FAKULTÄT
FÜR !NFORMATIK
Faculty of Informatics

Fast Virtual Character Modelling with Mesh Fusion

BACHELORARBEIT

zur Erlangung des akademischen Grades

Bachelor of Science

im Rahmen des Studiums

Software & Information Engineering

eingereicht von

Stefan Schuh
Matrikelnummer 01526991

an der Fakultät für Informatik

der Technischen Universität Wien

Betreuung: Ao.Univ.Prof. Univ.-Doz. Dipl.-Ing. Dr.techn. Eduard Gröller
Mitwirkung: Aleksandr Amirkhanov, MSc

Wien, 12. November 2018

Stefan Schuh	Eduard Gröller

FAKULTÄT FÜR !NFORMATIK
Faculty of Informatics

Fast Virtual Character Modelling with Mesh Fusion

BACHELOR'S THESIS

submitted in partial fulfillment of the requirements for the degree of

Bachelor of Science

in

Software & Information Engineering

by

Stefan Schuh
Registration Number 01526991

to the Faculty of Informatics

at the TU Wien

Advisor: Ao.Univ.Prof. Univ.-Doz. Dipl.-Ing. Dr.techn. Eduard Gröller
Assistance: Aleksandr Amirkhanov, MSc

Vienna, 12th November, 2018

Stefan Schuh

Eduard Gröller

Erklärung zur Verfassung der Arbeit

Stefan Schuh
1170 Wien, Lacknergasse 24/10

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

Wien, 12. November 2018

Stefan Schuh

Acknowledgements

I want to thank my sister, Sofie, who improved my thesis with her feedback. In addition I want to thank my sister, Katja, and her friend Ilja, who helped me to improve the thesis further. Last but not least, I want to thank my family and friends for their support during my whole bachelor.

Abstract

3D models are used in many areas, from medical applications to the development of games. Especially in games and animated movies a lot of 3D models are required and created and the creation process of these is very time consuming. This is one point, which makes creating games or animated movies very expensive and time-consuming in general. To create 3D models faster, artists could be supported by algorithms, which assist them in their workflow. The idea is to reuse parts of existing models and fuse them together with the help of algorithms. Thus in sum a huge amount of time could be saved in the creation of 3D models. We present an overview of existing blending techniques and their advantages and disadvantages. We create our own algorithms, which we use to evaluate how much time artists can save.

Contents

CHAPTER 1

Introduction

3D models are digital representations of objects in virtual environments. They can be, for example, characters or scene objects for animated movies or games. The models are also used to represent technical parts for fabrication or simulation and also for medical purposes, like 3D scans of an human. For the game development a high amount of different models is needed and the creation of these is a time-consuming task for the artists [MT09] [XH07] [AS06].

The goal of the thesis is to find ways, how to assist artists in the first part of their workflow, creating a 3D model. The thesis does not cover the rest of the workflow, like, for example, baking normal or UV-maps. Whenever artists start to create a new model, they can choose to either create it from scratch or use alternate methods like, for example, Photogrammetry, 3D Scans of objects or merge together parts of already existing models. Photogrammetry means to get the mesh of a real object by taking photos of it from different angles and calculate the shape from the image data. In this work we present an approach, improving the creation process of artists, when merging together existing models. The widely spread term for that is called mesh fusion [MT09] or mesh blending [YSL05]. Figure 1.1 shows the workflow of the artists when using parts of existing models with our approach. Using existing models only works, when there is already a good database of models available. This can be the models of, for example, the last game or animated movie, everything created so far for the current project or models from external sources. When an artist found reusable parts, the next step is to prepare them for the algorithm. First, the artist has to cut off the parts he or she wants to use from the existing models. We did this with the difference boolean operation in Blender [Fou18], for example. Afterwards all the parts have to be placed to fit the needs of the artist and the algorithm. For example, if we want to create a three-headed dragon, we place the three heads in front of the dragon with a gap between like shown in Figure 1.3a. Before running the algorithm the artist can parameterize

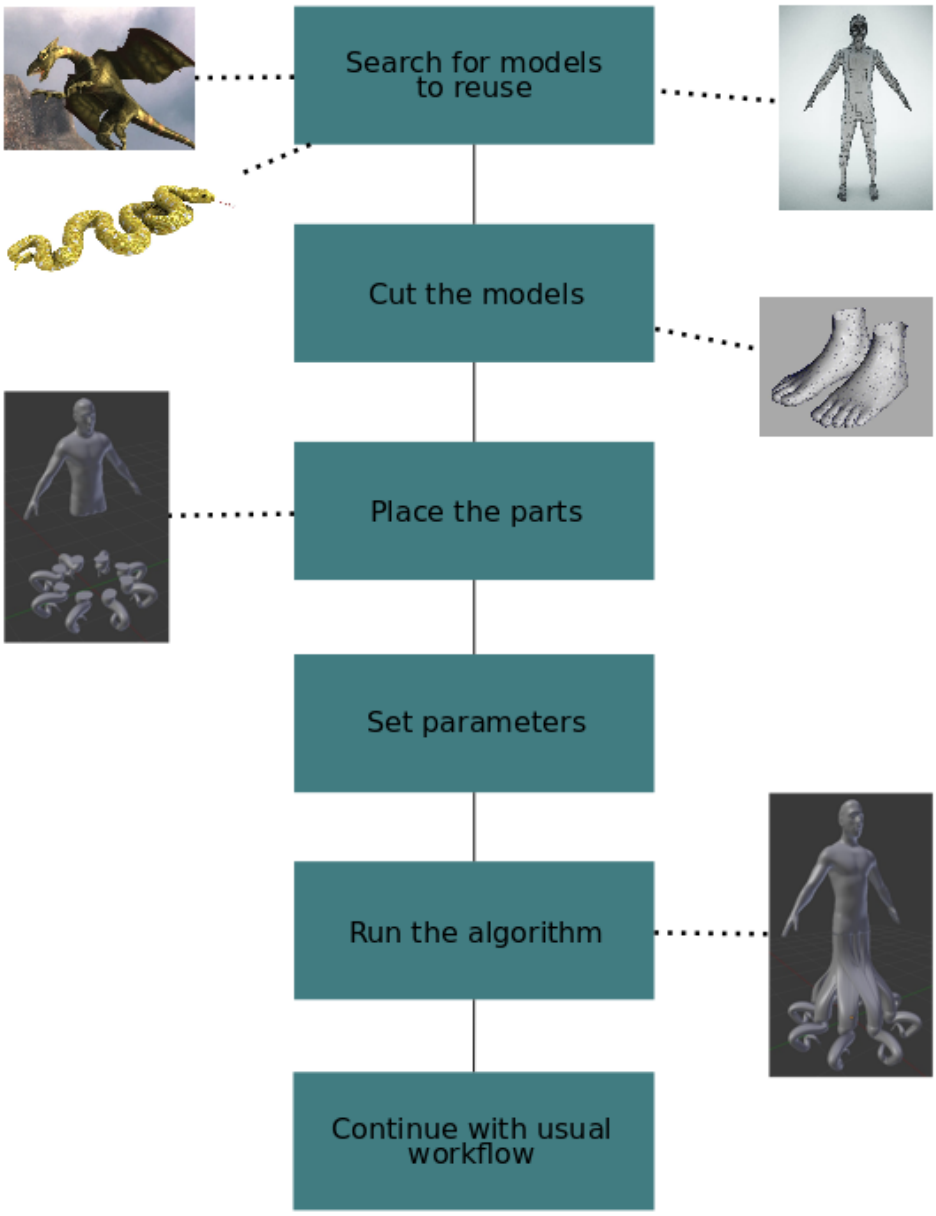


Figure 1.1: Workflow for the artists when merging together parts of existing models. The pictures of the 3D Models are from [Inf08], [fis11], [Shp15] and [Dyn08]

the algorithm if needed. This can be necessary, for example, when we have to specify a blending start and end point or optional, for example, when we want to change the transition by providing a transition function. After the algorithm is performed, an artist can still make changes and improvements by hand, if the result does not fit the artists needs perfectly. If the artist is satisfied with the model, he or she can proceed with

the usual workflow and create textures, normal maps, UV maps, animations and so on. Figure 1.2 illustrates a result of merging two source models provided by our approach. Figure 1.2a shows, what the artist has to create. We used the top of a human body and

(a) The body and tails, like the artist placed them.

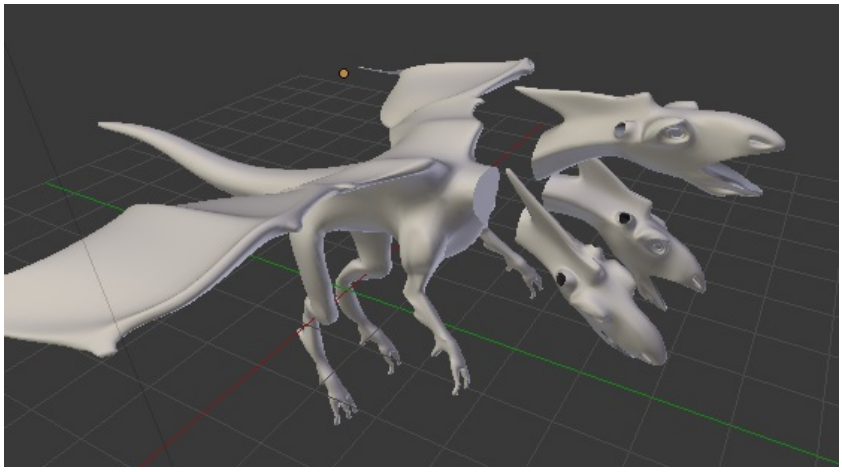
(b) After the use of the algorithm.

Figure 1.2: Model of the octopus man before the algorithm and afterwards. We used our non-overlapping algorithm with transition function e^{-5*x^3} and turn rate function $\sin(x * \pi)/6$. The base meshes were obtained from [Dyn08] and [Shp15].

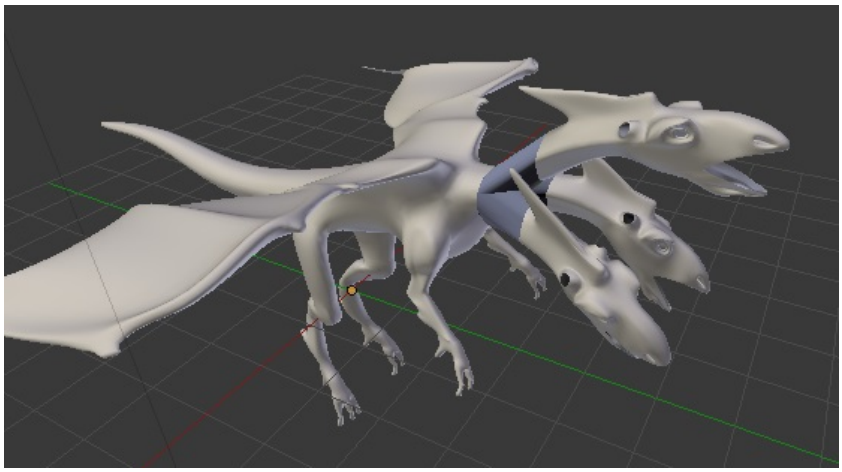
the tail of a snake and copied it several times. What is seen in Figure 1.2b has been inserted by our non-overlapping algorithm. Although, the user has to specify the space between the body and the tails by setting a start and end point for the blending. So when there are already base meshes, which the artist can combine, he or she can create new models much faster. Exact measurements will be discussed later.

