movement and the belts had solid buckles that Marvelous can not model. Furthermore, it is important when simulating cloth, that the avatar doesn't have any self-intersections where garments are present otherwise the cloth solver fails. Being that the animation files that were given presented many body-arm intersections, simulations were done removing the arms from the avatar.

The garments were exported as OBJ files and their animation data as MDD files. The MDD files can be read by Blender through the Mesh Cache modifier which, applied to the imported OBJ garment, results in the animation of the object.

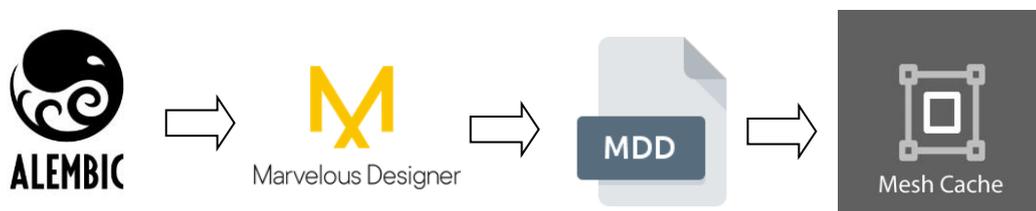


Fig. 3.25 Marvelous designer simulation workflow

Once imported and after having applied the Mesh Cache modifier, the clothing objects were finalized with the use of Solidify and Subdivision Surface modifiers. The objects exported from Marvelous are in fact 2D meshes and present some simulation errors that are smoothed out by subdividing the surface.



Fig. 3.26 Simulation in Marvelous Designer



Fig. 3.27 Results in Blender before modifiers



Fig. 3.28 Results in Blender after modifiers

## CHAPTER 4

# HAIR SIMULATIONS

Hair is one of the hardest things to realistically animate in CG, but it is also one of those elements that can add fluidity, character and an organic feel to a subject. One of the reasons for its complexity is that it reacts differently based on many factors like length, thickness, type and quantity. Furthermore, it is very difficult to hand animate because, if it's not perfect, it can immediately ruin the immersion and give a "very CG" look. Because of this, since the inception of CG animation, there has been a push to produce hair that is not hand animated but simulated through dynamics that would make the process easier and more realistic.

This chapter will go over the work that was done in simulating hair for the characters of the *Reverie Dawnfall* teaser trailer.

First, some early and important examples of hair and fur dynamics in the history of animation will be given in order to have an idea of the state of the industry and its past. Second, the tools that have been considered for tackling simulations will be analyzed and compared to the chosen one based on feedback and references from the head of production. Finally, the chapter will go over the conceptualization and coding of the Hair Curves plugin that was developed to simplify future animation and simulations.

## 4.1 Hair and fur simulations in history

One of the first examples of particle fur animation is the 1991 Fire Beast short clip. The video depicts a beast with iridescent fur in motion that was rendered with a digital differential analyzer that interpolates points on a line based on a starting and end point. These strands did not react to the environment or to movement. Just a few years later, 1994 saw the release of the Flinstones in which, for the first time in a feature film, a realistic CG fur coats one of the main characters. In 2001 Pixar released Monster Inc. For this film, the company created a new program called Fizz that would take the animated shots as input and simulate in a natural way the movement of over 3 million hairs present on the main character. Another milestone from the company was 2004's The Incredibles in which hair became a main plot point for the character of Violet that has a long and flowing hairstyle behind which she hides her face; a big departure from the short and simple ones of characters up to that point. The final film worth mentioning is Disney's 2010 Tangled. The main character of Rapunzel has hair that is more than 10 meters long and constantly interacts with the environment and other characters. The character TD had to come up with new ways of dealing with physics as Rapunzel's hair would have weighed up to 30 Kg in real life, so it was necessary to find a balance between fiction and realism.



Fig. 4.1 1992 Fire Beast vs 2010 Tangled

## 4.2 Hair sculpting and simulation tools

Before deciding how to approach hair simulations for Reverie's teaser trailer, it was important to look at the tools that were available and to analyze pros and cons for each one. In regards to the production, the available software was Blender.

### 4.2.1 Hair and fur particle systems

Blender offers its built-in hair particle system within the particles tab. The system is characterized by 4 steps:

- Growing: hair number and length is defined through interpolation of a particle's path during a lifetime of 100 frames.
- Styling: this can be done in different ways. Through the use of children that can increase the count of hair or through the particle edit mode where it is possible to comb, cut and trim the strands.
- Animation: with the use of the cloth solver hair can be made dynamic.
- Rendering: this can be done with the use of Principled Hair BSDF and Hair BSDF shaders.

There are many approaches to creating hair for a character and Blender's online community also offers a number of addons that can help make the process more streamlined but we will not go over these as it was decided to avoid particle systems in the workflow. The main reason for doing this was for stylistic and animation reasons.

### *Particle hair style*

Particle hair in Blender tends to have a realistic and natural look to it. This is in contrast with the more stylized look that was decided during pre-production. Characters in the Reverie universe have hairstyles that draw inspiration from anime and not from modern western animation. The hair shaders offered by Blender, although powerful, would not fit the desired target style.



