Thesis Title

Floor Verhoeven

Master Thesis
April 2018

Supervisors:
Dr. My Supervisor
Prof. Dr. Olga Sorkine-Hornung

ETH
Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

igl
INTERACTIVE GEOMETRY LAB

Abstract

This thesis addresses the development of a novel sample thesis. We analyze the requirements of a general template, as it can be used with the LaTeX text processing system. (And so on...) The abstract should not exceed half a page in size!

Contents

List of Figures

List of Tables

1

Introduction

Digital 3D shape modelling is a field that has received a lot of attention over the past years as a result of increasing rendering possibilities and a growing call for digital content. The recent surge in interest for VR (Virtual Reality) has driven the development of better GPUs (Graphics Processing Unit) even more, while also further increasing the demand for digital 3D content. The process of 3D shape modelling has so far been almost exclusively suitable for trained professionals or users with a lot of experience. While 3D modelling software generally comes with plenty of options and possibilities it also usually has a steep learning curve. This causes 3D modelling to become inaccessible to novice users who would like to develop some 3D content, but who do not have the time to learn how to use these programs.

Sketch-based 3D modelling seeks to simplify the process of 3D modelling in order to make it more accessible to this user group. Its goal is to provide the user with intuitive ways to interact with the mesh that is being modelled. Previous work regarding sketch-based modelling tools has produced multiple software products which present the user with this intuitive method of 3D modelling.

Recently a variety of art creation tools for usage in VR have been created. These applications range from painting in 3D to voxel modelling and 3D sculpting. VR gives artists the possibility to look at what they're creating from a literally new perspective. Directly modelling in 3D gives a better feeling of the proportions of different parts of a model and for example the angles between them.

insert screensho
existing softwar

The goal of this thesis is to develop a sketch-based 3D modelling program for usage in VR. This combines the intuitivity of sketch-based designing and the emersiveness of modelling in actual 3D space and the possibility of viewing the results immediately in 3D. The software is targeted to user that are new to 3D modelling and should therefore be straightforward to use and easy to learn.

2

Related Work

A lot of software for the purpose of 3D modelling exists, and they are all very different to use. Some of the best known ones are Autodesk Maya, Blender, Autodesk 3ds Max and Zbrush. Typically the user manipulates a mesh on vertex or face level, resulting in very fine-grained control over the appearance. Although this makes these programs very powerful and versatile for creating and editting 3D models, they have a very steep learning curve and are overcomplete for recreational users. This chapter will instead focus on describing some of the more intuitive and easy-to-use sketch-based modelling software, as well as a couple of new artistic tools for creating 3D content in VR.

include citations

Sample references are [Zwicker et al. 2004] and [Altman 1989].

remove sample

2.1. Sketch-based Modelling

Sketch-based modelling is a modelling technique that aims to transfer the way that people draw shapes with pen and paper to a method of modelling 3D shapes on the PC with a mouse. With the mouse the user draws 2D strokes that are meshed and then inflated to rotund 3D meshes. By specifying additional strokes the user can then edit this initial mesh. The software that was written as part of this thesis also adopts the sketch-based modelling paradigm.

2.1.1. FiberMesh

FiberMesh is a system that allows modelling freeform surfaces by drawing 3D curves [**?**]. The user-defined curves are used to create a 3D model and stay present on the model and can be used

to edit the geometry. This allows for a very intuitive method of deforming and editing meshes after their initial creation. FiberMesh lets users define curves as smooth or sharp, add and remove control curves on the mesh and pull curves to deform the mesh. Additionally it allows the user to change the mesh topology by cutting parts of the mesh and creating extrusions or tunnels.

Algorithmically FiberMesh depends on two main steps, namely curve deformation and surface optimization. In addition to these two steps, there are also a mesh construction and remeshing step, which only occur after new mesh topology has been created (e.g. after creation, cut or extrusion).

For curve deformation they used a detail-preserving deformation method that combines differential coordinates [**?**] with co-rotational methods [**?**]. A sequence of linear least-squares problems is solved, while satisfying the user-defined positional constraints on the drawn curves. The rotation matrices are explicitly represented as free variables, as they cannot be derived from the curves which are nearly collinear. In order to accommodate for large linear rotations, the gross rotation is computed by iteratively concatenating small delta rotations that are each obtained by solving a linear system. The linear system that is solved in each step can be seen in equation 2.1.

$$\arg\min_{\mathbf{v},\mathbf{r}} \left\{ \sum_i \|\mathbf{L}(\mathbf{v}_i) - \mathbf{r}_i\mathbf{R}_i\delta_i\|^2 + \sum_{i\in C_1} \left\|\mathbf{v}_i - \mathbf{v}_i^{'}\right\|^2 + \sum_{i,j\in E} \|\mathbf{r}_i\mathbf{R}_i - \mathbf{r}_j\mathbf{R}_j\|_F^2 + \sum_{i\in C_2} \left\|\mathbf{r}_i\mathbf{R}_i - \mathbf{R}_i^{'}\right\|_F^2 \right\} \tag{2.1}$$