Another example is the three-headed dragon (see Figure 1.3), which was created by cutting the head off an existing dragon, copying and placing the heads near to the body, leaving an empty area in between and then letting an algorithm do the rest.

There are rich possibilities of how tools can assist an artist. But one problem so far in the field of 3D mesh fusion is, that the algorithms still require high user interaction. At least the user has to decide which algorithm to use in which case, because one may produce better results in a particular case than another one. There is no algorithm, which detects every case perfectly and chooses the right blending technique. E.g., the leg and the body of a character is blended by technique A, and the leg and the foot



(a) The dragon body and heads, like the artist placed them.



(b) The heads are smoothly merged to the body with our non-overlapping algorithm.

Figure 1.3: Model of a three-headed dragon before the algorithm and afterwards. We used the linear transition function x and no turn rate function. The dragon model is from [Inf08].

is blended by technique B and this is detected automatically. But there is a chance, when there are good combinable techniques, to achieve better tools in the future. So for now, there only could be a set of tools, from which the artist has to select the right one for his or her particular case. Nevertheless, the use of different tools is easy to learn and experienced quickly. The work will prove the worthiness of creating small tools to enhance the modeling process, at least in some cases. This creates the basis for a variety of more powerful tools.

These are the main research questions:

- How can algorithms assist artists in their workflow?
- How much time can be saved in using such algorithms?

To answer that, a literature review is conducted to show what techniques are already present, a new algorithm is developed and the results are evaluated by comparing time requirements for the modeling process for both, crafting by hand and using the developed algorithm.

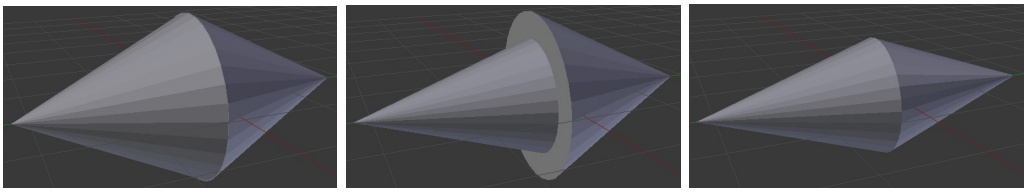
The structure of the work is as follows. First, existing techniques are presented, classified and discussed. Some subproblems that occur and different possible solutions for them will be discussed. Afterwards we present our methods and what problems and limitations we encountered. At the end it is shown, how much time an artist can possibly save, when using the introduced method.

CHAPTER 2

Overview of blending techniques

In this chapter we give an overview of the existing blending techniques and are classifying them. First of all every technique can have the following characteristics:

- Additive
- Subtractive
- Combined



(a) Additive technique. (b) Initial setup. (c) Subtractive technique.

Figure 2.1: Difference between additive and subtractive. Figure b) shows the initial setup. An additive technique would solve the problem by adding volume, see Figure a). A subtractive technique would solve the problem by removing volume, see Figure b).

Figure 2.1 illustrates the difference between the characteristics. The additive techniques are more commonly used. In these techniques new volume is created without changing the original data. Good examples for additive techniques are the ones from Lu et al. [MT09] and Wang et al. [XJ06]. The technique of Gao et al. [XW15], for example, is mostly additive, but they are changing the original data by smoothing the borders of the objects and thus is categorized as combined. A rare example of a technique, which can be purely subtractive as well, is the rolling ball described by Yong

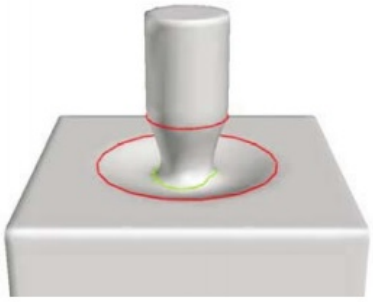


Figure 2.2: The Rolling Ball technique can be used as a subtractive technique. It is like milling the meshes with the ball.

et al. [YSL05], as seen in Figure 2.2 from Figure 4 of their work. In this example the volume is getting less, like we would mill with the ball. This can be achieved with the parameterization of their technique. In general their technique is categorized as combined.

All techniques that we came across can be put in one of the following four classes. We are introducing these names based on our own observation.

- Non-overlapping techniques
- Overlapping techniques
- Morphing techniques
- Volumetric techniques

2.1 Mesh techniques

2.1.1 Non-overlapping techniques

Most of the techniques build up a surface transition in the space between two or more objects. So the objects are not touching each other and the gap or missing information between objects is filled up. This is why we call them “non-overlapping techniques” in this thesis. Because of this nature, they are commonly purely additive. These techniques are very promising to aid artists in their work. There are fast ways to implement such a technique and in many cases it would take the artist a much longer time to fill the space by hand. We implemented such a technique and will show that it is able to provide good results.

A good example for a non-overlapping technique is the work of Lu et al. [MT09]. They had the goal to improve the modeling process as well. The three-headed dragon seen in 1.3b was achieved very similar to their technique described in the paper. The difference

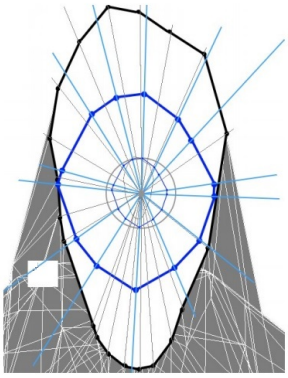


Figure 2.3: Create angular correspondence by overlapping the border contours of the meshes and shooting rays from the mid to all vertices. Points that are hit by the same ray do correspond. The black lines represent the contour of one mesh and the blue lines of the other one. The gray patches are the connecting surfaces of the first mesh.

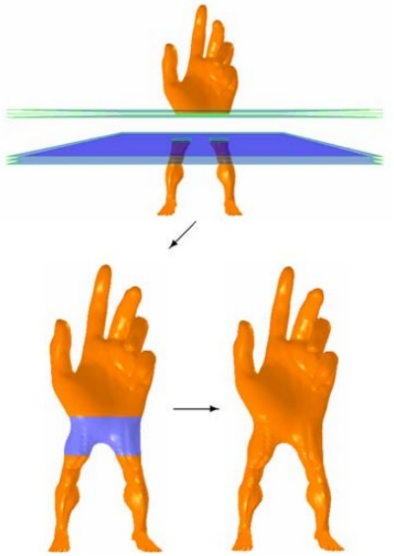


Figure 2.4: Example of the walking hand, which is created by merging together a hand with 2 legs. The objects are placed with a gap in between and a transition surface is created to connect them.

is, that we have to specify the start and end point in our technique and the transition can be parameterized with our solution. Their technique finds openings by checking which edges are only used once. Therefore, it only works for single openings. The, in this way found, contours are projected onto a plane. With that projections, they can overlap the contours, calculate mid-points and shoot rays from the mid to the outside in different angles. This is illustrated in Figure 2.3 from Lu et al. [MT09] Figure 7. The

intersections of one ray on the contours now have a perfect correspondence. This allows interpolations of the contours to get a smooth transition. The ray shooting is used and improved during our work and will be discussed again later on.

Wang et al. did something similar [XJ06]. The main difference is, that they are using implicit functions to describe the openings. In addition, they can cope with multiple openings. They use scattered data interpolation [GT05] to calculate the implicit functions of the openings. That means, that they are searching a function, which interpolates all vertices best. Mathematically this can be expressed as a linear equation system and solved. Another possible approach to calculate the implicit functions of the openings, is by using a signed distance field and sample it on a regular grid. They have implemented it as well and compare it in their work. Their technique produces very smooth transitions between arbitrary objects. Figure 2.4 is an example from Wang et al. Figure 3 [XJ06].

Xu et al. [YY04] use Poisson equations to generate missing mesh surfaces and create smooth transitions. To fuse together two meshes Xu et al. [YY04] extract the border points from each mesh and find correspondence between the vertices of the two borders. Then the local orientations of each vertex are computed. An intermediate border is created, which can be a simple interpolation of the two borders. The original borders are updated to have the same connectivity as the intermediate border. Then the local orientations of the vertices of the borders are compared to the intermediate border and propagated from the intermediate border to the two meshes. Finally a linear system is set up for all vertices and solved to create the new fused mesh. A disadvantage of all Poisson based techniques is, that the larger the gaps to fill are, the worse the result gets. Figure 2.5 shows some result from Xu et al. Figure 1 [YY04].

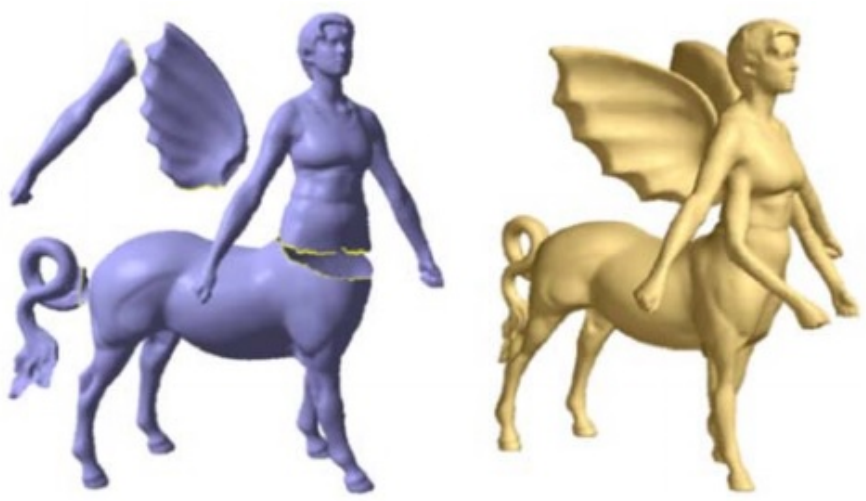


Figure 2.5: Result of the Poisson based technique. It merges together several different parts to create a four-armed wing centaur with smooth transitions.

Gao et al. presented a work, where a field of Bezier control points is used to construct a smooth surface between two meshes [XW15]. First they extract the borders of the meshes and smooth them. Then they create Bezier control points with their own technique. They are using so called control lines, which go parallel and perpendicular to the borders. This is how they overcome the fact, that the crown has much denser vertices than the root of the tooth. Then the Bezier surface is sampled to get new vertices. The sampling points are increasing when going from the root to the crown to match the density of the vertices. In the end they are smoothing their result.

