FINE CONTROL OF 3D MESHES IN TILT BRUSH THROUGH SCULPTING

CHINGIZ DADASHOV-KHANDAN

rijksuniversiteit groningen

Supervised by: Prof. Dr. Jiří Kosinka, Dr. Cara Tursun

BSc. Computing Science
Scientific Visualisation and Computer Graphics
Faculty of Science & Engineering
University of Groningen

July 2022

ABSTRACT

Tilt Brush, a Virtual Reality sketching application made in the Unity game engine, has a type of brushes called Hull brushes. These brushes create convex hulls that surround the path left by a user's gestures. According to a user evaluation by a professional graphic designer, this set of brushes is imprecise and unintuitive, so users are forced to repeatedly re-draw strokes until they are satisfied. This causes interruptions and delays in the creative process, compounded with general annoyance and discomfort.

This thesis project proposes to solve this issue by introducing an open-source digital sculpting toolkit to Tilt Brush, which allows users to retroactively modify these brush strokes, significantly reducing the need to redraw them.

Following the creation of a sculpting toolkit with four different variations and a scalable design, the subsequent user evaluation demonstrated a noticeable increase in precision and general variety of possible creations when using hull brushes, with room for improvement. As a result, this project has created a foundation for retroactive brush stroke editing in Tilt Brush, which can be further expanded in the future.

CONTENTS

1 INTRODUCTION

Virtual Reality (VR) is a new, direct form of experiencing and interacting with digital media, which has become more and more accessible over the years. Since the mid 2010s, VR technology has been experiencing one of its first big leaps into the consumer market, which has transformed its general impression from something far-fetched and experimental, to something novel and rising in popularity.

Most contemporary designs of VR devices consist of a head-mounted display and a controller for each hand[1]. With this configuration, users are able to observe and interact with virtual environments, all with an unprecedented level of immersion [12]. This form of interaction has led to the conceptualization of many uses and applications, such as entertainment, training, education [2], and media creation. This thesis focuses primarily on the last mentioned application, specifically with VR artistic tools.

VR's three-dimensional interaction encouraged researchers and developers in the world of digital art to create applications where artists can exploit the new-found third dimension in their artistic pieces. In 2016, Google LLC released Tilt Brush, one of the earliest commercial applications that tackled this realm of digital art.

Tilt Brush is a Virtual Reality drawing/sketching application developed in the Unity Game Engine. It allows users to sketch "mid-air" by having the application translate their hand/arm movements into various brush strokes (Figure 1.1). With this manner of drawing, artists can either simply paint a two dimensional image mid-air, or draw 3-dimensional objects using multiple paint strokes per face. Aside from the size and color of the paint brush, users can also choose between multiple types of paint brushes, each of which has a different shape, texture, and even animations.

In 2021, Tilt Brush's official development was terminated and it was re-released as open-source software [1]. This has created an interest in expanding Tilt Brush and tackling problems that were left unsolved.

To investigate the overall user experience with Tilt Brush, I conducted a short test with an expert from a professional background in Graphic Design and with no prior VR experience. In a 45-minute test with a commercial build of Tilt Brush, the tester was instructed to first freely explore the application and learn its tools. Then, once they felt ready, they were instructed to draw an object that they draw frequently on their own and to provide feedback on their overall user

1 The controllers transmit two forms of input: button/analogue stick input (similarly to a video game controller), and 6 degrees-of-freedom positional/rotational data to precisely track hand/arm movements.

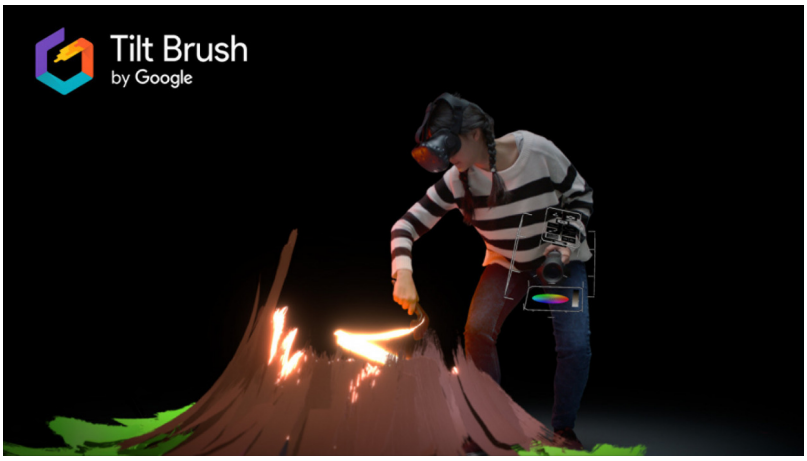

Figure 1.1: A person using Tilt Brush. Their right hand holds the brush, meanwhile their left hand holds the user interface of the application for selecting tools and adjusting settings. Reproduced from [1].