Fig. 4.2 Example of Principled Hair BxDF shader

### ***Particle hair animation***

As mentioned earlier, Blender simulates hair using its internal cloth solver. This works well and gives the animator the ability to play with parameters and obtain different results. Another method for animating is to control hair using influence curves that guide the hair. By animating the curves, it is possible to animate the hair. The problem with these two methods is that it becomes hard to mix them. Dynamically simulate the particles while fixing unwanted results using curves is tedious. Another reason for wanting to simulate hair, but to have the freedom of also controlling it by hand is for expressiveness. A character may have some hair that behaves in a cartoony way while the rest remains natural.

### **4.2.2 Modeling hair with curves**

Another very popular way to model hair in Blender is to only use curves. The workflow is simple and allows for creating a multitude of different hairstyles. The basics of the process are as follows:

1. Creation a curve path and a curve Bezier circle. The path will serve as the hair strand, while the Bezier curve as a bevel that extrudes along the object.
2. Under the Path Settings -> Geometry Tab, select the Bezier curve as the bevel object for the path curve.
3. Control the width and tilt of the hair by scaling (Alt + S) or tilting (Ctrl + T) the points on the path curve.
4. Control the shape of the hair by editing the shape of the Bezier curve.
5. Create hairstyle with different curves and paths.

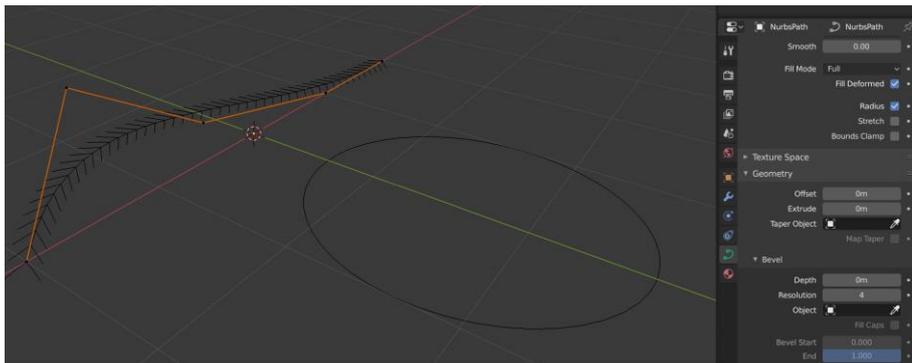


Fig. 4.3 Bezier curve and path

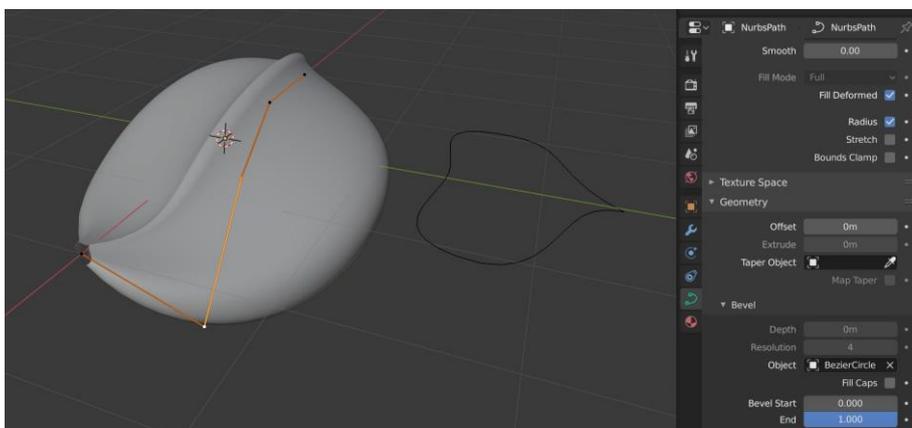


Fig. 4.4 Curve used as bevel object for path

This technique gives a lot of freedom in modeling as there is no constraints to the shape and number of hairs. Another advantage is that it is possible to have different Bezier shapes that target different paths and can add variety to a character's look. Because of these reasons it is possible to obtain a hairstyle no matter what the reference is. In particular, this method is well suited for creating hairstyles that closely resemble the ones of Anime characters and therefore fits to the style that Reverie's characters are trying to emulate. It was decided that hair would be modeled this way.

## ***Nadya's hair***

Nadya's model will be taken as an example and further analysis will be done on her. The hair of Jameela, the second character in the trailer was also modeled and animated with the same methods.