From our own observation the most of the non-overlapping techniques all share similar steps, which are:

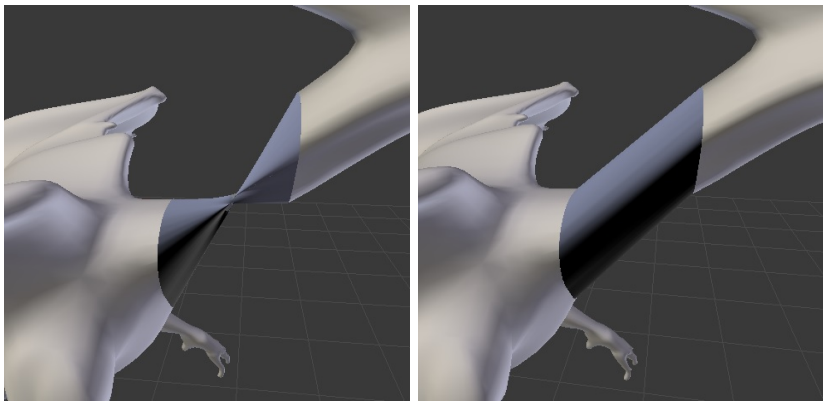
1. Extracting contours of the borders of the objects
2. Finding correspondence between the contours
3. Create transition via interpolation
4. Tessellation

Extracting contours

There are various ways to achieve it. For example, in the approach presented by Lu et al. [MT09] edges, which are only used in one face, are searched in the mesh and are thus borders. The disadvantage of their technique is, that it only works for open meshes. However the advantage is, that the artist does not have to specify where to fuse the objects together. Wang et al. [XJ06] use defined planes to get the borders. The advantage is, that we get a coplanar contour and do not have to project them onto a plane like Lu et al. [MT09] did. The disadvantage is, that the user has to specify the planes.

The technique presented by Gao et al. [XW15] is a special case, because they do part of the step "Cut parts" from the artists workflow. This is because in their work they have given the crown of the tooth, which is already a cut part, and the whole tooth in a lower resolution, from which they want to cut off the crown at its border. As the first step they search for the mesh opening of the crown like Lu et al. [MT09] did. With that border they use a closest point algorithm to get border points on the tooth root mesh. For their purpose using a plane would be unacceptable, because they would lose information that they want to preserve as much as possible. Using this technique in the modeling process means, that the artist has to cut only one mesh and place it over the other one. The second mesh is cut automatically according to the border of the first mesh. This technique could also be an overlapping technique, but we decided to classify it as a non-overlapping technique. After the first step we get two borders with a gap in between, which has to be filled.

Finding correspondence



(a) Contours having the wrong correspondence.

(b) Contours having the correct correspondence.

Figure 2.6: Transition with once the wrong and once the correct correspondence. We achieved Figure a) by defining a correspondence with the vertex located at the opposite site. So vertices at the bottom are corresponding to vertices at the top and vice versa.

To make the next step, creating transitions, even possible, we have to think about the correspondence problem. If we do not take care of it, the result could look something like in Figure 2.6a. The figure shows, that, for example, the point on the bottom of one contour is interpolated with the point on the top of the other contour. Figure 2.6b shows how the transition looks like, if the correspondence is correct. This section gives more insight into this problem.

One possible solution for this problem would be expressing the contours as implicit functions and work with them like Wang et al. [XJ06] or O'Brien et al. [GT99] did. For every contour, the x and y axes has to be the same, because the implicit functions are defined based on them. This ensures perfect correspondence. It requires to work with mathematical functions, but seems to work for arbitrary contours.

Xu et al. [YY04] are projecting the contour of one object onto the other, in the user defined direction. For each vertex they shoot a ray in the defined direction and select the nearest vertex on the second object.

Another solution is to create an angular correspondence in the projected 2D space by shooting rays like Lu et al. [MT09] did, which is displayed in Figure 2.3 out of their work. They lay the contours on top of each other and shoot rays from the midpoints of the contours to every vertex. Every ray hits both contours and creates correspondence this way. This works well in many cases, but has some downsides. A ray can either hit no edge of the contour or multiple edges, which leads to problems, that are explained further. Figure 2.7 shows an example, which is described in the following. If we have

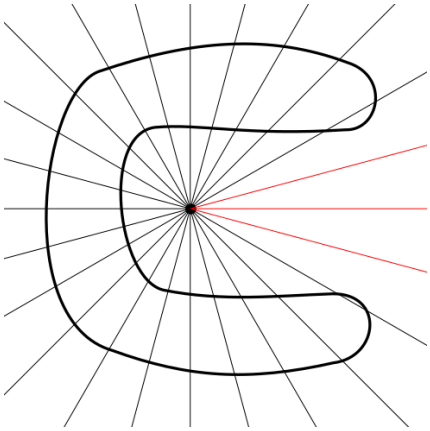


Figure 2.7: Mid-Point of all vertices is outside of the contour. So some rays, marked in red, do not hit any edge of the contour. Therefore no correspondence can be found.

given such a contour, the mid point will be definitely outside and thus it will be hard for any other contour to create a correspondence. The rays at the angles marked in red do not hit the contour, consequently another contour can not match its hit to something.

Even if the mid point is inside the contour, another problem can occur. An example can be seen in Figure 2.8. Here, each ray has one to three intersections. The

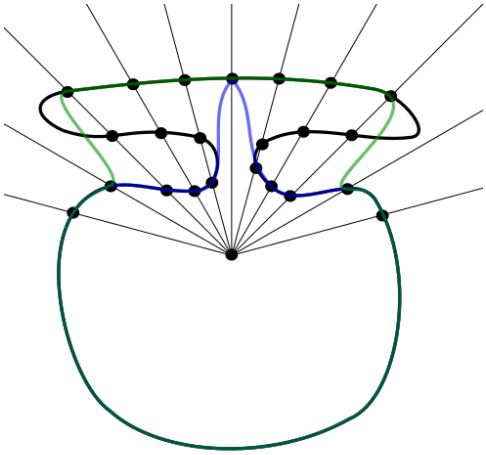


Figure 2.8: Contour with multiple correspondence solutions and information loss. If we want to interpolate to that contour, it will either interpolate to the green or to the blue contour. We are losing the information of some features.

algorithm has to decide, which point it should take. It could take the most farthest away point, then the resulting contour would be like drawn in green. If it takes the most nearest or first point, then the resulting contour would be like drawn in blue. In both

cases we lose some of the information of the original contour. Interpolating any contour onto this points will not give us the expected result.

In general, this angular correspondence approach only works for convex contours. Otherwise, it will fail completely or at least produces some unwanted artifacts, where the transition does not fit to the border contours. An advantage of that approach is, that it is very fast to implement.

In conclusion, it is very important how a correspondence between the contours is created, otherwise it will not produce pleasant results. Fred Park [Par11] shows some other possibilities to describe shapes. That could be potentially used to create a good correspondence.

Create transition

The transition can be done in several ways. Using implicit functions like Wang et al. [XJ06] and O'Brien et al. [GT99], or Poisson equations like Xu et al. [YY04]. In detail Wang et al. [XJ06], for example, use another plane behind the border plane to calculate the difference of the implicit function. That way they get the change of the function in the blending direction. With the implicit functions of the border contours and the change of them given, they use Hermite basis functions to create the transition surface.

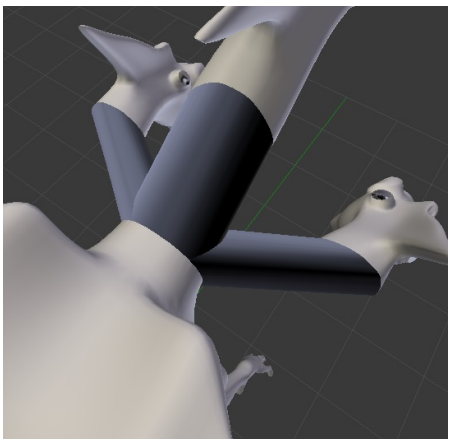


Figure 2.9: An example of a linear interpolation with our algorithm. The results are good, but can produce sharp edges at the seams.

Gao et al. [XW15] are creating control points using their own alternate Bezier surface. One way is to just interpolate the corresponding contour points, like Lu et al. [MT09] did and we did as well. When we are just using linear interpolation the results look already good - see Figure 1.3b for reference. But when the heads are moved apart from each other (see Figure 2.9) linear interpolation probably gives not the preferable result in that case. Therefore we advise to offer the artist some options and possibilities to

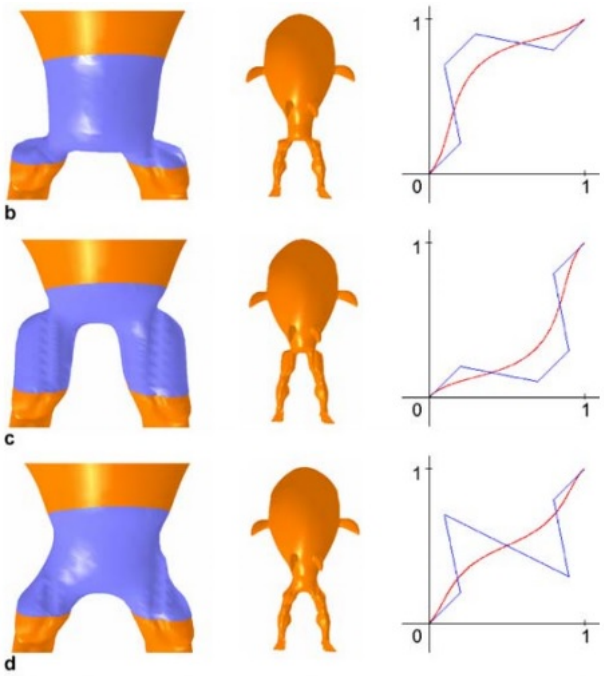


Figure 2.10: Different parameterizations of the transition and their outcome are shown. This leads to more flexibility and more possible transitions.

parameterize the transition. Wang et al. are offering parameterization as well, as shown in Figure 2.10 from [XJ06] Figure 4.

Tessellation

Tessellation is an finalization step, which only makes the mesh more appealing and easier to process further. The point of that step is, to give the generated mesh the same structure of vertices and edges as the source objects. Generally speaking the more homogeneous the resulting mesh is the better. Several authors like [NC03] Oñate et al., [BL10] Liu et al., [Mor01] Moreton and [MF09] Boulos et al. covered tessellation.

Some of the authors use their own tessellation technique like, for example, Wang et al. [XJ06]. Signoroni et al. [MC15] are using an adapted version of the Delaunay tessellation. The tessellation of Gao et al. [XW15] is immanent in their technique. They have given a high resolution crown and a low resolution root and are thus using their own Bezier surface, which is creating an already tessellated transition surface.

2.1.2 Overlapping techniques

Other than the non-overlapping techniques, there are no similar steps to be observed in other techniques. They all have unique approaches. We are naming the overlapping

techniques after the fact, that models have to be placed above each other, so that they are intersecting. These techniques are resolving the intersection with a smooth transition. The overlapping techniques are not so promising for artists, because the merging often can be done very fast by hand. More details will be shown in the evaluation. Nevertheless they could be used for automated creation of random characters, for example, or when the artist has to merge many parts a lot of the time.