Here $\mathbf{L}(\cdot)$ is the differential operator, $\mathbf{v}_i$ is the coordinates of vertex i, $\|\cdot\|_F$ is the Frobenius norm, E is the set of curve edges and C_1 and C_2 are the sets of constrained vertices, primed values are constraints, $\mathbf{R}_i$ represents the gross rotation (in the deformed curve) corresponding to vertex i obtained from the previous iteration, and finally $\mathbf{r}_i$ is a linearized incremental rotation for vertex i given by a skew symmetric matrix with 3 unknowns

$$\mathbf{r}_i = \begin{bmatrix} 1 & -r_{iz} & r_{iy} \\ r_{iz} & 1 & -r_{ix} \\ -r_{iy} & r_{ix} & 1 \end{bmatrix}.$$

Rotations are updated by setting them to $\mathbf{R}_i = \mathbf{r}_i\mathbf{R}_i$ and orthonormalizing the result using polar decomposition [**?**]. In the iterative process of estimating rotations, they use first order differentials (L_0), and for computing the final vertex positions using this estimated rotations they use the second order differential (L_1).

$$L_0 = \mathbf{v}_i - \mathbf{v}_{i-1}, \quad L_1 = \mathbf{v}_i - \frac{1}{|N_i|}\sum_{j\in N_i} \mathbf{v}_j.$$

For surface optimization, FiberMesh solves 3 sparse linear systems in order to compute a smooth mesh surface that adheres to the user-defined constraint curves. The first system solves

for a set of smoothly varying Laplacian magnitudes $\{c_i\}$ which approximate the scalar mean curvature values. The least-squares minimization that it solves is as follows:

$$\arg\min_{c} \left\{ \sum_i \|\mathbf{L}(c_i)\|^2 + \sum_i \left\| c_i - c_i^{'} \right\|^2 \right\} \tag{2.2}$$

Where $\mathbf{L}(\cdot)$ denotes the discrete graph Laplacian, which is independent of the exact mesh geometry, allowing us to reuse it in multiple iterations. In the first iteration, the target Laplacian magnitudes are only set for the constrained curves by using the scalar mean curvature along the curve.

To obtain a geometry that satisfies these target Laplacian magnitudes, Nealen et al. use the uniform Laplacian as an estimator of the integrated mean curvature normal, which is computed by $\delta_i = A_i \cdot c_i \cdot \mathbf{n}_i$, where A_i is an area estimate for vertex i, c_i is the target Laplacian magnitude and $\mathbf{n}_i$ is the estimated normal from the current face normals. However the uniform Laplacian is not a accurate estimator of the integrated mean curvature normal when the incident edges to a vertex are not of equal length. To solve for this problem without using a geometry dependent discretization and thus avoiding recomputing the matrix for every iteration, they prescribe target edge vectors in an attempt to achieve equal edge length. This is done by solving the following linear system (which uses the same matrix as the system that solves for target Laplacian magnitudes) to obtain a smooth set of target average edge lengths e_i:

$$\arg\min_{e} \left\{ \sum_i \|\mathbf{L}(e_i)\|^2 + \sum_i \left\| e_i - e_i^{'} \right\|^2 \right\} \tag{2.3}$$

Again the first iteration is performed with only the edge lengths along the constrained curve. The target average edge lengths are then used to compute target edge vectors for a subset B of mesh edges (in the first iteration this subset only contains edges along the constrained curves, and afterwards it contains all edges incident to the constrained curves) as follows:

$$\eta_{ij} = (e_i + e_j)/2 \cdot (\mathbf{v}_i - \mathbf{v}_j) / \|\mathbf{v}_i - \mathbf{v}_j\| . \tag{2.4}$$

These target edge vectors are then used to solve a linear system that gives the updated vertex positions as follows:

$$\arg\min_{\mathbf{v}} \left\{ \sum_i \|\mathbf{L}(\mathbf{v}_i) - \delta_i\|^2 + \sum_{i \in C} \left\| \mathbf{v}_i - \mathbf{v}_i^{'} \right\|^2 + \sum_{(i,j) \in B} \|\mathbf{v}_i - \mathbf{v}_j - \eta_{ij}\|^2 \right\} \tag{2.5}$$

The systems for solving for target Laplacian magnitudes, average edge lengths and optimal vertex positions are solved iteratively until convergence (approximately 5-10 iterations). Since only a geometry independent discretization of the Laplacian is used, the system matrix can be reused between iterations, until the mesh topology is changed (for example by a cutting action). Because of this, the slow matrix factorization only needs to happen once for a given mesh topology, resulting in a fast algorithm that allows for interactive rates.