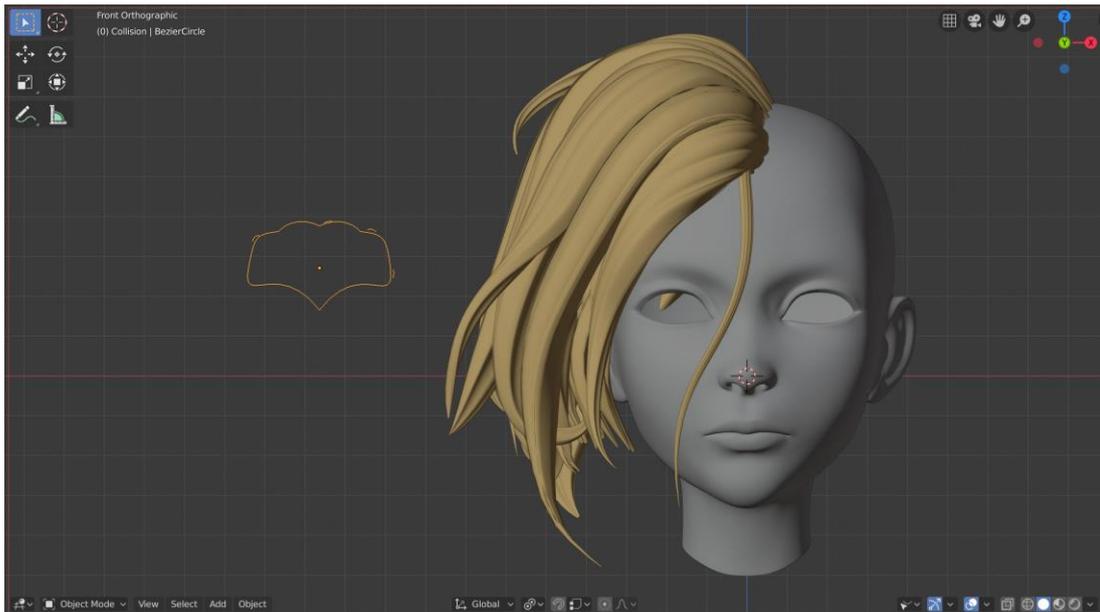


Fig. 4.5 Nadya's modeled hair using curves

### **4.2.3 Animating hair, the Spring example**

While particle hair in Blender offers the possibility of being simulated, curves can not be simulated in the same way. Although it is possible to assign a Soft Body Modifier to a curve, the simulation is not ideal and there aren't many options to control its behavior. Furthermore, the problem of simulating and manually animating at the same time remains. It was necessary to find another way of breathing life into the hair and, luckily, the Blender Cloud gave an idea on how it could be done.

Included in the material that the Blender Animation Studio offers to its Blender Cloud subscribers, is a tutorial on how the team animated the main character Spring's hair from the open movie Spring. This was helpful as Spring's hair is not simulated with particles and therefore the solution could be ported to Nadya's model.

The simulation rig starts with the hair object mesh as the base.



Fig. 4.6 Hair mesh

Then, it is necessary to create a simple sim mesh of quads that completely covers the hair mesh, a Cloth Modifier is applied to the mesh. It is also necessary to create vertex groups for each loop in the mesh and save them. These vertex groups will be used later.

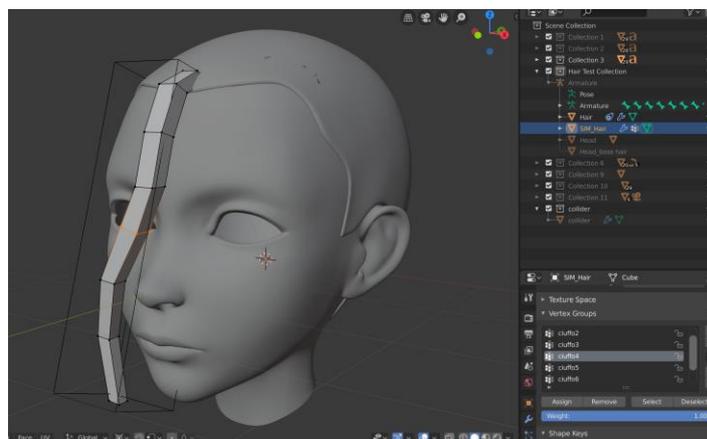


Fig. 4.7 Sim mesh around hair. Example of edge loop vertex group selected

After this, the sim mesh is rigged with an armature. It is important that each loop of quad faces corresponds to a bone and that the bone sits precisely in the middle of the loop. This can be easily done with Blender as it offers a magnetic cursor that attaches to the median point of a loop of vertices. Each bone then has to have two constraints added, an IK and a Copy Location. The IK constraint has as a target the vertex group of the sim mesh that is directly after the bone. The Copy Location has the vertex group just before. Doing this allows the bone to follow the sim mesh around.