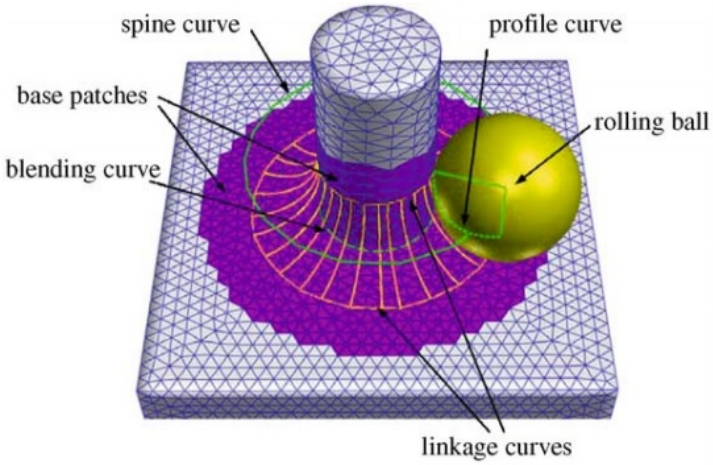


Figure 2.11: The rolling ball technique with all the terminology and what they mean graphically.

The approach of Yong et al. [YSL05] is a typical example for an overlapping technique. Their technique is called "Rolling Ball", because the blending result can be imagined as a surface formed by a ball rolling around two target meshes. Figure 2.11 from Yong et al. Figure 1 [YSL05] illustrates the main idea of the technique. The first step is to get the intersection contour of the two meshes, which they call blending curve. This can be done in various ways, which are suggested by the authors. For example, the intersection can be detected automatically or selected by an user. The next step is to get base patches of the two meshes. This is done by extracting all polygons in a specified radius r around the blending curve. The radius is multiplied with a factor α. The α ensures, that an intersection will exist, even when the vertices are far apart. Typically α is the factor 1.75, but if the vertices are further away from each other, the value should be higher. All vertices are moved along their normal for the radius r. Then the moved patches are intersected. The intersection of the patches is then called the spine curve. For all vertices in the spine curve a profile curve is created and then connected to a blending surface. This gives a smooth blending between the objects around the intersection contour, like we would draw with an opaque ball (see Figure 2.12). The transition can be parameterized to be additive or subtractive, so there is a variety of different transitions that can be achieved.

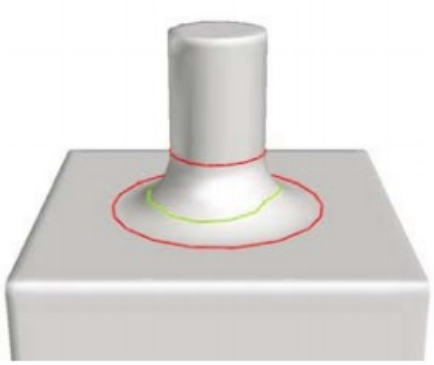


Figure 2.12: Result of the rolling ball technique. The sharp edge at the intersection got blended with a smooth transition.

Sharf et al. [AS06] is another good example. The artist has to drag one part over the other until they overlap and their algorithm automatically snaps them together. It uses a closest point algorithm to find correspondence in the overlapping area of the models. The algorithm creates a smooth transition by locally interpolating corresponding vertices. The big advantage here is, that the artist does not have to place the parts precisely.

2.1.3 Morphing techniques

The workflow of the artists is different, when using morphing techniques, depending on the technique used. For example, instead of cutting and placing parts of objects, the artist has to specify a border on both objects, which should correspond. All vertices at a specified side of the borders are then morphed via interpolation or replaced entirely with something else.

A good example for a morphing technique is the work of Mitani et al. [TK99]. They create correspondence between the faces of the meshes and interpolate them using a parameterizable curve to create smooth transitions. The user just has to select the corresponding contours and specify the interpolation function to get a new morphed mesh. Figure 2.13 shows some result from Mitani et al. Figure 9 [TK99].

Lévy et al. [BL03] did something similar. With their technique it is possible to combine meshes in their parameter space, which is the 2D representation of the mesh like the UV-Map. By approximating a minimal energy surface, they are recalculating the new 3D object. The user specifies the parts, which he wants to merge together and a transition area on both models. Their algorithm searches for corresponding vertices and cuts the models on the found contours to merge them together using Poisson equations.

Another good example is the work of Au et al. [XH07]. They use Poisson equations as well. With their technique the user has to only select an area, where the border should

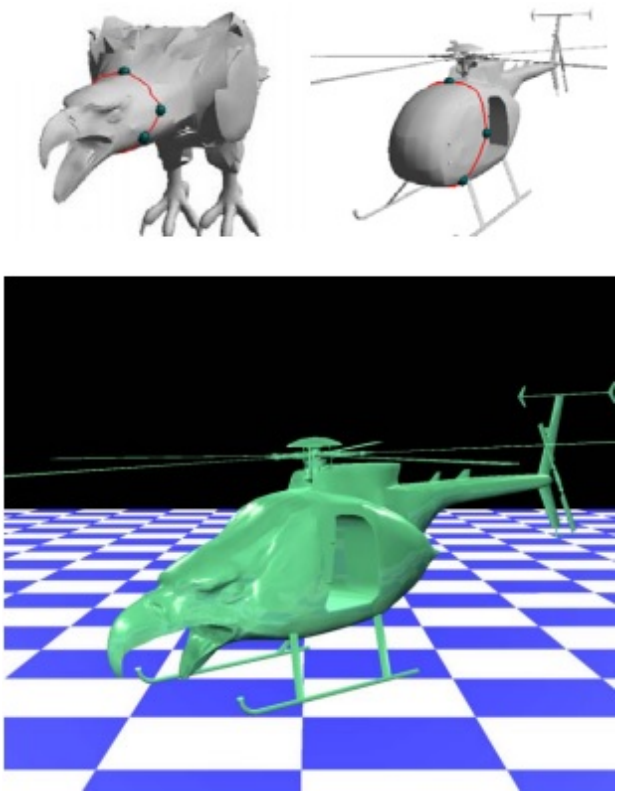


Figure 2.13: The morphing of the head of an eagle onto the front of a helicopter. The artist has to define the border on each source object, marked in red.

be, which is easier and faster for the user. They are finding the best contours by taking the border with the shortest path in the user specified area of one mesh and trying to find a similar corresponding border with a least-squares solution in the user specified area on the other mesh. With this approach they try to minimize the distortion at the seams.

Shilane et al. also presented a solution to fuse together 2 meshes, described in [TF04] chapter 6.2. In general their work empowers everyone to select parts of objects from their 3D model database and combine them to a new model.

The morphing techniques work best, when there are similar parts in the source meshes, that should be replaced or mixed, but that is not required but can lead to distortions at the seams.

2.2 Volumetric techniques

The difference of the volumentric techniques to the others is, that they are based on volumetric representations of the models. A possible approach would be to convert to

a volumetric representation, process it further and convert it back to a mesh again. Museth et al. [KM02] used this approach and presented various operations which can be applied on volumetric models, including blending, smoothing, opening and closing. In particular they are using Level Set Models to work with, which are deformable implicit surfaces, where the deformation is controlled by a speed function. The resulting models can be transformed back into meshes with, for example, the Marching Cubes Algorithm [WEL87]. Great advantages of their technique are, that they always produce non-self-intersecting surfaces and there are no issues with edge connectivity or mesh quality in general. Recalculating back to a mesh always gives a high quality mesh, which is perfectly fit to process further. Disadvantages can be a higher memory consumption and overhead when transforming models from one representation to another. In general these techniques are better, when smooth results are needed, but Museth et al. [KM02] claim, that they also can produce and keep sharp features. Figure 2.14 from Museth et al. Figure 11 [KM02] shows how they created a winged 2 headed dragon with their technique.

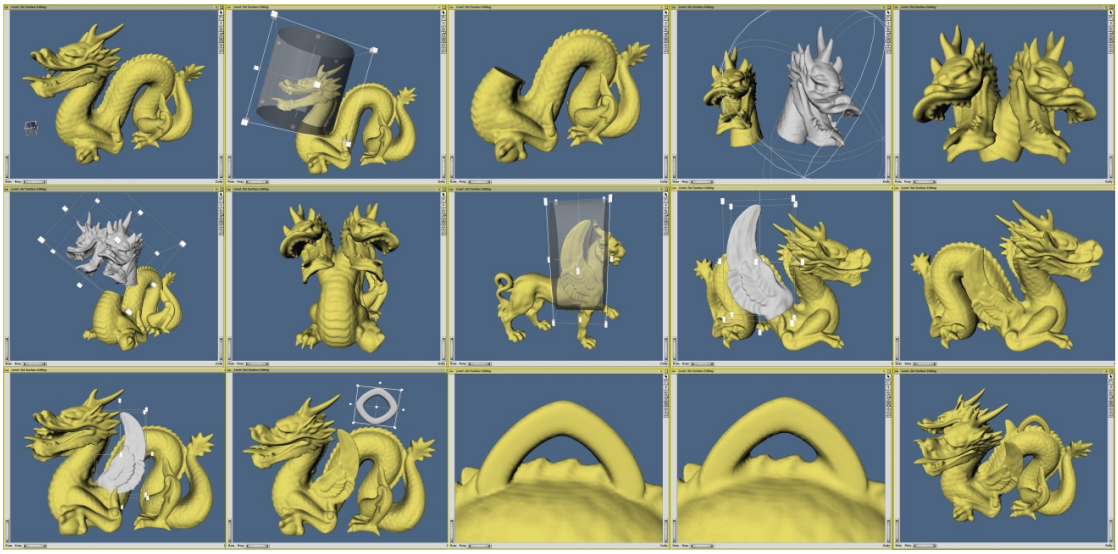


Figure 2.14: The head of the dragon is cut off, duplicated, merged together and again fused with the body. Further the wings are taken from another model and merged with the body. Finally a ring is blended onto the back of the dragon.

Signoroni et al. [MC15] use volumetric representations as well. They convert the models to octree representations with their own vertex protection technique. They memorize vertices around the borders to preserve them, when converting back to a mesh. The borders have to be chosen by the user. The gap between the borders is filled following Poisson equations. After they created the missing parts, they cut out these new parts and insert them back into the original mesh. The protected border vertices are used to make this possible. The great advantage is, that the transformation to volumetric representation and back does not change the mesh globally, preserving all details.

Another technique would be to use Metaballs. The advantage is, that they are smoothing merging objects naturally. There can not be self-intersecting surfaces either and all transitions are smooth. The disadvantage is, that we need source models in metaball presentation, otherwise we need to convert them to metaball presentation first.

CHAPTER 3

Our methods

We implemented our own algorithms for evaluation purposes. One algorithm is an additive non-overlapping technique. It produces good results, which are shown in the evaluation. Another algorithm is an additive overlapping technique. It does not produce good results, but it is presented as an reference for the reader.