2.2. 3D Modelling in Virtual Reality

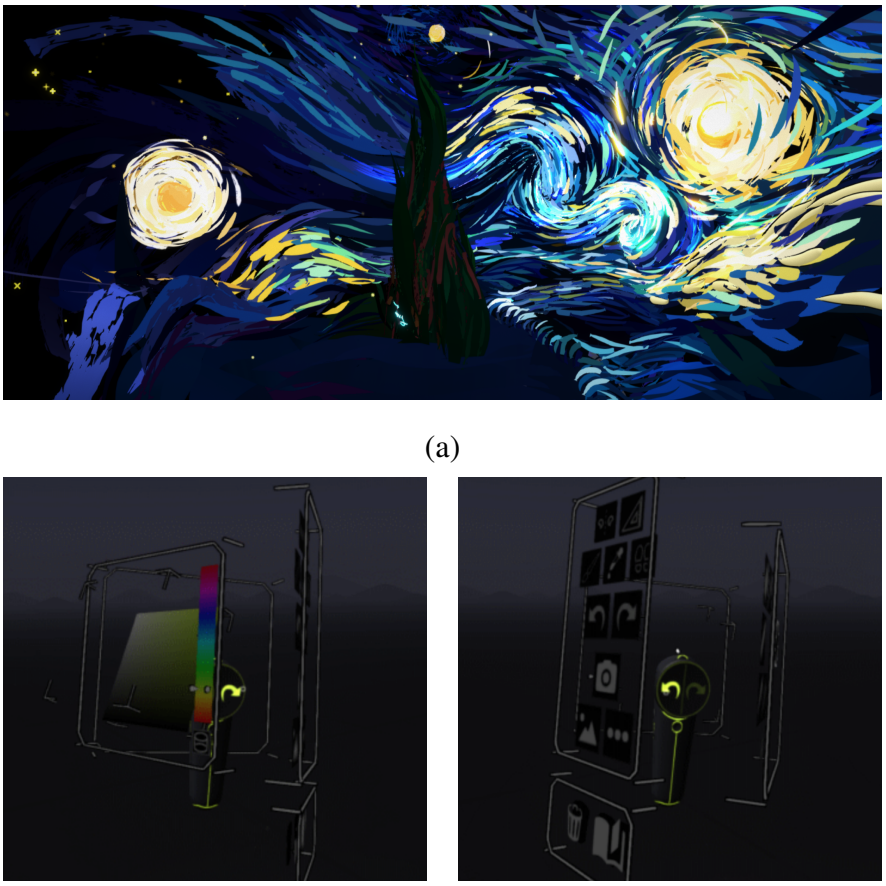
Over the last couple of years, plenty of VR applications have been published that involve creating some type of digital 3D content. The software can roughly be split into 2 categories, namely for creating 3D paintings and for creating 3D models (in many different ways). This section will described a variety of 3D virtual reality applications, their possibilities and interfaces.

2.2.1. Google Tilt Brush

Google's Tilt Brush is available for Oculus Rift and HTC Vive and allows the user to create room-scale art in the style of 3D paintings. The software provides different types of tools and brushes, including special effects tools like a fire or stars brush. Users cannot export their creations to typical model formats such as .obj or .off, but they can turn them into a GIF or upload them to Google Poly where other users can explore their work. Figure 2.1 (a) shows a screenshot of example 3D artwork that has been made with Google Tilt Brush (the original work has stars that change their light intensity). In order to give the user access to the large set of different painting tools, Tilt Brush has created an embedded menu in the form of painters palette that is attached to the controller in the user's non-dominant hand. Several of the menus allow for browsing through further submenus. The dominant hand is used to select a tool and paint with it. Figures 2.1 (b) and 2.1 (c) show two examples of what the hand-held palette menus looks like.

2.2.2. Oculus Medium

Oculus Medium is a 3D sculpting tool that is only available for the Oculus. Conceptually the user can paint with volumetric brushes, much like sculpting with clay. For example the user will draw a circle and this will result in a donut shape. With several tools the user can sculpt away or add extra clay, as well as assigning multiple colors to the surface. Unlike Google Tilt Brush, Oculus Medium does allow the user to store created models in .obj format, so they can be exported for use in other programs. The interface is quite intuitive since it shows a very strong connection to the method of creating 3D shapes with the use of clay in real life. Users can quickly create rough shapes and refine them by adapting the brush size and adding small-scale details. The dominant hand of the user serves as the so-called "Tool hand" which can be used to sculpt with, while the non-dominant hand serves as the "Support hand" which can be used to open up several hand-held menus. Figures 2.2 (a) and 2.2 (b) show examples of complicated 3D models that have been made in Oculus Medium. One of the unique features of Oculus Medium is the functionality to create and use stamps, which helps easily create interesting textures on the models like the fur and grass in Figure 2.2 (a).



(a) (b) (c)

Figure 2.1.: *Google Tilt Brush.* (a) *Recreation of Van Gogh's Starry Night (moving version available at https://poly.google.com/view/e-Zqenw7Dui).* (b) *Left-side view of the "hand-held" user interface of Google Tilt Brush.* (c) *Right-side view of the "hand-held" user interface of Google Tilt Brush.*

2.2.3. MasterpieceVR

MasterpieceVR brings a combination of the design paradigms used by Oculus Medium and Google Tilt Brush, letting the user mix volume sculpting with 3D painting. It is available for Oculus, HTC Vive and Windows Mixed Reality, and for each of them it requires the accompanying controllers for input. One of MasterpieceVR's biggest selling points is that it allows multiple users to work on one model simultaneously, allowing interactive collaboration with direct feedback between artists. Like Oculus Medium, MasterpieceVR lets the user export models (including vertex colors) to .obj, as well as .fbx format. Additionally, users can load reference images or models into the program, which can greatly simplify modelling. MasterpieceVR implements menus in a similar fashion as Google Tilt Brush and Oculus Medium, with a menu of icons held in the non-dominant hand. Compared to these two, MasterpieceVR however has a larger and less intuitive menu layout. Figure 2.3 (a) shows a complex model created in MasterpieceVR and Figure 2.3 (b) shows the menu interface.