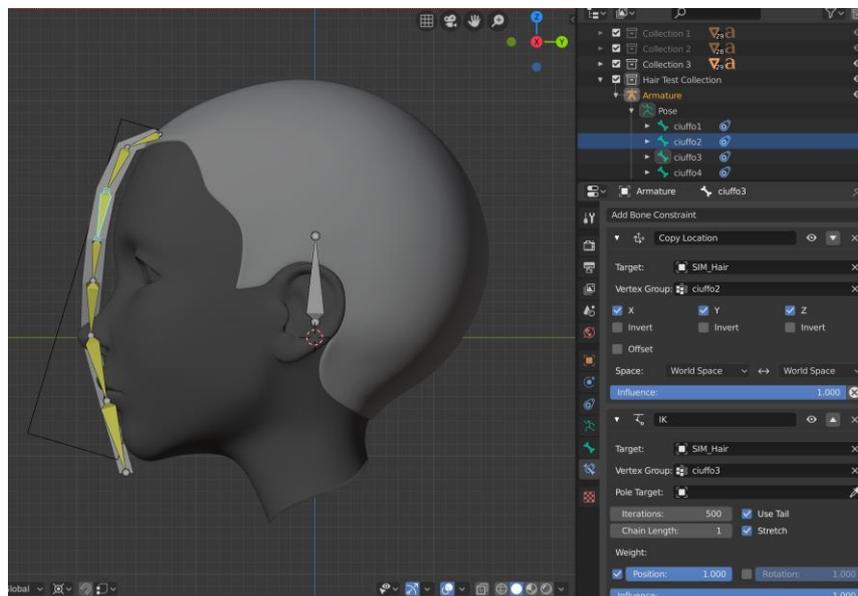


Fig. 4.8 Armature for simulation mesh with modifiers

An Armature modifier is added to the sim mesh so that the sim mesh follows the armature. This creates a Loop Cycle as the armature follows the sim mesh and the sim mesh follows the armature. It is important to have only the armature or the constraints and cloth sim active at one time as Blender does not know how to handle Loop Cycles.

The final step is to add a Mesh Deformer modifier to the base hair mesh with the sim mesh as target. This way the hair mesh follows the sim mesh.

So how does the simulation work?

By deactivating the armature and activating the constraints, the hair and bones should follow the cloth simulation of the sim mesh. Once the simulation has finished running, through the Bake Action command, the animation data can be saved to the bones so that they move with keyframes, following the simulation even when the constraints and cloth sim that tie them to the sim mesh are deactivated.

After activating the Armature modifier, the sim mesh will keep following the bones that are keyframed and the hair will move because of the Mesh Deformer modifier. Now it is possible to move around the bones and overwrite the keyframes if it is desirable.

The results that can be obtained with this method of simulating hair are very pleasing. The mesh moves nicely and there is plenty of control over the simulation as the Cloth Modifier offers many different parameters to play with. Furthermore, the animator can easily edit the simulation after the fact just by moving bones like in any rig. The issue with this approach is that it is quite complicated to set up and requires a rigging process that is very time consuming. From this, the idea of developing an addon that could streamline and simplify the process so that it could be quickly applied to the characters of the teaser and for future work also.

## 4.3 Hair Curves Plugin

The idea of the plugin came from the desire to adapt the Spring hair rig method seen in the previous chapter to the needs of the Reverie production. The plugin works by taking as input hair that has been modeled with paths and curves and giving as output a working simulation rig.

The objectives for the development were:

- Necessity to be able to simulate the hair animation and manually edit it.
- Automatic creation of the rig and necessary components from the modeled hair curves.
- Ability to manage simulation parameters from the scene file and not only from the character file.
- Simple and intuitive use. Ability to easily explain the plugin to others so that it can be used in future productions.

### 4.3.1 Hair curves structure

The plugin works in steps that can be summarized into 6 different phases. Here they are analyzed and one by one and explained.

1. Setting of parameters.
2. Creation of hair and simulation meshes from hair curves.
3. Dynamic adapting of simulation meshes to hair mesh.
4. Rigging.
5. Assigning drivers and modifiers.
6. Output.

## Setting Parameters

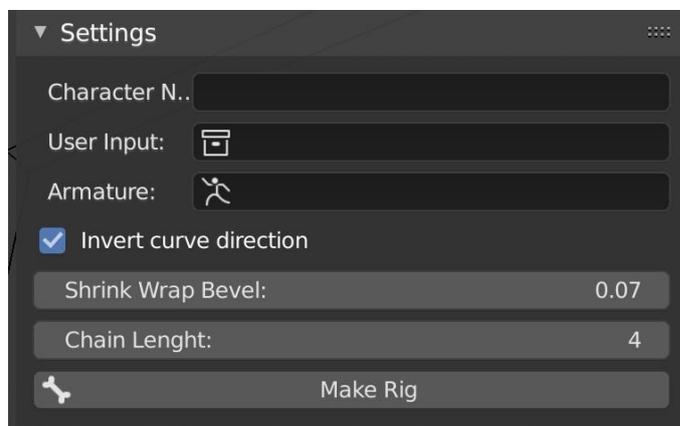


Fig. 4.9 Hair Curves plugin panel

The first parameter of the add-on is the Character Name, this is used to identify the character and to be able to modify its simulation values by addressing its name.

After, there is the User Input. Here, the user indicates in which collection the paths that make up the hair are. The plugin will work only on those collection's paths.

The Armature field indicates to what armature the new rig will belong to. After indicating the armature, it is necessary to also indicate a bone to which the hair will be parented. This is needed so that the hair rig will move together with the rest of the character's rig. That bone is indicated as head bone from now on.

Invert curve direction is a toggle that allows the user to invert the path direction of the hairs if the armature were to be flipped after creation.

Shrink Wrap Bevel is a parameter that influences the dynamic adapting of the simulation meshes to the character's hair. It is suggested to keep it at 0.07 but can be changed if the rig generation fails. Generally, the bigger the size of the hair, the bigger the parameter should be.

Chain Length indicates how many bones should there be per hair. The more bones, the more accurately one can animate the hair, but the more parts need managing.