3.1 Non-overlapping technique

3.1.1 Extracting contours

For the implementation in this work a similar teachnique Wang et al. [XJ06] presented is used. The user has to specify two points. The direction from one point to the other is the blending direction. Each point has to lie on the borders of one object. A plane at each point is created orthogonal to the blending direction. The intersection with the plane gives the contour of each object (see Figure 3.1). The yellow dot on the left plane is the blending start point and the orange dot on the right plane is the blending end point. It also could be the other way round.

In the example we can see, that each object can have multiple borders. The dragon heads got some kind of horn on top of them. In general, that is a huge problem, because how should the algorithm know, from which border it should create a transition - or should it even create a transition from all borders and how should this look like, if that is the case. The solution is to just use the border with the largest perimeter. A tool, which should help artists, has to offer some options here, so that it is useful in many cases. In this case, it would be the option “Only use border with highest perimeter”. Another solution is to select the border, which midpoint is the closest to the midpoint of the contour to blend to.

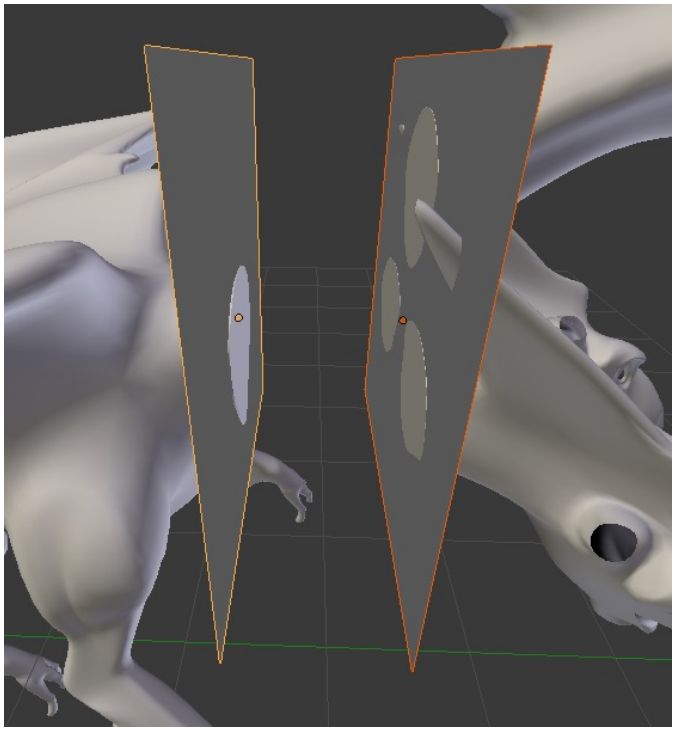


Figure 3.1: Getting borders of the objects by intersection them with defined planes. The yellow and orange dots mark the blending start and end point, set by the user. The planes are created orthogonal to the direction from the start to the end point. The objects do not have to be perfectly cut, so, for example, the horn on the right side is preserved in the final mesh.

3.1.2 Finding correspondence

We are using a method based on the angular correspondence by Lu et al. [MT09]. Because of the disadvantages of that method, another method, which can cope with the presented problems in chapter 2.1.1 was found, but it is not perfect either. The approach is explained based on Figure 3.2. First, the midpoint of one contour is computed and a ray in any direction is shot. We are using the direction of the x-axes, for example. We get the intersection with the ray of each contour and take the nearest vertex. The two in this way found points are now the start points of each contour. In Figure 3.2 that is the green and blue dot on the left side. They are at distance zero. Then we calculate the distances from the start vertex to the next vertices clockwise for each contour. This is simply done with the linear distance between the vertices. This is enough for good results, but one can also use more accurate and complicated methods. As a result, we get two contours with their perimeter. We calculate a corresponding point for each vertex, when simply calculating how much percent of the perimeter this vertex is away from the start point. So we can calculate the point on the other contour, that lies on that percentage of its

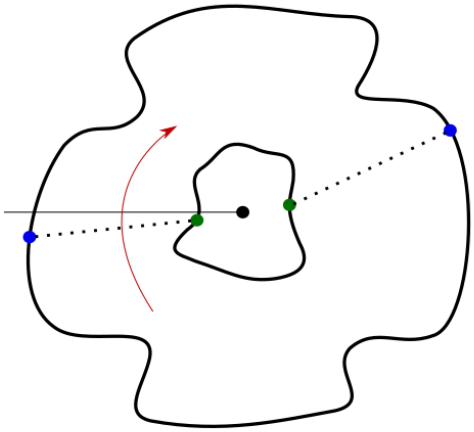


Figure 3.2: Two contours and the found correspondence with our distance matching approach. A ray is shoot to the left or in any other direction. The nearest vertex counter clockwise is selected on each contour. They do correspond and are set to distance zero. A correspondence for the other vertices is found by moving along the contours. The dots on the right side, for example, are at about 50% of the distance along the contour.

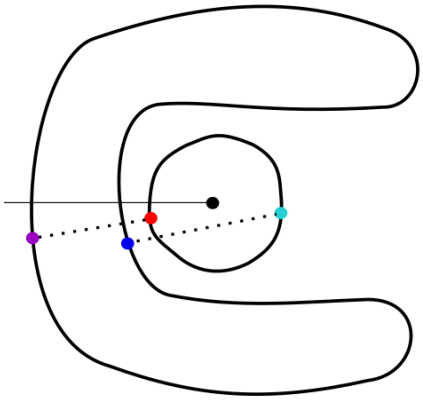
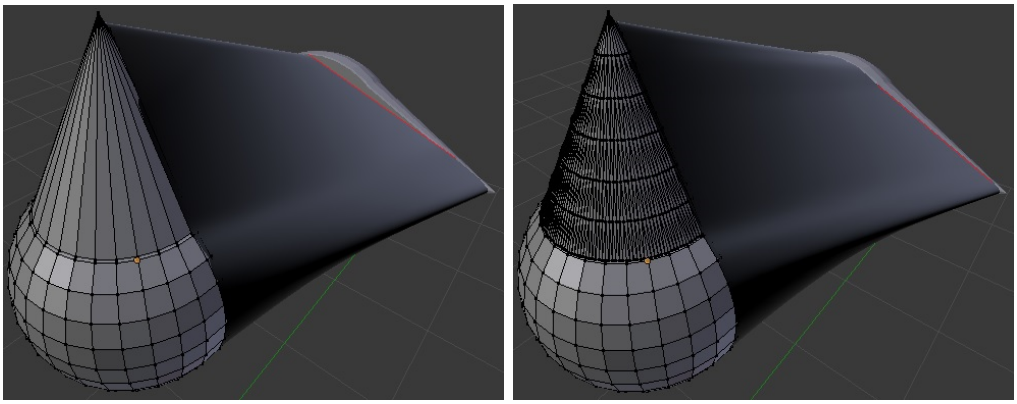


Figure 3.3: Correct correspondence is found with our distance matching approach, even on non-convex contours. When multiple edges of the same contour are hit by a ray, the vertex on the most farthest away edge is selected.

perimeter. If we are, e.g., calculating 25% and the contour got vertices at 20% and 30%, then linear interpolation between the two points is used. So, e.g., the second green dot on the right side of the circle in Figure 3.2 is on about 50% and it corresponds to the second blue dot on the right side, which lies on about 50% on the other contour.

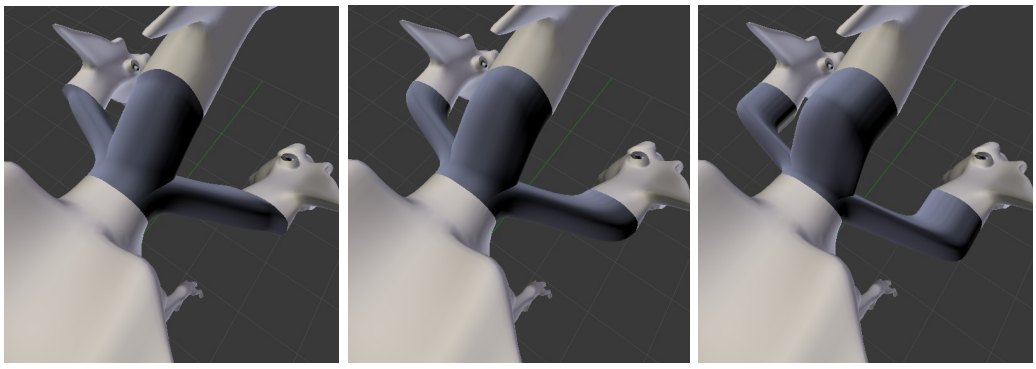
To overcome the problem stated in Figure 2.7 and Figure 2.8, we try out several rays in



(a) Not enough points on the contour. (b) Enough points on the contour.

Figure 3.4: Our distance matching method has some downsides as well. The Figure a) shows an artifact which can occur. When there are not enough vertices on the contour, some arcs of the other contour can be cut off, shown with the red line. In Figure b) the problem is solved by subdividing the cone.

different angles, until we get an intersection with both contours. In addition, we take the intersection which is the farthest away, in the case there are more than one, to avoid to get a twist as shown in Figure 2.6a. Based on Figure 3.3 we explain why this can happen. It shows the correct correspondence when shooting a ray to the left. We take the red dot for the circle, because there is no other option. But for the other contour, we have to take the farthest away intersection, marked as purple. If we would take the blue dot, the interpolation would have a twist in it, because it does not correspond to the red dot, but to the cyan dot. The correct correspondence is marked with the dotted lines.



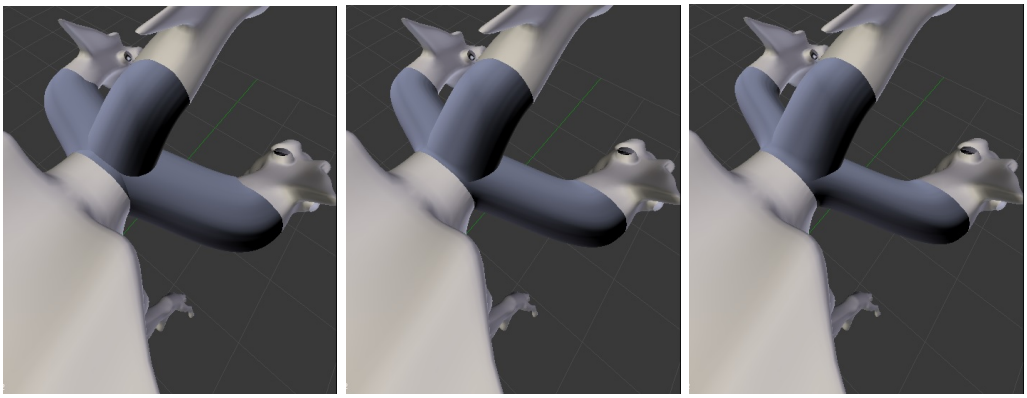
(a) $e^{-2.5*x^3}$. (b) e^{-5*x^3}. (c) e^{-25*x^3}.

Figure 3.5: Different transition functions and their outcome. By parameterizing the blending with a function, many different transitions are possible. This gives artists more options and freedom.