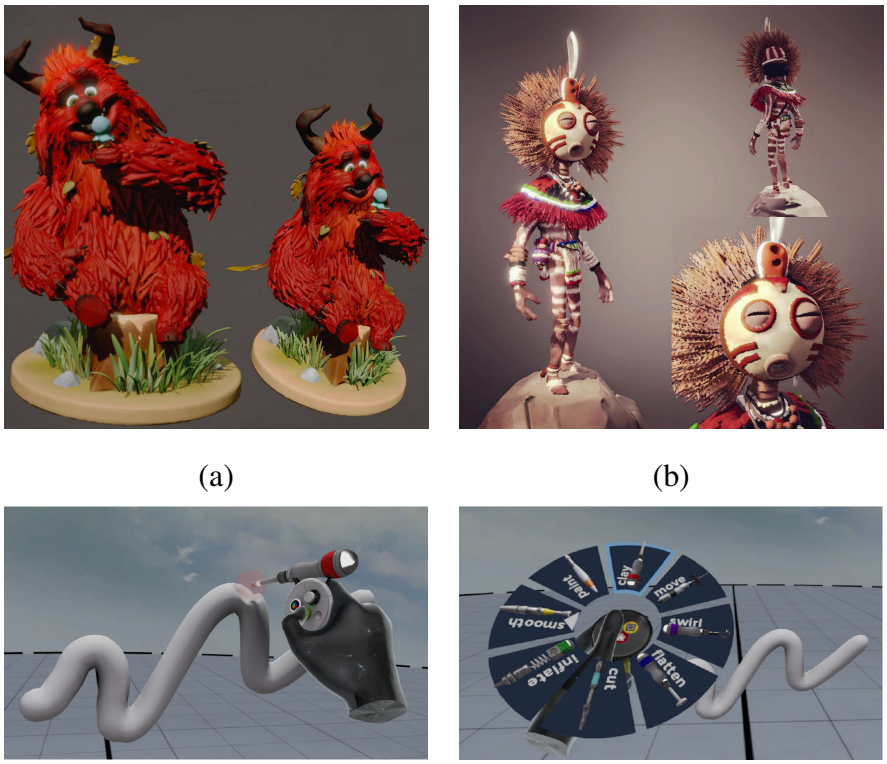


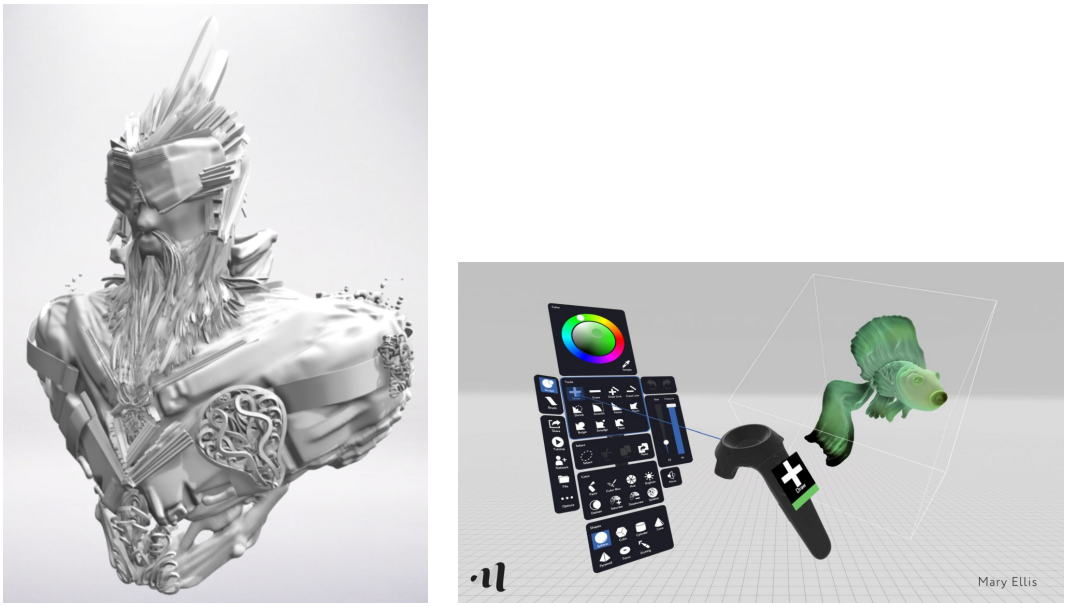
Figure 2.2.: *Oculus Medium.* (a) *Model of a furry monster created in Oculus Medium by artist Goro Fujita.* (b) *Model of an African figure created in Oculus Medium by artist Goro Fujita.* (c) *Sculpting view in Oculus Medium.* (d) *Menu view in Oculus Medium.*