## ***Creation of hair and simulation meshes***

The next step that the plugin takes is to convert the hair into hair meshes and to create the simulation meshes. Hair Curves creates two new collections, HAIR and SIM. The hair meshes go into the HAIR collection and the simulation meshes into the SIM one. Simulation meshes are created by taking the hair's path, removing the bevel object and selecting from the curve's settings a FULL Fill Mode. This creates a square tube with a number of edge loops that follows the shape of the path. It is then converted to mesh. For each strand, the plugin makes one hair and one sim object.

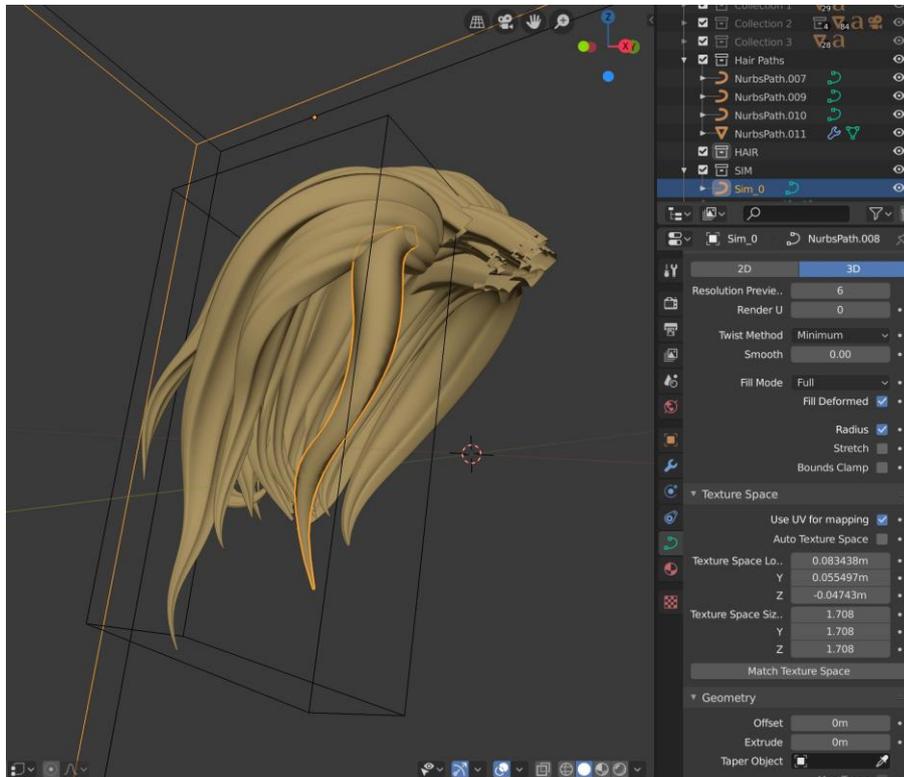


Fig. 4.10 Sim object tube before Shrinkwrap modifier

## ***Dynamic adapting of simulation meshes to hair meshes***

For the simulation rig to work the hair meshes have to be contained into the sim meshes. The closer the fit, the better the simulation will approximate the

behavior of the hair mesh. In order to obtain a snug fit, the plugin applies a Shrinkwrap modifier to the sim object and targets the correspondent hair. The Shrinkwrap modifier allows an object to “shrink” to the surface of another object. It moves each vertex of the object being modified to the closest position on the surface of the given mesh. The modifier also offers an offset parameter that allows the shrunk mesh to grow further from the target mesh in the closest normal’s direction. What the plugin does is to slowly increase the offset parameter and to check for collisions between sim and hair. Once Hair Curves finds no more collisions, it applies the modifier to the sim object and obtains a close fitting skin around the hair. The collision detection is done thanks to Blender’s BVHTree API.

```
def check_intersection(obj1,obj2):  
    depsgraph = bpy.context.evaluated_depsgraph_get()  
    the_object = obj1.evaluated_get(depsgraph)  
    the_other_object = obj2.evaluated_get(depsgraph)  
  
    BMESH_1 = bmesh.new()  
    BMESH_1.from_mesh(the_object.to_mesh())  
    BMESH_1.transform(the_object.matrix_world)  
    BVHtree_1 = BVHTree.FromBMesh(BMESH_1)  
  
    BMESH_2 = bmesh.new()  
    BMESH_2.from_mesh(the_other_object.to_mesh())  
    BMESH_2.transform(the_other_object.matrix_world)  
    BVHtree_2 = BVHTree.FromBMesh(BMESH_2)  
  
    inter = BVHtree_1.overlap(BVHtree_2)  
  
    return(len(inter))
```

The code shows the function used for checking intersections between obj1 and obj2 which are the hair and sim object. The function is called in a loop that increases the offset until no collision is detected.

```
while i==0:
    mod.offset = j
    layer.update()
    print('finish loop')
    if(check_intersection(Hair.objects[n],obj) == 0):
        i = 1
        print('Hair finished')
        obj.select_set(state=True)
        bpy.context.view_layer.objects.active = obj
        if j < 0.06:
            bpy.ops.object.modifier_apply(modifier='SW', apply_as='DATA')
        obj.select_set(state=False)
        layer = bpy.context.view_layer
        layer.update()
    else:
        j = j + 0.001
    if j > 0.06:
        i = 1
```

A control is inserted in the loop. If the offset grows too much, the program assumes that something did not work and the Shrinkwrap modifier won't be applied so that the user can then manually fix the error.