As mentioned before, the approach has some downsides as well. If there are not enough points on a contour, artifacts will occur such as the transition does not fit perfectly to parts of the contours. This is showed in Figure 3.4. In Figure 3.4a it can be seen, that the inserted surface does not perfectly fit onto the mesh. This occurs, because the cone has just one vertex at the tip and the next vertices are on the sphere. Assuming, the tip got a distance percentage of 25%. Then a point at 25% on the other contour is found. Assuming the next vertex got a distance percent of 40%, a point at 40% on the other contour is found and connected. This is marked as the red line. When there is a huge distance between 2 points, it could be, that an arc of the other contour is cut off. In Figure 3.4b this problem is fixed by subdividing the cone a bit. Another solution would be to just take, e.g., 100 points on each contour, uniformly sampled at every percent. This way, there should be enough points as well, but probably the merging with the meshes and tessellation needs to be better.

3.1.3 Create transition

We are creating the transition by interpolating one contour to the other. The transition can be expressed by a function and therefore parameterized. For the approach in this thesis, the function itself is of less importance, as long as it goes from zero to one in the function range zero to one. In Figure 3.5 a variation of a Gaussian-like function can be seen. It may not look aesthetic in this example, but this depends on the user's preferences. The parameterization with any function is very powerful and allows almost every imaginable transition.



(a) $\sin(x * \pi)/2$. (b) $\sin(x * \pi)/6$. (c) $\sin(x * \pi)/10$.

Figure 3.6: Transition function combined with different turn rate functions. We used the transition function e^{-5*x^3} to produce the shown results. With the possibility of an additional turn rate function, even more freedom for creating the transition is reached.

The only thing that looks strange in Figure 3.5b is, that the middle parts of the necks are too thin. To correct this issue, the idea is to rotate the contours in the middle. For

this purpose a turn rate function in addition to the transition function was introduced. The axis for rotation is calculated depending on the relation between the midpoints of the contours. As the contours are defined in the local space of the intersecting planes, the up-axis is always the blending direction. It is as easy as taking the crossproduct of the up-vector and the difference between the midpoints. The rotation value is defined by $functionvalue * \pi$. A function value of one equals 180 degrees of rotation. If the turn rate function is not used, it simply has to be set to zero. The rotation is executed in the local plane space, before transforming the contours back to world space.

In search for a function, that would slightly turn the contours, until the mid of the transition was reached and turns back again afterwards, the function $\sin(x * \pi)/6$ was found to be suitable. With that function, in addition to the transition function, the results shown in Figure 3.6 were reached.

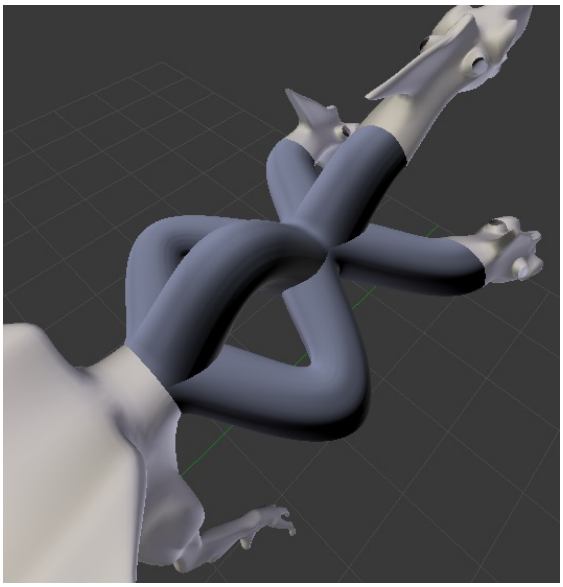


Figure 3.7: An example showcase of a snaky dragon neck to demonstrate the power of our parameterizable functions. We achieved that with the transition function $(sin(\pi/2 + x * 3 * \pi) + 1)/2$ and the turn rate function $\sin(x * 3 * \pi)/6$.

As a result, the combination of that two functions allows even more freedom in parameterizing the transition. With the transition function $(sin(\pi/2 + x * 3 * \pi) + 1)/2$ and the turn rate function $\sin(x * 3 * \pi)/6$ the snaky looking transition shown in Figure 3.7 has been created.

3.2 Overlapping technique

We tried to find an overlapping technique that works similar then our non-overlapping technique. Although this approach was not successful, it is presented as an reference for

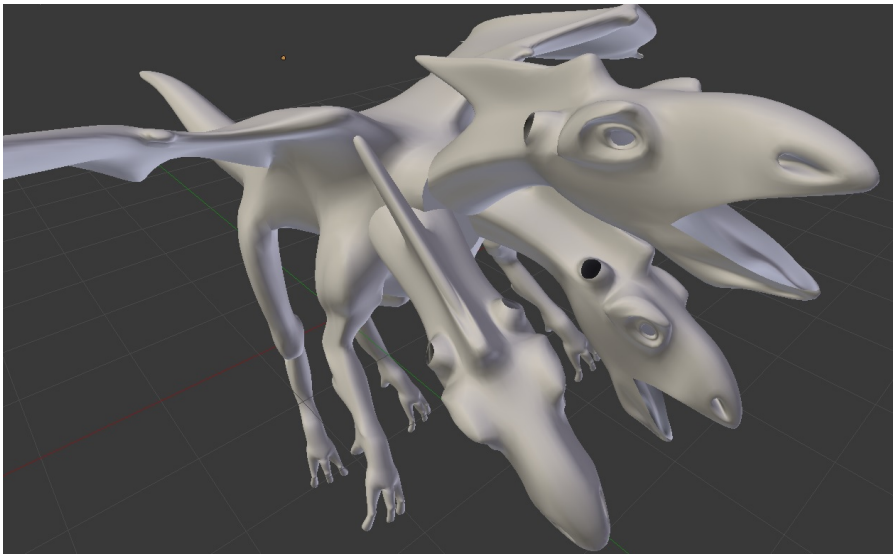


Figure 3.8: The setup of the three-headed dragon with overlapping meshes.

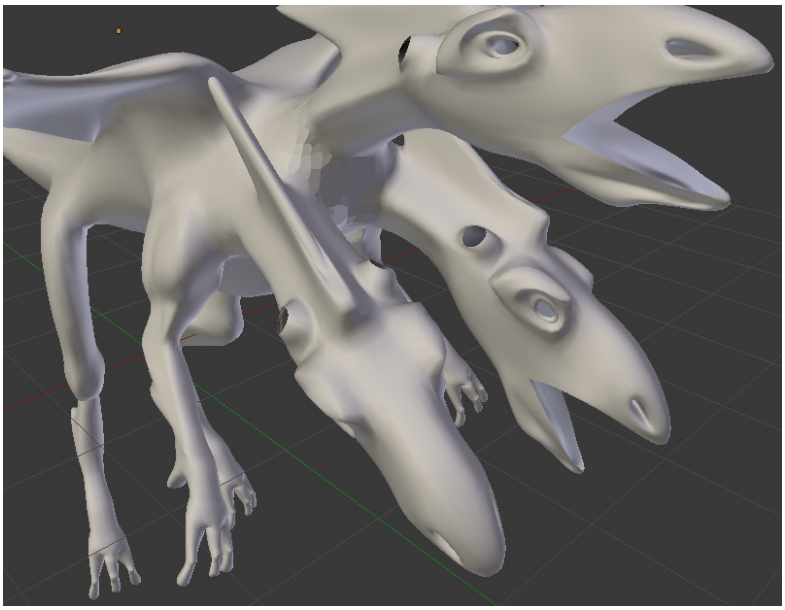


Figure 3.9: The blending of the three-headed dragon done by hand.

the reader. There are some drawbacks and thus the results do not look satisfying. The drawbacks are explained while describing our method. An example is shown in Figure 3.8. In Figure 3.9 we accomplished to do the blending by hand. That is how it should look like.

The user has to specify a starting and ending point for blending. After that the blending space between the 2 points is sampled by creating many planes in regular intervals. For

a better understanding the Figure 3.10 is shown. To obtain the contours the planes are

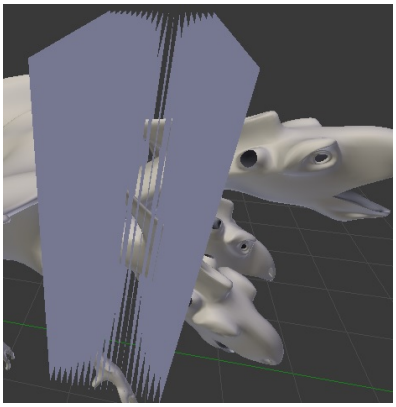


Figure 3.10: The three-headed dragon sliced by many planes. The contours of the intersections with each plane are used for the further blending.

intersected with all objects. In the following sections the body of the dragon is called root and the heads are called objects. There are two major steps which have to be done. One step is, to make the contours of all objects overlap in the mid of the root contour. The second step is, to make a transition from the root contour to the merged contour of all objects.

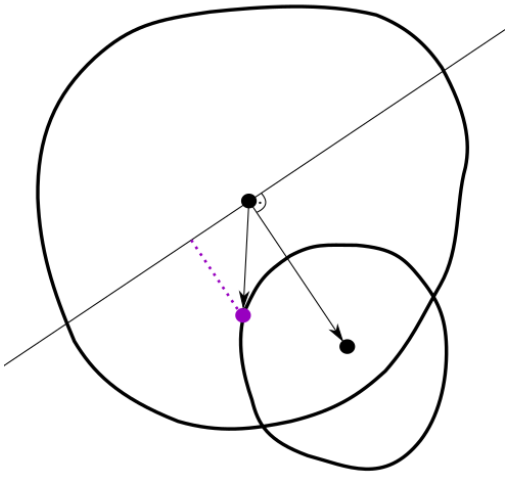


Figure 3.11: Dragging the contours of the object to the mid of the root contour. The purple point is dragged onto the orthogonal plane, which goes through the root mid. The nearer the plane is, the more the point gets dragged to it.

The first step is done for each object separately, which is described in Figure 3.11. The vector between the mid points of the contours is calculated. Then, for each vertex in the

object contour, the distance to the orthogonal plane to the mid-mid-vector is calculated. The vertex in the graphic is marked as a purple dot. The vector from the root mid to the vertex is taken and then simply the dot product with the mid-mid-vector is calculated. This is the distance from the vertex to the orthogonal plane, marked as purple dotted line. The minimal distance is saved and used later again.

For each vertex of the object contour new coordinates are calculated by dragging the vertex to the orthogonal plane. This happens in a Gaussian ratio with the function $\alpha * e^{-gaussiness*(distance-minDistance)^2}$. α should be greater than 1 to ensure, that the contours of all objects are actually overlapping. In this case $\alpha = 1.15$ was used which showed good results. The gaussiness describes how fast the ratio gets to zero. The greater the distance deviates from the minDistance the more it reaches zero. In the described example a gaussiness of 17 was used. The ratio is further multiplied by a ratio, which goes from zero to one, to get a smooth transition surface. The new point for each vertex is then calculated by $oldPoint - midMidVectorNormalized * minDistance * ratio$. A defined slice between the end blend point and the last contour of the root object should look like in Figure 3.12.