2.2.4. Google Blocks

Google Blocks is available for Oculus abd HTC Vive and is Google's software for creating 3D objects in VR. From the programs mentioned in this section, Google Blocks seems the most similar to professional non-VR based modelling software like Blender or Maya. Users can start with several predefined shapes like spheres, cubes or cones and can edit the models on face, edge or vertex level. While this gives the user a lot of artistic freedom and extra modelling possibilities, it also makes its usage a lot less intuitive then the other VR 3D modelling software. Although the user has low-level control, the created meshes are very low-poly compared to the models created with for example Oculus Medium. The user does not have control over the number of polygons, resulting in less realistic models. Again the created models can be exported as .obj file. Figures 2.4 (a) and 2.4 (b) show a simple model of an anglerfish created in Google Blocks, and the simple Google Blocks interface.

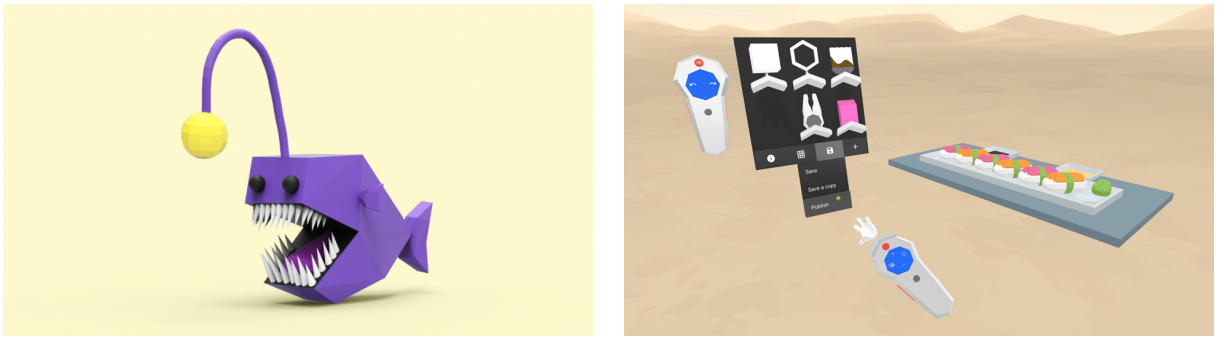
2.2.5. Makebox

remove this,
lly fit in with



(a) (b)

Figure 2.3.: *MasterpieceVR.* (a) *Model of a warrior created in MasterpieceVR by artist Vladimir Ilic.* (b) *Interface of MasterpieceVR (model by Mary Ellis).*



(a) (b)

Figure 2.4.: *Google Blocks.* (a) *Model of an anglerfish created in Google Blocks.* (b) *Interface of Google Blocks.*

3

System Description

This chapter will give a description of the VR sketch-based modelling software, named SketchMeshVR, that was developed as part of this thesis. The software aims to provide users with a simple and intuitive sketch-based 3D modelling tool within a VR environment. Specifically the software is developed for use with the Oculus Rift including one Touch controller.

3.1. Algorithms

Algorithmically, SketchMeshVR largely follows FiberMesh by [**?**]. Similar to FiberMesh, users can draw strokes that define the 3D mesh surface, and these drawn strokes stay on the model surface to later function as deformation handles. Users can create an initial mesh, cut parts of the mesh, create extrusions, add and remove additional control curves and deform control curves on the mesh. Unlike in FiberMesh, SketchMeshVR does not give users the option to choose between smooth and sharp constraint curves and instead all curves are treated as smooth. The functionality to create tunnels through meshes is also not present in SketchMeshVR.

The system is multithreaded such that the VR display keeps on being updated while the mesh computations are being executed. When multithreading is not enabled the screen will freeze during the mesh computations, and therefore will stop updating when the user moves his head, breaking immersion and likely inducing motion sickness.

Due to the direct availability of the actual 3D coordinates, some changes were made in comparison to the non-VR version. Whereas the non-VR version in drawing mode starts by converting mouse positions to screen coordinates and then unprojects these to 3D coordinates with a z-value of 0, SketchMeshVR takes the actual 3D coordinates and only projects them to 2D to allow triangulation. In the non-VR case, adding control curves on the mesh and later removing