experience. Following the test, the tester expressed dissatisfaction with the Hull brushes, a subset of Geometry brushes[2] (Figure 1.2). Hull brushes work by creating a convex hull that envelops control points through which a user's hand has travelled. These brushes also happen to be Tilt Brush's only way of creating 3-dimensional geometry that is not a linear brush stroke by nature. The tester found these brushes limited and difficult to control, having to work with rough results that frequently needed to be re-drawn since Tilt Brush offers no way of editing strokes after their placement.

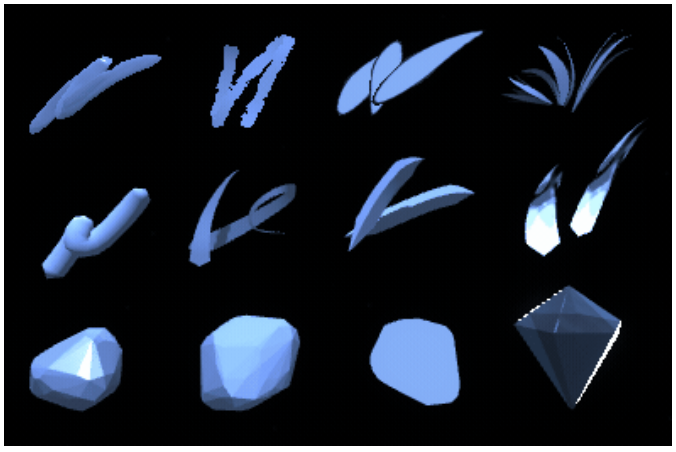

Figure 1.2: All of the Geometry brushes in Tilt Brush. The bottom row contains all of the Convex Hull brushes, which the tester was dissatisfied with. Reproduced from [5].

The goal of this thesis project is to solve this issue by granting users the ability to fine-tune 3D brush strokes after they were drawn.

2 Brushes that create meshes more complex than the planar meshes from other brushes, such as cylindrical shapes.

Given that all of the strokes in Tilt Brush are fundamentally meshes, Sculpting was chosen as the best method for solving the problem.

The rest of the document is organized as follows: I first describe work related to Tilt Brush in Chapter 2, then the solution's design and development in Chapters 3 and 4, respectively. Finally, I provide a reflection on the project's results and future in Chapters 5 and 6, respectively.

2 RELATED WORK

The creation of artistic tools like Tilt Brush has introduced a new form of artistic work spaces: ones that require artists to work in three dimensions instead of two, while also maintaining a direct presence in the workspace. This change is difficult for people to grasp, evidenced by the fact that people of both professional [3] and non-professional [8, 10] backgrounds encounter difficulties with such applications early on. Users report major inaccuracies with their brush strokes (e.g. attempts at straight strokes ending up as concave ones) [3], and some struggle to intuitively grasp the new-found third dimension [8]. As a result, a trend can be observed where VR artistic tools often lack the desired precision and have room for improvement with regard to user input [3].

However, these tools also offer new forms of both creating and perceiving art, leading to the pursuit of new avenues of research. The direct capture of a user's head and hand motions enables new possibilities for tools, applications, and presentation of graphical work.

Aside from evaluating/improving the approachability and the utility of artistic tools in VR, there is also interest in exploring and exploiting said possibilities. For instance, research has been conducted on the creation of tools that make use of 3-dimensional inputs, such as generating hairstyles on models of heads [13] or generating trees from the ground up [14]. Both of these examples grant the user the ability to directly trace a 3-dimensional brush stroke for their respective algorithms to interpret, which is not as convenient or intuitive with conventional input devices such as a computer mouse or a pen tablet, as they are only able to directly input 2-dimensional strokes. With all this in mind, artistic applications in Virtual Reality have the potential to improve with regards to precision and at the same time there is a lot of potential for reinventing conventional methods of user interaction with software.