## ***Rigging***

The rigging step of the process works by distributing the amount of bones indicated in the chain length parameter equally along the simulation hair strand. If an equal distribution is not possible, the program attempts to distribute as evenly as possible. The following code indicates how.

```
def part(n, k):
    def _part(n, k, pre):
        if n <= 0:
            return []
        if k == 1:
            if n <= pre:
                return [[n]]
            return []
        ret = []
        for i in range(min(pre, n), 0, -1):
            ret += [[i] + sub for sub in _part(n-i, k-1, i)]
        return ret
    return _part(n, k, n)
```

n is the number of face loops/segments present in the simulation mesh and k is the desired chain length.

The armature is created by positioning a first bone at the center of the first edge loop of a simulation tube. Based on the desired chain length and the number of edge loops in a sim mesh, an edge loop offset is calculated and the tail of the bone is positioned at the center of the edge loop that is offset amounts away from the first loop of the sim object. The bone is then extruded and the next tail is positioned. This is done until the whole sim object is filled with a number of bones

indicated by chain length. During this, vertex groups are created on the sim mesh with the same name as the bones so that when the armature is applied, there is no need for weight painting.

```
# Make the armature based on user input
N = mytool.chain_lenght
arm_obj = bpy.data.objects[mytool.arma]
bpy.context.view_layer.objects.active = arm_obj
bpy.ops.object.mode_set(mode='EDIT', toggle=False)
for sim in bpy.data.collections['SIM'].objects:
    #inside sim collection
    name = sim.name
    number = name.split('_')[1]
    # Get number of vertices for sim elements
    vert_n = len(sim.data.vertices)
    # Get offset number based on N
    segments = (vert_n//4)-1
    offset = part(segments,N)[len(part(segments,N))-1]
    edit_bones = arm_obj.data.edit_bones
    base = 0
    # Get bone number
    for i in range(N):
        if (base==0):
            pin_group = sim.vertex_groups.new(name=f'pin{number}')
            group_mod_last = sim.vertex_groups.new(name=f'hairbone_mod{number}_{N}')
            n_verts = len(sim.data.vertices)
            pin_group.add([0,1,2,3], 1.0, 'ADD' )
```

```

        group_mod_last.add([n_verts-1,n_verts-2,n_verts-3,n_verts-
4], 1.0, 'ADD' )
        b = edit_bones.new(f'hairbone{number}_{i}')
        group = sim.vertex_groups.new(name=f'hairbone{number}_{i}')
        group_mod = sim.vertex_groups.new(name=f'hairbone_mod{number}_{i}
')
        for j in range(0,4):
            group_mod.add([j], 1.0, 'ADD' )
        for j in range(0,(offset[i])*4):
            group.add([j], 1.0, 'ADD' )
        b.head = (sim.data.vertices[0].co + sim.data.vertices[1].co + sim.data.verti
ces[2].co + sim.data.vertices[3].co)/4.0
        b.tail = (sim.data.vertices[0+(offset[i])*4].co + sim.data.vertices[1+(offset[
i])*4].co + sim.data.vertices[2+(offset[i])*4].co + sim.data.vertices[3+(offset[i])*4].co)
/4.0
        base = 1
        tmp = (offset[i]*4)
        a = b
        b.parent = arm_obj.data.edit_bones['Bone']
    else:
        b = edit_bones.new(f'hairbone{number}_{i}')
        b.use_connect = True
        group = sim.vertex_groups.new(name=f'hairbone{number}_{i}')
        group_mod = sim.vertex_groups.new(name=f'hairbone_mod{number}_{i}
')
        for j in range(tmp+0,tmp+4):
            group_mod.add([j], 1.0, 'ADD' )
        for j in range(tmp+0,tmp+offset[i]*4):
            group.add([j], 1.0, 'ADD' )

```

```

        if(i == N-1):
            group.add([j+1,j+2,j+3,j+4], 1.0, 'ADD' )
        b.parent = a
        a = b
        b.head = (sim.data.vertices[tmp+0].co + sim.data.vertices[1+tmp].co + si
sim.data.vertices[2+tmp].co + sim.data.vertices[3+tmp].co)/4.0
        b.tail = (sim.data.vertices[tmp+offset[i]*4].co + sim.data.vertices[tmp+1+
offset[i]*4].co + sim.data.vertices[tmp+2+offset[i]*4].co + sim.data.vertices[tmp+3+o
ffset[i]*4].co)/4.0
        base = 1
        tmp = tmp+offset[i]*4
    bpy.ops.object.mode_set(mode = 'OBJECT')
    for obj in Sim.objects:
        bpy.context.view_layer.objects.active = obj
        bpy.ops.object.mode_set(mode='EDIT', toggle=False)
        n = obj.name.split('_')[1]
        bpy.ops.object.vertex_group_set_active(group=f'pin{n}')
        bpy.ops.object.vertex_group_select()
        bpy.ops.object.vertex_group_smooth(factor=1, repeat=1, expand=1)

```

## ***Assigning drivers and modifiers***

The next step is to add all the necessary modifiers and drivers for the simulation to work.