them is done by unprojecting the screen coordinates onto the mesh in order to get 3D positions and decide where the user wanted to add/remove a curve. In VR, instead of unprojecting the screen coordinates of the hand or taking the direct hand coordinates, we cast a ray from the hand position in the direction that the controller is being held. This intersection between this ray and the mesh is then used as the final 3D position for adding/removal. When cutting in the non-VR program, the drawn stroke is interpreted purely in 2D. In order to create a loop on the front and back of the mesh surface, it is checked what edges are crossed (in 2D) by a line segment between two consecutive stroke points. Since the same 2D points are used for creating a surface path on the backside of the mesh, this results in perpendicular cuts (meaning that the "cutting knife" always goes perpendicular through the screen, and never at an angle). On the other hand, when cutting in VR we take the coordinates of both the first and second intersection between the cutting ray and the mesh. By doing this, we enable the possibility for the user to cut the mesh diagonally. In order to check which edges are crossed by the stroke segments, we find intersections between edges and the plane that is formed by the two points that make up the stroke segment and the position of the hand at the time of drawing that stroke segment. If an edge is intersected within its range, we know that the stroke segment crosses that edge. The surface path for the extrusion base is created in a similar fashion, except that only the first hit between ray and mesh is used. One problem that arises due to the fact that we're using 3D intersection points between the cutting ray and the mesh, is that we cannot easily derive the start and end points of the cutting stroke. These points are the final point outside of the mesh before we start drawing on top of the mesh and the first point outside of the mesh after drawing on top of the mesh respectively. We need these points in order to be able to find the mesh boundary edges that we need to cross to wrap the surface path around to the backside of the mesh. As mentioned before, in the non-VR case we simply use 2D coordinates based off the screen coordinates of the mouse pointer. In VR we cannot take the position of the controller as the user is likely cutting from a distance. What we've done in order to determine the start and end point is storing the controller position and direction for every sampled point that does not intersect the mesh. Then when the user releases the controller buttons after drawing the cut stroke, we take these positions and directions and along these rays find the two points that are closest to the first and last point drawn onto the mesh. Under normal circumstances this will then give the two points outside of the mesh that are closest to the first and final cutting stroke points on the mesh. This method does however somewhat restrict the user in their freedom on how to draw the cut stroke. If the user holds his hand close to the mesh when starting the cut stroke, it is possible that the closest point to the initial on-mesh point actually is projected onto the mesh. This will make it impossible to create a looped surface path over the mesh. If this happens, a beep will sound and the cut stroke will be shown in black, allowing the user to try again.

Drawing the extrusion silhouette in VR is significantly different from the way it is done in non-VR. When defining the silhouette stroke in non-VR, the user first has to rotate the mesh such that it is looked at from a side view. Only then can the user sensefully specify the shape and depth of the extrusion silhouette, since we cannot gain any depth information from a frontal view. In VR on the other hand, we know the actual 3D positions of the controller and can therefore directly specify the extrusion silhouette in the frontal view. If the user prefers to do this from a side view instead (to get additional visual feedback about the silhouette depth on top of the tactile feedback), this is also possible. Finally

3.2. User Interface

For the user interface in virtual reality it was important to keep the controls for all actions very intuitive. Unlike the VR 3D modelling programs that were discussed in the previous work section, SketchMeshVR does not have any menus in its interface. Instead all functionality in the program can be used with just the right Touch controller (in future versions the user should be able to choose whether to use the left or right controller as the primary one). To simplify orientating the user's hand in the scene, a sphere is displayed at the location of the right Touch controller, together with a ray from the hand following the orientation of the controller. When the user is in draw or pull mode or is drawing the extrusion silhouette, the ray disappears and only the hand sphere shows. This is done in order to emphasize that the actual position of the hand is used to draw a stroke, instead of the intersection points of the ray and mesh.

put a ref to the section here

Compared to drawing strokes with a mouse on a PC screen (which happens purely in a 2D plane), drawing strokes in virtual reality with the Oculus Touch controller allows the user to draw strokes directly in 3D. In SketchMeshVR we made use of this advantage whenever possible. However, when the user draws the stroke that is used to create the initial mesh, the software does assume that the stroke is drawn mostly on the plane that the user is looking at. This is due to the fact that the drawn points will be projected to this 2D plane before they are triangulated. Subsequent cutting or extrusion operations do not have to be made in the plane that is being viewed, as for these actions the actual 3D positions of the drawn points are used.

TEST DRAWIN IN YZ PLANE (BUT NOT FUL PERPENDICUL

To create an initial mesh, the user has to simultaneously press and hold the grip and trigger buttons while drawing a stroke in the air and subsequently release both when the stroke is finished. If the drawn stroke results in a non-edge manifold mesh (for example if the stroke intersects itself), a beep will sound and the stroke is displayed in black. Otherwise the created mesh will be shown, with the originally drawn stroke overlayed.

position touch c image

In order to deform the mesh, a user can drag on all present mesh curves. The user can toggle between drawing and deformation mode by pressing the B button. Upon startup draw mode is selected. To perform the dragging, the user has to simultaneously press and hold down the grip and trigger buttons while his hand is close to the curve vertex that he wants to pull to a new position. Then while still pressing the grip and trigger buttons, the user needs to move his hand to the desired new position, and release both buttons when the desired result is achieved. The new vertex position is interpreted directly as the 3D position of the Touch controller, therefore also allowing deformation outside of the viewing plane (as compared to the non-VR version, which only allows in-plane deformation).

The combination of grip and trigger buttons was chosen for these actions as they most resemble the feeling of holding a pencil and grabbing something in order to pull on it.

To add extra control curves to an existing mesh, the user has to press and hold the trigger button while drawing the stroke onto the mesh. All points that are outside of the mesh will be ignored and will not be added to the control curve.

In order to remove control curves (all except for the curve that created the initial mesh), the user can toggle from stroke adding mode to stroke removal mode by pressing down the right thumbstick. Upon startup the stroke adding mode is selected.

say something a mesh intersectio add and remove

We decided to map the trigger button to the functions of adding and removing control curves since this feels like a natural way to select objects (removal) and to highlight areas (addition).