2.1 TILT BRUSH

Before its open-source release, Tilt Brush also experimented with new forms of VR tools. In 2018, Tilt Brush was updated and introduced a new type of brushes into its set of tools: Hull brushes (Figure 1.2) [5]. Hull brushes introduced a new form of control for users, allowing them to capture a full 3-dimensional object in one cumulative brush stroke instead of multiple brush strokes per face [5]. Unfortunately, even though these tools introduce a new, advanced way of drawing

for artists, they do not facilitate any control or fine-tuning of existing strokes. As a result, the aforementioned problem of general VR imprecision combines with the complexity of this tool to create a problematic toolset that is difficult to control. Although Tilt Brush has some assistive tools to mitigate issues like this, they only allow the user to draw primitive shapes or straight lines, which can be insufficient for complex artworks.

Less than a month after the Open-Source re-release of Tilt Brush, a successor called Open Brush was created as a public fork. Since then, Open Brush has been in active, open-source development. It has introduced new features, such as better precision, bi-manual brushes[1], etc. [11]. However, it still does not facilitate retroactive editing of hull brushes.

2.2 SCULPTING

This thesis seeks to resolve the imprecision of Tilt Brush's Hull Brushes by introducing Digital Sculpting into its selection of tools. Digital sculpting is a well-established concept that has existed on desktop computers for decades [7]. It is a method of reshaping/manipulating meshes that is more natural to artists and can often result in smoother meshes. Thus, it is frequently used for modelling organic subjects, such as humans, animals, etc. (Figure 2.1).

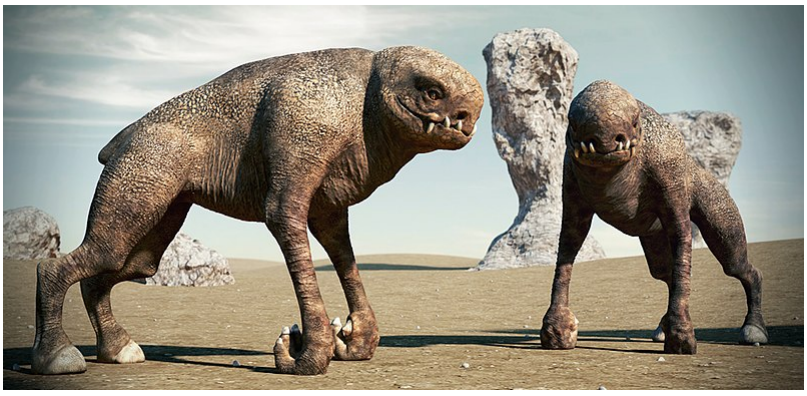

Figure 2.1: A piece of digital artwork depicting alien creatures which were sculpted in ZBrush, a popular desktop sculpting application. Reproduced from [15].

In VR, sculpting is not a new concept either. Since the commercial release of VR, several commercial sculpting applications have been released, which have already established a precedent in the design of such applications. Most VR sculpting applications strive to mimic/translate desktop sculpting applications, but variations exist that instead attempt to translate traditional sculpting into VR, by creating virtual replicas of real-life sculpting tools [9]. Overall, there is a varied

1 Brush tools that require control from both hands, instead of just the dominant hand.

selection of sculpting applications and designs that artists can choose from.

In general, digital sculpting has two variations: voxel-based sculpting and mesh-based sculpting. Voxel-based sculpting provides significantly more precise control that is not affected by the vertex layout of the sculpture. Mesh-based sculpting is heavily dependent on the vertex layout, but it has better performance due to its non-uniform resolution [9]. Tilt Brush itself does not have any support for voxels, therefore the proposed sculpting toolkit is mesh-based.

Although the solution that this thesis proposes already exists in many other forms, it stands out in the fact that it is an *Open-Source* VR Sculpting solution. There are few open-source VR sculpting applications and only one was found to have been in active development at the time of writing [6]. This project is expected to be a noteworthy addition to the realm of Open-Source VR Sculpting, with the potential to be merged into Open Brush and to become more accessible to the average VR artist than previous Open-Source VR sculpting applications.

3 CONCEPT

As previously stated, Tilt Brush's design philosophy does not support significantly modifying existing brush strokes, instead expecting users to redraw undesired strokes. Even tools that do make some retroactive edits, such as the Recolor tool[1], delete the original stroke and make a modified recreation of the original stroke in its place.

Although this limitation is manageable for simple brush strokes, hull brushes are fundamentally too intricate to be treated the same way as the rest of the brush/stroke types. As a result, having no option to edit them considerably limits their potential.