- A Mesh Deform modifier is added to the hair strands so that they follow the simulation tubes.
- A Cloth modifier is added to the simulation objects so that they can move and be simulated.
- IK and Copy Location constraints are added to the bones of the armature.
- A Child Of constraint is added to the simulation tubes and to the hairs. They are children of the head bone.
- An Armature modifier is added to the simulation tubes
- On the head bone custom properties are created so that the simulation parameters for all the simulation tubes can be controlled from there with drivers. A toggle to enable and disable the simulation is also created.
- The enable/disable simulation toggle is connected to the modifiers in the following way:
  - Armature: 1-var
  - Cloth: var
  - Child of: var
  - IK and Copy Location: var

The result is that when the simulation is toggled on, the armature follows the cloth simulation, when it's off, the cloth simulation tubes follow the armature.

## Output

The output is a completely rigged hair simulation setup. The user only has to toggle on the simulation and let the animation play. Once the simulation is done, using the Bake Action command, the user bakes the keyframes on the armature (It is necessary to toggle on the delete constraints button). After toggling off the simulation, the hair will follow the armature and adjustments to bone positions can be made.

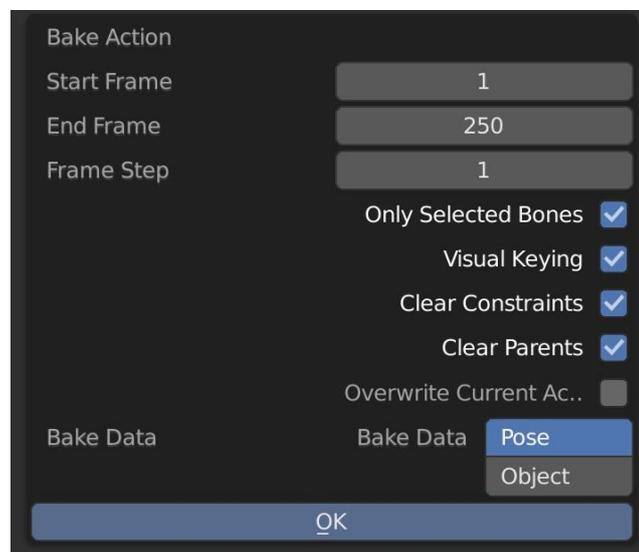


Fig. 4.11 Bake action control with correct toggles on

The addon offers the advantage of also allowing the animator to keyframe the simulation parameters through the head bone drivers and have a hair behavior that changes over time. This has been extremely useful as it allowed Nadya's hair to be stiffer when erratic movement was happening and less stiff otherwise.

Another pro of this approach is that it is possible to add colliders to the character so that the simulation objects can detect objects and interaction between hair and environment is possible.

Although the head bone has custom properties to drive the simulation parameters, the Hair Curves tab also offers the possibility of customizing the parameters and updating them on the character whose name is indicated in the Character Name box. This can be useful when having multiple characters in a scene.