Finally with the B button the user can switch between cut and extrusion modes. Upon startup the toggle is set to cutting mode, which allows the user to cut off parts of the existing mesh. To make a cut the user has to first point the laser ray to somewhere outside of the mesh, and then while holding down the grip button draw the cutting stroke over the mesh and finally releasing somewhere outside of the mesh again. In order to minimize unwanted behaviour, the user's hand should be a distance of approximately the mesh diagonal away from the mesh when performing a cut action. If anything went wrong during the cutting procedure, the drawn stroke will show on the mesh in black and a beep will sound. When in extrusion mode, the user has to start on the mesh and then while pressing the grip button has to draw the extrusion base on the mesh and release the button when the stroke is complete. Just like in the control curve addition mode, any points that are drawn outside of the mesh will be ignored. To ensure a correct functioning of the algorithm, the drawn base stroke has to have at least one vertex on its interior. Then to complete the extrusion, the user has to draw the extrusion silhouette stroke that will define the height of the extrusion. When drawing this stroke, the user has to press and hold the grip button again but this time the actual position of the Touch controller will be used instead of the ray-mesh intersection point. Releasing the grip button will start the extrusion computations.

When pressing only the grip button, the hand often naturally assumes a position that resembles the way we symbolize pistols with our hands. Intuitively this maps to the concept of shooting a laser ray from your hand, which would be a sensible tool to cut a mesh with. For the case of extrusion we can argue something similar, as part of the mesh topology is also cut away. For this reason the grip button was mapped to the cut and extrude modes.

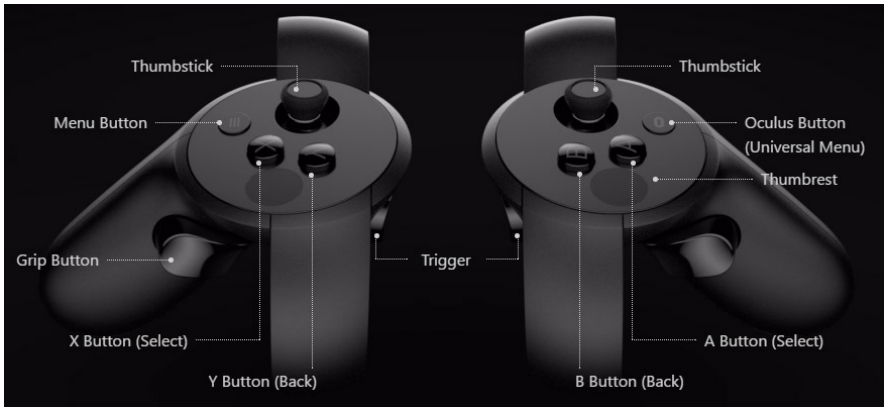


Figure 3.1.: *Button layout of the Oculus Touch controllers.*

Quant.	*Ingredient*
200g	Weißmehl
1/4	Packung Frischhefe
4EL	lauwarme Milch
4EL	Öl
1TL	Zucker
1TL	Salz
	lauwarmes Wasser

Table 3.1.: *Flammkuchenteig. The ingredients have to be carefully chosen.*

4

Results

4.1. Interface

add images of in

4.2. User review

write what users
experienced and
some created m

5

Conclusion and Outlook

5.1. Future work

The developed software offers a lot of potential for future expansion and improvements. The software misses some of the useful tools that are present in other modelling software such as simple predefined shapes (like cubes or cylinders), mirroring and merging meshes. Using a second VR controller (e.g. the Oculus Touch controller) gives enough buttons to map these functions to, allowing us to implement them without the addition of menu, thus staying with the principle of "intuitive" hand gestures. Adding an extra controller also allows for the user to switch between smooth and sharp curve deformation. The functionality for this is embedded in the software, but is not mapped to any button because all buttons of the first controller are already occupied by other functions.

Another functionality that would be useful in many cases is the the possibility to use blueprint images. Letting the user define multiple images, taken from different angles, of the object they want to model, allows them to trace these different silhouettes and precisely recreate the desired object.

adapt this senter something reada

Additionally, the quality of the created meshes could greatly be improved by applying intermediate remeshing. As can be seen from the mesh examples, the triangle sizes differ greatly between triangles that are part of the initial created shape and triangles that are part of a subsequent cut surface or extruded part. This makes the appearance of the mesh rather chaotic and this could be avoided by remeshing every time a new surface is created. The purpose of the remeshing would be to make the edge lengths more uniform across the different parts of the mesh, resulting in a much more homogeneous mesh structure.

Final possibilities for improvements to the software lie in improving the interface. Implementing

hand avatars for hand orientation instead of a simple sphere greatly increase the feeling of immersion. Additionally displaying the currently selected tool modes (for example cutting versus extrusion) and enabling the user to navigate the mesh with hand gestures or the controllers are valuable additions that will improve the usability.