With the ability to shape strokes to their liking with a varied selection of tools, users could both reduce the number of re-drawings they would have to make and increase the general creative potential of these brush types in their applications as well.

As previously mentioned in Chapter 2, sculpting is a well-established 3D modeling technique, familiar to many people with an interest in 3D art. Therefore, sculpting tools within Tilt Brush are expected to be an approachable concept to artists as long as they provide similar forms of interaction to existing sculpting tools.

3.1 EXISTING AND PROPOSED TOOLS

As mentioned before (Chapter 1), VR sets often consist of a head-mounted display and two controllers, one for each hand. Within Tilt Brush, each of the two controllers serves a different purpose. The dominant-hand controller[2] acts as the primary way of interacting with the virtual canvas[3]. Meanwhile, the non-dominant hand holds the user interface, which includes many settings such as tool selection and brush parameters. The two controllers and their functions can once again be seen in Figure 1.1.

Tilt Brush's dominant-hand controller can primarily interact with the virtual canvas in two ways: Brushes and Tools. Brushes serve the purpose of creating strokes on the canvas. All brushes share the same UI appearance, where a cone-shaped pointer indicates the position and a circle indicates the size and orientation of the brush stroke (Figure 3.1).

1 A tool that changes the color of an existing brush stroke while retaining all of its other properties.

2 Within Tilt Brush, a user can specify which hand/controller is the dominant one.

3 The canvas is an implicit concept within the functional implementation of Tilt Brush. From a user's perspective, there is no explicit, visible canvas. A user can just draw mid-air, ergo the entire virtual space around them acts as the canvas.

Tools are a way of making additional modifications to the drawing process, many of which are similar to that of desktop art applications. Examples include an eraser tool for deleting strokes and a selection tool for grouping and repositioning strokes. Tools in general are represented differently; instead of a pointer and a circle, they are represented as a spherical region, where the sphere dictates the area of influence. When the tool is activated by the user, the tool's assigned actions are performed with whichever strokes the sphere comes in contact with. Examples of such tools can be seen in Figure 3.2.

Figure 3.1: Example of a brush in Tilt Brush. Note the conical pointer and circle protruding from the controller.

Figure 3.2: Examples of tools in Tilt Brush, namely the Eraser and Selection tools.

Considering that the sculpting tool should fundamentally resemble other tools, there was a need to design a sculpting tool that can support non-spherical interactions/intersections, while still maintaining a spherical shape like the other tools. This is not just due to aesthetics; there is also a functional reason: The base implementation of all of the stroke interaction tools uses GPU-level calculations that exploit the spherical shape of the tool to calculate positional intersections optimally. Therefore, any other base shapes would require a significant

amount of additional development effort, where a lot of fundamental modifications would need to be made.

To solve this problem, I conceptualized a design where the sculpting tool is primarily a sphere just like the other tools, but it can contain *subtools* inside the sculpting sphere to provide both a visual indication to the user and also change the way the tool interacts with a target mesh.

With these tools, a user would be able to manipulate groups of vertices inside the sphere's radius and change their positions based on various functions built into the subtools. The base sphere would determine if the tool is in contact with the stroke, and then the subtool inside (or lack thereof) would perform manipulations on the vertices within the sphere. To detect and manipulate the right vertices, the sculpting tool must access the geometry data of the stroke.

3.2 GEOMETRY DATA

Tilt Brush has multiple modes of representing stroke data, however only one is actively used, which is the "Batched Stroke" Mode. In this mode, strokes are bundled together in batches and represented as a hierarchy of various classes, which perform the following roles:

- **BatchPool:** Stores all of the batches on the canvas;
- **Batch:** Stores all strokes of the same type bundled in BatchSubsets, along with all of their geometry in a GeometryPool[4];
- **BatchSubset:** Stores the Stroke object along with data that specifies which parts of the parent batch represent the geometry of the stroke;
- **Stroke:** Holds the basic properties of a brush stroke, i.e. colour, type, control points, and Transformation data[5] relative to the canvas.

For further elaboration, see an example scenario in Figure 3.3. Since all of the strokes' geometry is stored in the parent batch of a Stroke, the goal of the tools is to retrieve the right portion of geometry data based on which stroke they are in contact with, then perform a modification to a subset of that geometry data that is within the tool's area of influence.

4 This is a class that holds all of the geometry data needed to represent a stroke. It serves as an interface between Unity's mesh data and the rest of Tilt Brush, which makes it the best way to access and change a stroke's mesh.