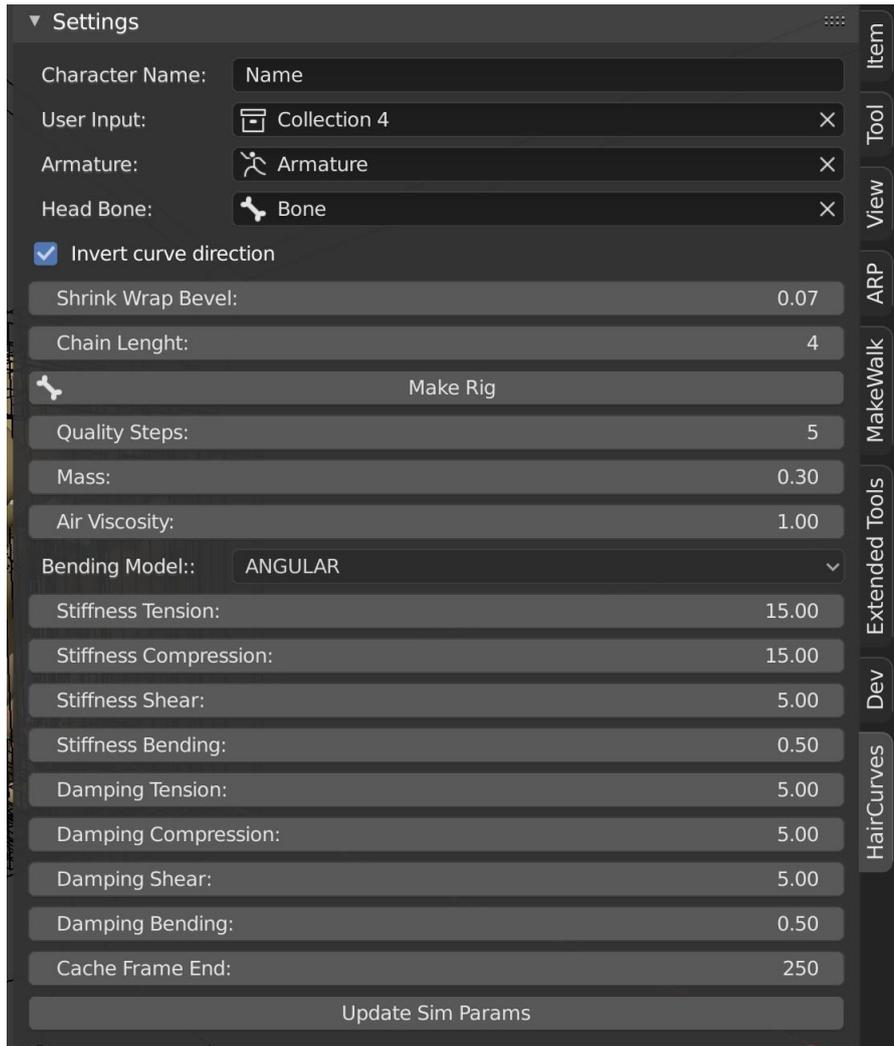


Fig. 4.12 Hair Curves addon control tab for updating sim parameters

The Hair Curves addon has been used on both the characters present in the teaser trailer and has also been tested on other characters that are not present in the trailer. The results have been very satisfactory and, although a lot of time was spent

developing the software, the time that it will save for future production is very significant.

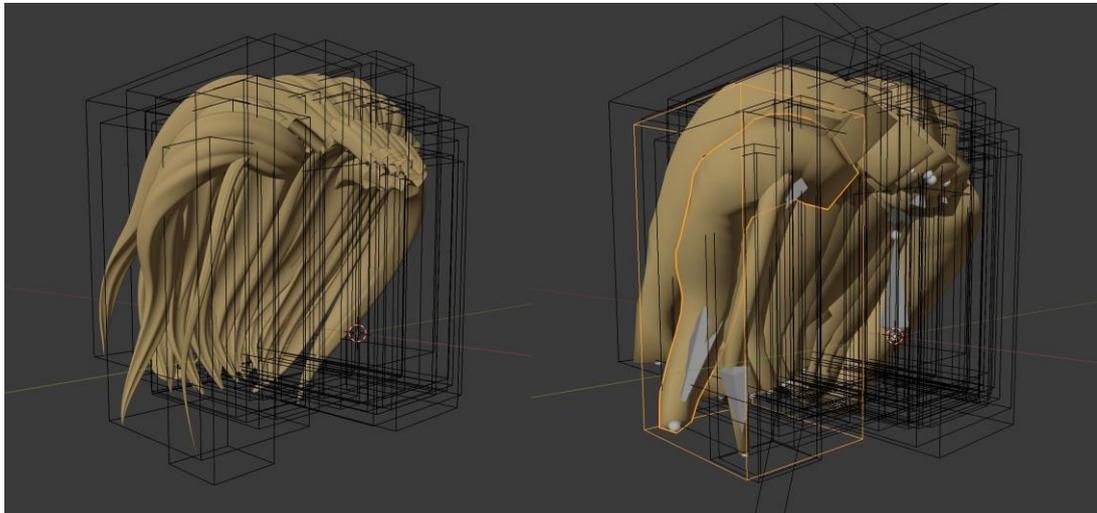


Fig. 4.13 Nadya's hair and hair rig



Fig. 4.14 Jameela's hair and hair rig

## CHAPTER 5

### CONCLUSION

The team was very happy with the results that were obtained. Given the small number of members and the inexperience, being able to finish the trailer was a hard goal, but one that was accomplished.

One of the things that especially brings joy is the fact that the animation has a strong sense of style and direction. The long pre-production work shows in the fact that the inspiration behind the product is clear, but it does not look like a copy of anything. The company that supervised the work was impressed and no major notes were given on the final version.

Working on the project allowed me to develop new 3D skills, to gain new coding notions and to further develop my problem solving abilities, but the thing that I find most important is that it taught me how to work in a real production team.

The cloth simulations workflow was an already established one and it was a matter of applying it to the characters of the trailer. It was challenging to get the simulations right, but I found the development of an entirely new solution to tackle hair simulations more intense.

The Hair Curves tool that I developed could be used in future productions and could be improved even more by further working on it. One of the features that could be added includes a way to simply adjust the number of bones in the rig retrospectively without the need of deleting and creating an entirely new rig. Another aspect that could be improved is the UI. It is functional, but not necessarily very informative if someone where to go into it blindfolded. Finally, it would be nice to integrate the Bake Action tool in the addon panel so that the user doesn't have to search for it by himself. None the less, I am happy with the result and I believe that the tool could also become useful for projects that are not related to Reverie Dawnfall. Maybe one day it might end up online for other to download and use.

## 5.1 Final product







## Bibliography

- [1] <https://docs.blender.org/api/current/index.html> - Blender API
- [2] <https://cloud.blender.org/> - Blender Cloud
- [3] <https://computeranimationhistory-cgi.jimdoofree.com/> - History of computer animation
- [4] <https://www.marvelousdesigner.com/learn/tutorial> – Marvelous Designer tutorials
- [5] <https://robin.studio/> - Robin Studio website
- [6] <https://clothweaver.com/> - Cloth Weaver
- [7] <https://blendermarket.com/products/modeling-cloth> - Modeling Cloth
- [8] [https://docs.blender.org/manual/en/latest/advanced/scripting/addon\\_tutorial.html](https://docs.blender.org/manual/en/latest/advanced/scripting/addon_tutorial.html) - Blender addon manual