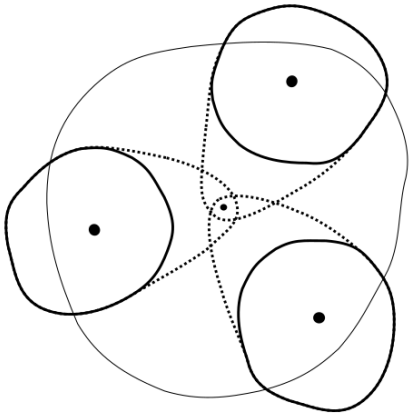


Figure 3.12: All the object contours dragged together to the root mid. The solid lines are the contours of the objects and the dotted lines are the resulting contours after dragging.

The next step calculates the contours from the defined slice to the blend start point as displayed in Figure 3.13. The object contours as well as the root contour are visible. The inner contour, marked as blue, has to be interpolated to the outer contour, marked as red. An unsuccessful attempt was made to get the inner and outer contour and to build up correspondence in a single step by using the angular approach like in Figure 2.3. The main reason why this test failed are the horns of the dragon heads. Trying a different approach, however, would not have been within all reasonable efforts and would have exceeded the extent of this work. The main problem which was faced, was to merge the contours to get the inner and outer contour. The results in Figure 3.14 are not satisfying.

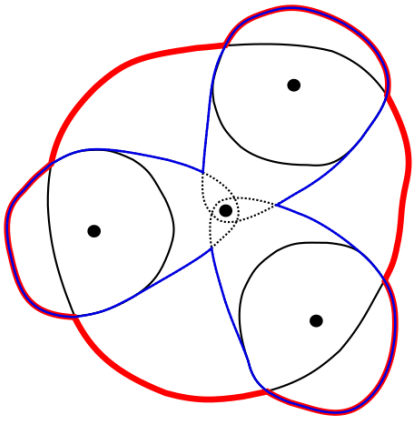


Figure 3.13: The inner and outer contour extracted of the merged dragged contours. The red line indicates the other contour and the blue line the inner contour.

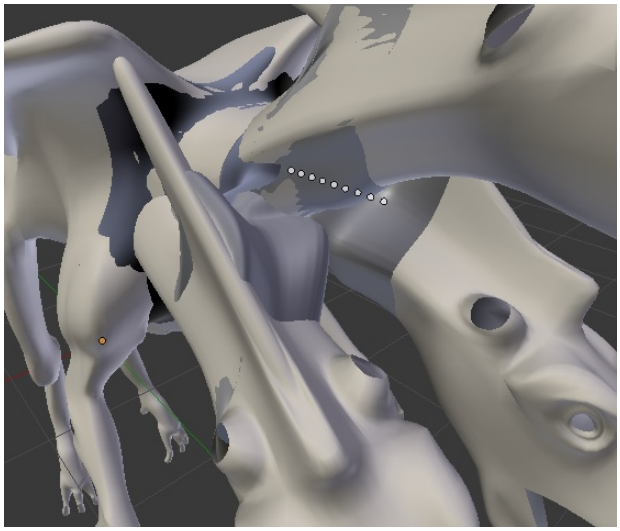


Figure 3.14: The three-headed dragon after applying the algorithm. Some artifact can be seen between the necks on the left side, for example.

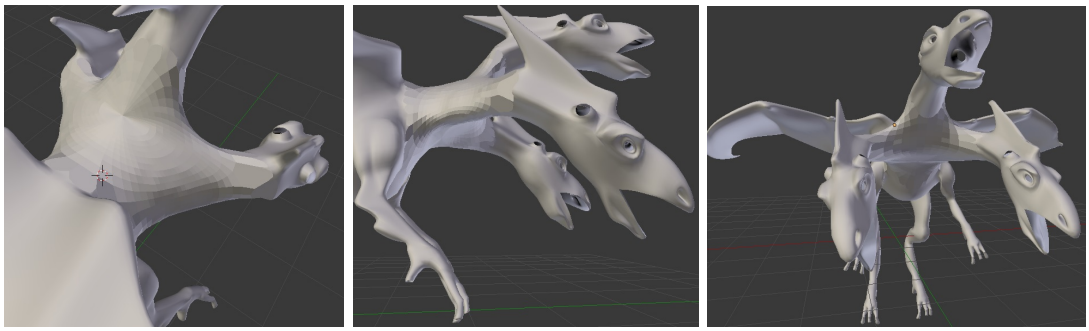
As the described approach is additive, maybe a combination of additive and subtractive methods would show better results.

CHAPTER 4

Results and evaluation

This chapter shows, how much time can be potentially saved with the use of tools. We are evaluating only looking at the left path of the artists workflow from Figure 1.1. We compare the time the artist needs to merge parts of models together by hand and the time the artist needs to merge them with the algorithm, which is mainly the time needed to setup the parameters correctly. Our experiments are all built up as follows: We are cutting and placing parts of models in the scene. This is our initial setup. Then we start to measure the time until the models are merged together. Once we measure the time to merge the models by hand and once the time to merge the models with help of the algorithm.

The first example is the three-headed dragon. It took us 24 minutes and 14 seconds to get an acceptable result by hand, which is shown in Figure 4.1.

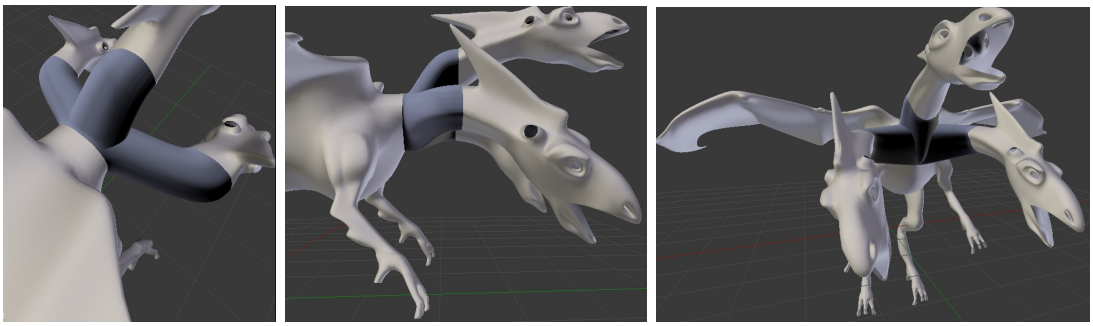


(a) Dragon from the top. (b) Dragon from the side. (c) Dragon from the front.

Figure 4.1: The three-headed dragon connected by hand from different point of views.

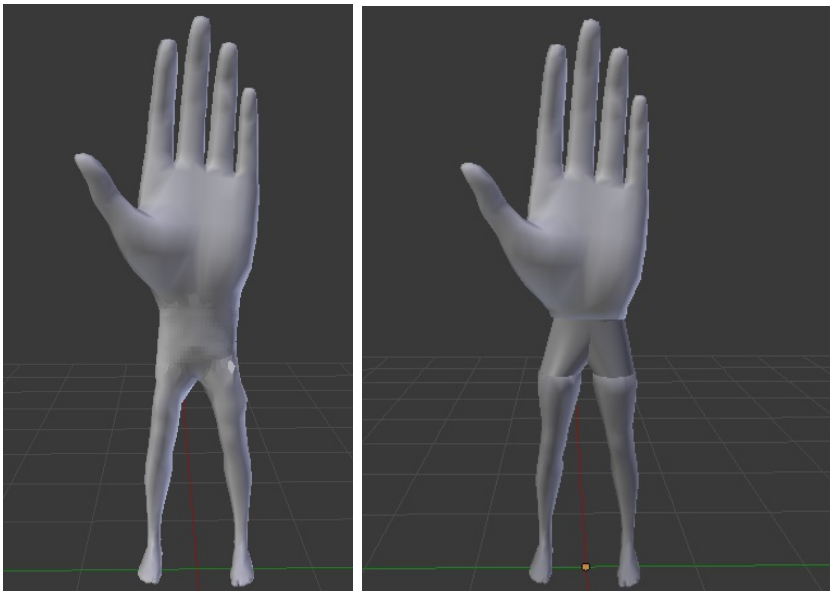
The generated transition only took 1 minute and 37 seconds. This is the time we

needed to specify the blending start and end point. The results are displayed in Figure 4.2. All the examples and the time needed are summarized in the table 4.1.



(a) Dragon from the top. (b) Dragon from the side. (c) Dragon from the front.

Figure 4.2: The three-headed dragon connected with our algorithm from different point of views.



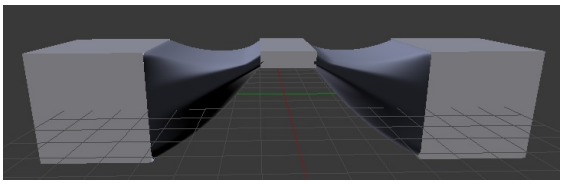
(a) Created by hand. (b) Created by algorithm.

Figure 4.3: The merging result of the walking hand. One time done by hand and one time done by our algorithm.

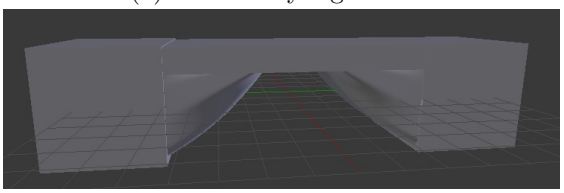
The results show, that the algorithm is a great enhancer of time efficiency, if the results are satisfying enough for the respective purpose. If needed, optimizations are still possible by hand afterwards. In Figure 3.9, the heads are overlapping the body, which took us only 2 minutes and 54 seconds to be merged by hand. Through these quick

retouches the result is improved immediately. Therefore, the use of algorithms does not give great advantages in overlapping cases.

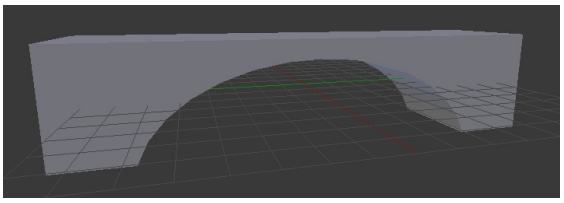
The second example is the walking hand seen in Figure 4.3. The meshes from [fis11] and [Shp15] were used as a basis. It took us 1 minute and 45 seconds with the algorithm and 4 minutes and 59 seconds by hand to finish the process. Accordingly, the smaller the distance between the meshes, the faster the merging by hand.



(a) Created by algorithm.



(b) Created by algorithm and edited by hand.