5 Position, rotation, and scale

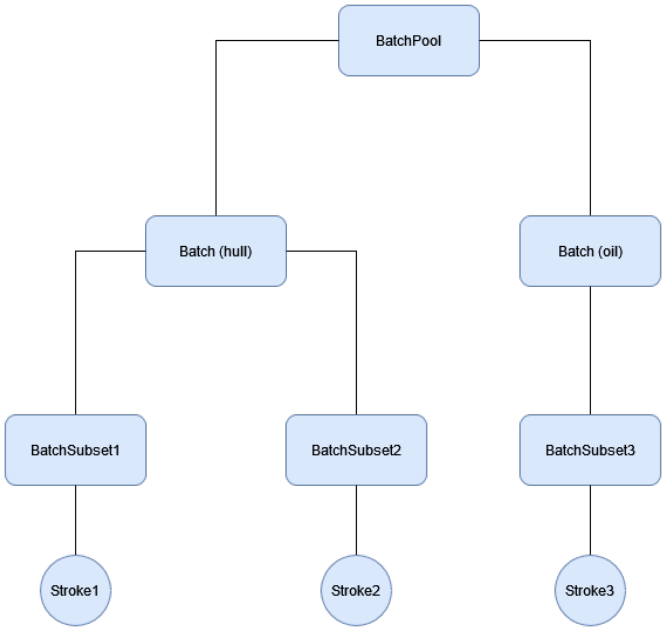

Figure 3.3: Diagram of the object hierarchy in an example canvas, where there are two hull brush strokes and a single oil brush stroke.

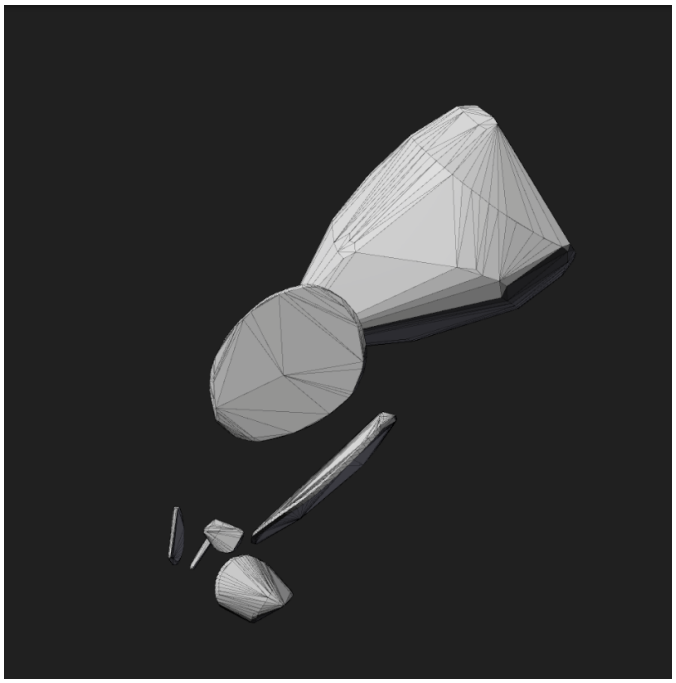

Figure 3.4: Meshes of arbitrary hull brush strokes. Note the non-uniform triangle distributions.

In Tilt Brush, the geometry data of strokes is often allocated together in the parent GeometryPool as a single, one-dimensional array of vectors, which makes topology modifications dangerous to implement due to cascading indexing errors. As this would be difficult to address in the allotted time-span of this project, the sculpting toolkit currently

only performs vertex layout modifications, with no changes to topology. As a consequence of this, the existing topology of a hull mesh might not always be ideal for making sculpting modifications, as they can have a non-uniform triangle distribution (Figure 3.4).

3.3 APPLICATIONS AND IMPACT

The proposed sculpting tools facilitate the ability to modify strokes retroactively, which is expected to reduce the need to re-draw unsatisfactory strokes and make much more intricate Hull brush strokes easier to create. By granting this additional possibility to users, overall precision of Hull brushes and even other brush types is expected to increase.

These changes could significantly expand the creative potential of Tilt Brush and encourage new developments in retroactive edits of brush strokes, beyond the existing tools such as Recolor and beyond even the proposed sculpting tool.