5.2. Conclusion

A

Information For The Few (Appendix)

A.1. Foo Bar Baz

A.2. Barontes

A.3. A Long Table with Booktabs

Table A.1.: *A sample list of words.*

ID	Word	Word Length	WD	ETL	PTL	WDplus
1	Eis	3	4	0.42	1.83	0.19
2	Mai	3	5	0.49	1.92	0.19
3	Art	3	5	0.27	1.67	0.14
4	Uhr	3	5	0.57	1.87	0.36
5	Rat	3	5	0.36	1.71	0.14
6	weit	4	6	0.21	1.65	0.25
7	eins	4	6	0.38	1.79	0.26
8	Wort	4	6	0.30	1.62	0.20
9	Wolf	4	6	0.18	1.54	0.19
10	Wald	4	6	0.31	1.63	0.19
11	Amt	3	6	0.30	1.67	0.14

continued on next page

Table A.1.: *(Continued)*

ID	Word	Word Length	WD	ETL	PTL	WDplus
12	Wahl	4	7	0.36	1.77	0.42
13	Volk	4	7	0.45	1.81	0.20
14	Ziel	4	7	0.48	1.78	0.42
15	vier	4	7	0.38	1.81	0.42
16	Kreis	5	7	0.26	1.62	0.33
17	Preis	5	7	0.28	1.51	0.33
18	Re-de	4	7	0.22	1.56	0.33
19	Saal	4	7	0.75	2.10	0.43
20	voll	4	7	0.48	1.82	0.24
21	weiss	5	7	0.21	1.59	0.36
22	Är-ger	5	7	1.16	2.69	0.59
23	bald	4	7	0.18	1.56	0.19
24	hier	4	7	0.40	1.70	0.43
25	neun	4	7	0.17	1.52	0.26
26	sehr	4	7	0.36	1.85	0.43
27	Jahr	4	7	0.50	1.82	0.43
28	Gold	4	7	0.04	1.35	0.20
29	Tä-ter	5	8	0.15	1.39	0.59
30	Tei-le	5	8	0.30	1.71	0.46
31	Na-tur	5	8	0.18	1.59	0.41
32	Feu-er	5	8	0.30	1.71	0.45
33	Rol-le	5	8	0.15	1.46	0.45
34	Rock	4	8	0.29	1.68	0.25
35	Spass	5	8	0.28	1.64	0.32
36	Gäs-te	5	8	0.49	1.75	0.66
37	En-de	4	8	0.36	1.72	0.33
38	Kunst	5	8	0.26	1.59	0.35
39	Li-nie	5	8	0.45	1.88	0.63
40	Bäu-me	5	8	0.48	1.92	0.45
41	Büh-ne	5	9	0.94	2.48	0.62
42	Bahn	4	9	0.21	1.62	0.42
43	Bür-ger	6	9	0.38	1.70	0.65
44	Druck	5	9	0.60	2.03	0.31
45	zehn	4	9	0.41	1.84	0.42
46	Va-ter	5	9	0.36	1.78	0.40
47	Angst	5	9	0.29	1.56	0.35
48	lei-der	6	9	0.13	1.47	0.52
49	häu-fig	6	9	0.82	2.31	0.52
50	le-ben	5	9	0.38	1.85	0.40
51	aus-ser	6	9	1.20	2.26	0.57
52	be-vor	5	9	1.28	2.75	0.39

continued on next page

Table A.1.: *(Continued)*

ID	Word	Word Length	WD	ETL	PTL	WDplus
53	Kai-ser	6	9	0.92	2.37	0.53
54	Markt	5	9	0.23	1.58	0.28
55	Os-ten	5	9	0.21	1.54	0.48
56	Krieg	5	9	0.33	1.67	0.50
57	Mann	4	9	0.31	1.47	0.25
58	Hal-le	5	9	0.24	1.65	0.45
59	heu-te	5	9	0.44	1.87	0.46
60	in-nen	5	10	0.36	1.80	0.45
61	Na-men	5	10	0.28	1.72	0.41
62	jetzt	5	10	0.70	2.07	0.32
63	kei-ner	6	10	0.28	1.62	0.53
64	Schu-le	6	10	1.02	2.12	0.48
65	Ar-beit	6	10	0.34	1.70	0.52
66	An-teil	6	10	0.27	1.63	0.53
67	di-rekt	6	10	0.67	2.04	0.47
68	vor-her	6	10	0.78	2.25	0.47
69	wol-len	6	10	0.44	1.85	0.51
70	Kampf	5	10	0.70	1.96	0.27
71	än-dern	6	10	1.18	2.62	0.65
72	lau-fen	6	10	0.21	1.64	0.52
73	Eu-ro-pa	6	10	0.23	1.53	0.66
74	statt	5	10	1.61	2.86	0.39
75	Wes-ten	6	10	0.29	1.60	0.54

Bibliography

ALTMAN, S. L. 1989. Hamilton, grassmann, rodrigues, and the quaternion scandal—what went wrong with one of the major mathematical discoveries of the nineteenth century? *A Mathematical Association of America journal* (dec).

ZWICKER, M., RÄSÄNEN, J., BOTSCH, M., DACHSBACHER, C., AND PAULY, M. 2004. Perspective accurate splatting. In *Proceedings of Graphics Interface*, 247–254.