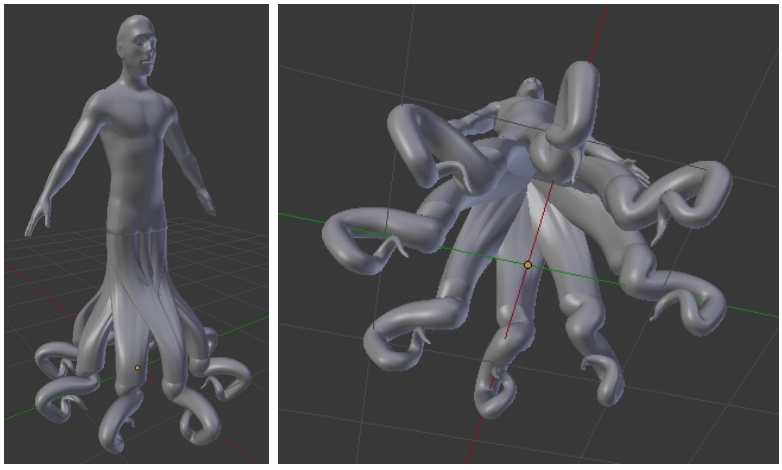


(c) Created completely by hand.

Figure 4.4: Example of the construction of a bridge. The Figure a) shows the result of our algorithm with the auxiliary part and b) how we updated it afterwards. Figure c) shows the result without the use of algorithms.

The next example is the construction of a bridge given 2 boxes on the left and right side. A small piece of mesh was inserted in the middle as auxiliary. Then we ran the introduced algorithm twice with linear transition function x and turn rate function $x/2$. The output is shown in Figure 4.4a. Afterwards a long box on the top of it was added. This process took 3 minutes and 3 seconds in total. Doing everything by hand took us 1 minutes and 19 seconds. The results are displayed in Figure 4.4. Using the algorithm in this case is possible, but not reasonable. A bridge is easily created by subtracting a cylinder from a long box. With this example we want to show, that the use of a tool does not save time by itself. It can be used, where it creates no benefit or even decreases the productivity.

Figure 4.5: Initial Setup of the octopus man.

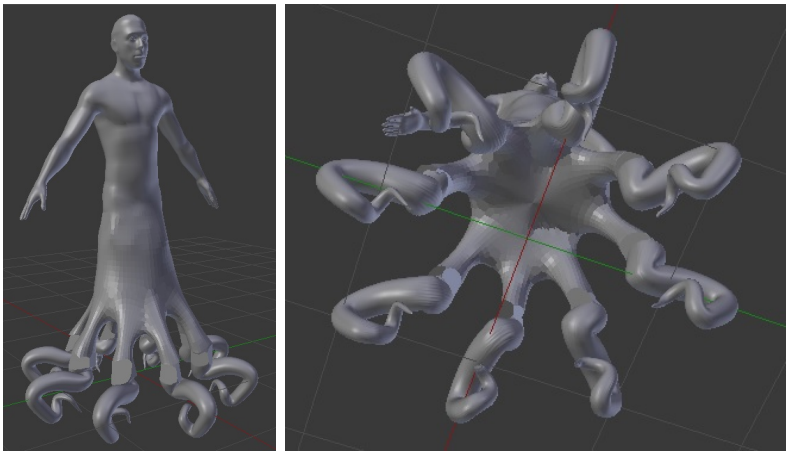


(a) From the front/side. (b) From the bottom.

Figure 4.6: The octopus man created by our algorithm from different point of views.

Furthermore the octopus man is introduced as an example. The setup is shown in Figure 4.5. Using the algorithm it took 1 minute and 44 seconds to get the result seen in Figure 4.6. Connecting everything by hand took us 16 minutes and 24 seconds, shown in Figure 4.7. In this example the time difference is significant. Generally speaking the more objects have to be merged in one single step and the more complex the transition is, the more time efficient is the use of an algorithm. In our opinion the generated transition also looks better than what we have done by hand. In the end this depends on the preferences of the user.

To summarize this section, it has been proven, that a tool can be a huge help for modelers. It has to be kept in mind that this is just another algorithm and no algorithm fits every possible scenario. The usability of the algorithm depends on the intended



(a) From the front/side. (b) From the bottom.

Figure 4.7: The cctopus man created by hand from different point of views.

Example	Time by hand	Time by algorithm
three-headed dragon, non-overlapping	24min 14sec	1min 37sec
three-headed dragon, overlapping	2min 54sec	no good result
the walking hand	4min 59sec	1min 45sec
bridge	1min 19sec	3min 3sec
octopus man	16min 24sec	1min 44sec

Table 4.1: Example list with the time needed to merge the objects by hand and with our algorithm.

results of the user, as stated in the examples. In some cases there is a significant saving of time, whereas in other cases using conventional methods may be the better choice.

CHAPTER 5

Conclusions and future work

The work gives an overview of possible techniques the user can test in specific scenarios. We highlighted the opportunities and limitations of every technique. The blending techniques were categorized and it was summarized what is important, e.g., contours and correspondence. Moreover, we have introduced own implementations with some improvements and different attempts. We believe there is a huge potential for future research in this field. For example, an algorithm could add some features on the transition surface, like bumps, spikes or something else.

The non-overlapping methods have a greater impact on the modeling time, because it takes long to merge the meshes together by hand when there is a greater distance between the objects, compared to cases, where the meshes are overlapping each other. Nevertheless, there are possible advantages of overlapping techniques, e.g., when we want to generate random arbitrary creatures. This would save a huge amount of time for artists as well.

On the whole, there is a huge potential for new tools, which help to fuse meshes together. Currently there is no tool, which is using different techniques and automatically decides which one to use for the current case. This would be a field for future research to create constantly improved tools for artists. But even our basic algorithm can decrease the modeling time by a sufficient amount of time.

On the other side using such tools do not save a single user time by itself. To be more productive it is key to know when to use which method. Choosing a wrong tool leads to less productivity. Knowing the advantages and disadvantages of existing tools and brand new ones and in which cases we should use them, is the key to success.

List of Figures

List of Tables

Bibliography

[AS06] Ariel Shamir Daniel Cohen-Or Andrei Sharf, Marina Blumenkrants. Snappaste: an interactive technique for easy mesh composition. *The Visual Computer*, 22:835–844, 2006.

[BL03] ISA INRIA Lorraine Bruno Lévy. Dual domain extrapolation. *ACM Transactions on Graphics (TOG)*, 22:364–369, 2003.

[BL10] Yang Liu Bruno Lévy. Lp centroidal voronoi tessellation and its applications. *ACM Transactions on Graphics (TOG)*, 29, 2010.

[Dyn08] Dynamyte. Snake.zip. `https://www.turbosquid.com/FullPreview/Index.cfm/ID/398555`, 2008. Accessed: 2018-08-03.

[fis11] fishzombie. Hands and feet. `https://www.turbosquid.com/FullPreview/Index.cfm/ID/587188`, 2011. Accessed: 2018-08-03.

[Fou18] Blender Foundation. Blender. `https://www.blender.org/`, 2018. Accessed: 2018-10-30.

[GT99] James F. O'Brien Greg Turk. Shape transformation using variational implicit functions. *SIGGRAPH '99 Proceedings of the 26th annual conference on Computer graphics and interactive techniques*, 26:335–342, 1999.

[GT05] James F. O'Brien Greg Turk. Modelling with implicit surfaces that interpolate. In *SIGGRAPH '05 ACM SIGGRAPH 2005 Courses*, Los Angeles, California, Jul. 31-Aug. 04 2005.

[Inf08] Infinite3dfx. Free winged dragon gargol. `https://www.turbosquid.com/FullPreview/Index.cfm/ID/407721`, 2008. Accessed: 2018-08-03.

[KM02] Ross T. Whitaker-Alan H. Barr Ken Museth, David E. Breen. Level set surface editing operators. *ACM Transactions on Graphics (TOG)*, 21:330–338, 2002.

[MC15] Alberto Signoroni Marco Centin, Nicola Pezzotti. Poisson-driven seamless completion of triangular meshes. *Computer Aided Geometric Design*, 35-36:42–55, 2015.

[MF09] Solomon Boulos Kurt Akeley-William R. Mark Pat Hanrahan Matthew Fisher, Kayvon Fatahalian. Diagsplit: parallel, crack-free, adaptive tessellation for micropolygon rendering. *ACM Transactions on Graphics (TOG)*, 28, 2009.

[Mor01] Henry Moreton. Watertight tessellation using forward differencing. *HWWS '01 Proceedings of the ACM SIGGRAPH/EUROGRAPHICS workshop on Graphics hardware*, pages 25–32, 2001.

[MT09] Tainchi Lu Minchih Tsai. A rapid mesh fusion method to create 3d virtual characters in games. In *Fourth International Conference on Computer Sciences and Convergence Information Technology*, Seoul, South Korea, Nov. 24-26 2009.

[NC03] Eugenio Oñate Nestor Calvo, Sergio R. Idelsohn. The extended delaunay tessellation. *Engineering Computations*, 20:583–600, 2003.

[Par11] Fred Park. Shape descriptor/feature extraction techniques. `https://pdfs.semanticscholar.org/presentation/5ca6/92b4a78b5c11154b7ddf4d495db0d384a78f.pdf`, 2011. Accessed: 2018-08-03.

[Shp15] Shprott. male mid poly character. `https://www.turbosquid.com/FullPreview/Index.cfm/ID/889072`, 2015. Accessed: 2018-08-03.

[TF04] Philip Shilane Patrick Min-William Kiefer Ayellet Tal Szymon Rusinkiewicz David Dobkin Thomas Funkhouser, Michael Kazhdan. Modeling by example. *ACM Transactions on Graphics (TOG)*, 23:652–663, 2004.

[TK99] Jun Mitani Fumihiko Kimura Takashi Kanai, Hiromasa Suzuki. Interactive mesh fusion based on local 3d metamorphosis. 1999.

[WEL87] Harvey E. Cline William E. Lorensen. Marching cubes: A high resolution 3d surface construction algorithm. *ACM SIGGRAPH Computer Graphics*, 21:163–169, 1987.

[XH07] Oscar Kin-Chung Au-Chiew-Lan Tai Xiaohuang Huang, Hongbo Fu. Optimal boundaries for poisson mesh merging. In *SPM '07 Proceedings of the 2007 ACM symposium on Solid and physical modeling*, Beijing, China, Jun. 04-06 2007.

[XJ06] Charlie C.L. Wang Jieqing Feng-Hanqiu Sun Xiaogang Jin, Juncong Lin. Mesh fusion using functional blending on topologically incompatible sections. *The Visual Computer*, 22:266, 2006.

[XW15] Chaowei Gao Xiaomeng Wei, Li Chen. Automatic mesh fusion for dental crowns and roots in a computer-aided orthodontics system. In *8th International Conference on Biomedical Engineering and Informatics (BMEI)*, Shenyang, China, Oct. 14-16 2015.

[YSL05] Jun-Hai Yong Pi-Qiang Yu Jia-Guang Sun Yu-Shen Liu, Hui Zhang. Mesh blending. *The Visual Computer*, 21:915—-927, 2005.

[YY04] Dong Xu-Xiaohan Shi Hujun Bao Baining Guo-Heung-Yeung Shum Yizhou Yu, Kun Zhou. Mesh editing with poisson-based gradient field manipulation. *ACM Transactions on Graphics (TOG)*, 23:644–651, 2004.