As stated in Chapter 2, Open Source VR Sculpting is a currently inactive realm of Open-Source software. This project has the potential to rekindle that avenue of development, especially if it is integrated into a larger repository such as the aforementioned Open Brush.

4

REALIZATION

When the user activates the sculpt tool, they are presented with a pop-up UI panel that allows them to switch between sculpting subtools (Figure 4.1). On the Brush controller, a button icon is displayed that indicates to the user that they can switch between the positive and negative modes of the various tools, which switches the direction in which vertices are manipulated (Figure 4.2). For the simpler subtools, the positive mode implies going forward (away) from the user, while the negative mode implies going backward (towards) the user. However, the meanings of these modes can vary for other subtools.

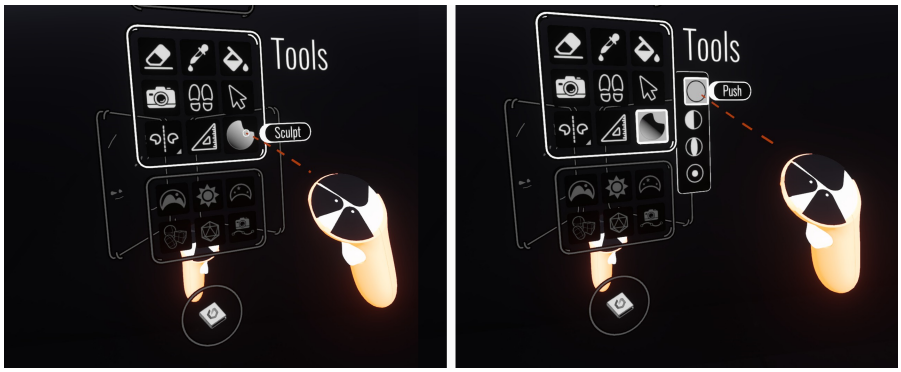

Figure 4.1: The user interface for selecting the sculpt tools. On the left, the sculpting toolkit is inactive. On the right, the sculpting toolkit is active and the menu with subtools is displayed, with the Push tool selected.

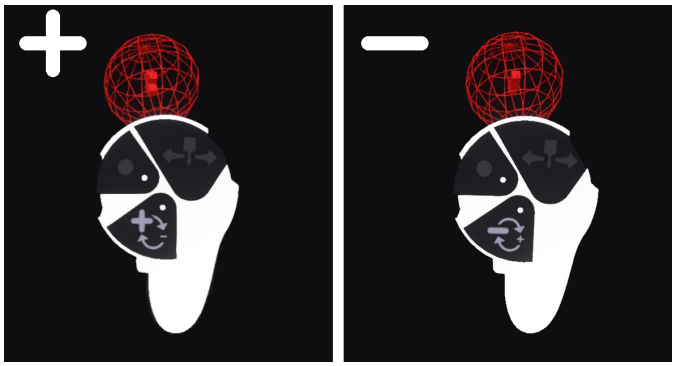

Figure 4.2: The button to switch between positive and negative modes, displayed on the bottom left of the dominant-hand controller. For additional clarity, the icon of the button varies in each mode.

Fundamentally, the base sculpting tool mimics existing Tilt Brush tools in terms of its code implementation, due to the reasons previ-

ously mentioned in Chapter 3. In addition, the entire toolset is designed with scalability in mind, allowing future developers to quickly include a new sculpting subtool by inheriting from a base class and then registering it in a manager class that keeps track of all of the tools and handles UI interaction.

4.1 SUBTOOLS

Four tool variations were implemented in total, the designs of which were inspired by tools in Blender, an open-source 3D Modeling/Sculpting application [4]. All of these options can be seen in Figure 4.3. Below are the descriptions of each subtool, including pseudocode algorithms[1] and images depicting their results.

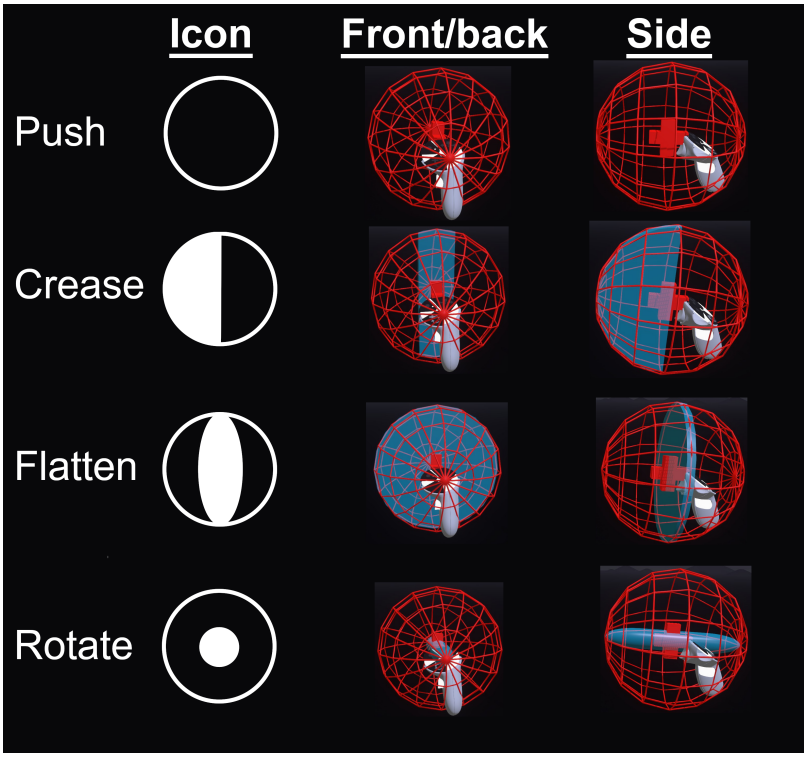

Figure 4.3: The graphical representations for all of the subtools, including both their menu icons and the tools displayed on the user's dominant hand.

4.1.1 *Push/Pull*

The Push/Pull subtool is the simplest variation of the sculpting tool, having originally acted as the foundation for the entire toolset during prototyping. It has no visual subtool, with the base sculpting sphere

1 NB: These are not direct representations of the actual code. Some engine-specific code is omitted and some code is restructured.

doing all of the interaction. The purpose of the tool is to push spherical dents into meshes in its positive mode, and making spherical bulges in its negative mode. To push dents, it linearly translates vertices away from the center of the tool sphere. In its negative mode, the vertices are translated towards the center, but not in a linear fashion. Instead, the distance they are translated by depends on their original distance from the center of the tool, squared. This choice was made as a linear translation towards the center would result in spike formations, which is not the expected outcome if conceptualized as the opposite of a dent shape. See Figure 4.4 for results and Listing 4.1 for pseudocode.

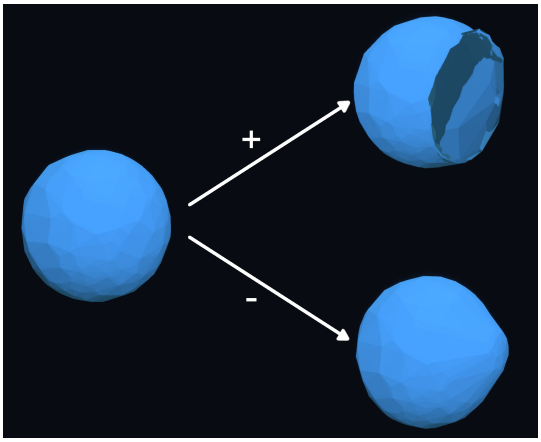

Figure 4.4: Results of the Push tool when applied to a spherical stroke.

```
if (vertex is inside sphere)
    // Vector from the vertex to the center of the tool.
    direction = vertex - toolCenter
    if (positiveMode)
        vertex += (defaultStrength * direction).normalized()
    else
        vertex -= ((defaultStrength * direction.magnitude^2) *
            direction).normalized()
```

Listing 4.1: Pseudocode for the Push tool.

4.1.2 *Crease*

The Crease subtool serves the purpose of making a narrow crease on a shape, in a narrow row of vertices. It contains a crescent shape that limits the area that the crease tool can affect. Unlike the Push tool, this subtool pushes/pulls based on the center of the geometry that is being interacted with, instead of the center of the tool itself. See Figure 4.5 for results and Listing 4.2 for pseudocode.

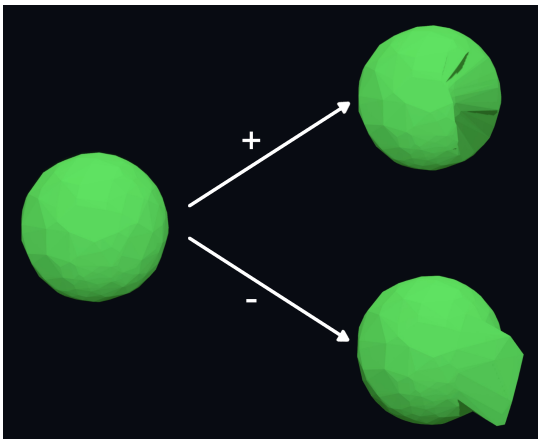

Figure 4.5: Results of the Crease tool when applied to a spherical stroke.

```
if (vertex is inside sphere && vertex is inside subtool)
    // Vector from the vertex to the center of the stroke.
    direction = vertex - strokeGeometryCenter
    if (not positiveMode)
        direction = -direction
    vertex += (defaultStrength * direction).normalized()
```

Listing 4.2: Pseudocode for the Crease tool.

4.1.3 *Flatten*

The Flatten subtool contains a disk shape inside the sculpting sphere. When the user activates the tool, any vertex inside the sphere will gravitate towards the disk and stop on its surface. Unlike the other subtools, this subtool does not have a negative mode. Nevertheless, it still has two philosophies of usage. When activated while the disk is *outside* of a shape, all vertices within the sphere form a flat protrusion. On the other hand, when activated while the disk is *inside* of a shape, the shape compresses flat against the disk. See Figure 4.6 for results and Listing 4.3 for pseudocode.

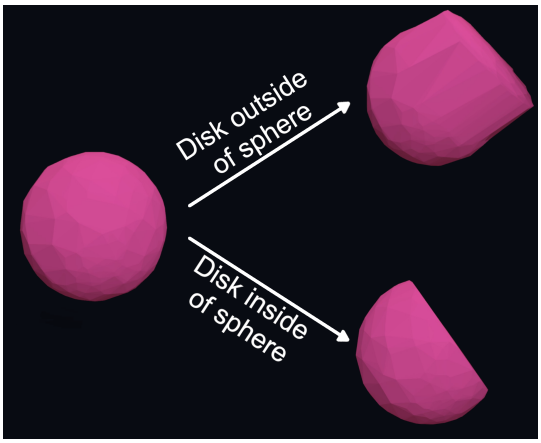

Figure 4.6: Results of the Flatten tool when applied to a spherical stroke.

```
if (vertex is inside sphere)
    closestPoint = subTool.GetClosestPointTo(vertex)
    vertex = closestPoint
```

Listing 4.3: Pseudocode for the Flatten tool.

4.1.4 *Rotate*

The Rotate subtool has a tube shape in the center that indicates a rotational pivot. When the user activates this tool, vertices inside the sculpting sphere will circularly rotate around the pivot, in a circular motion around a plane, which is perpendicular to the local z-axis of the pivot. The positive/negative modes determine whether the vertices will rotate clockwise or counter-clockwise. See Figure 4.7 for results and Listing 4.4 for pseudocode.

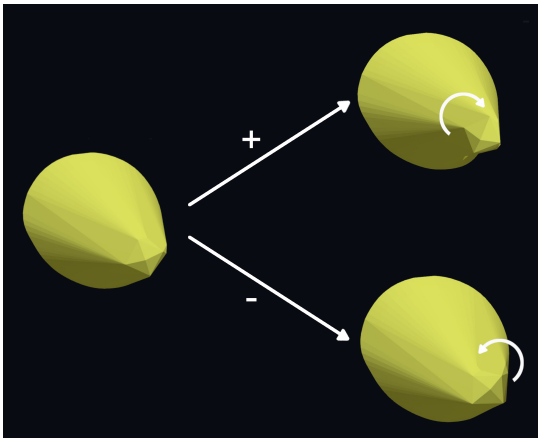

Figure 4.7: Results of the Rotate tool when applied to the tip of a cone-shaped stroke facing the camera.

```
closestPoint = subTool.GetClosestPointTo(vertex)

angle = -90
strength = 0.05
if (not positiveMode)
    angle = -angle // Rotate counter-clockwise.

// rotation is a Quaternion that represents rotation around the
    localZ axis of the tool.
rotation = RotateAround(angle, tool.localZ)

// Add a small incremental rotation that is dependent on the
    vertex's distance to the subtool.
vertex += ((vertex - closestPoint).magnitude * strength) *
    closestPoint * rotation
```

Listing 4.4: Pseudocode for the Rotate tool.

4.2 DEVELOPMENT PROCESS

Although the Tilt Brush codebase is open-source, it lacked the proper documentation to facilitate open-source development. A significant portion of the development time was spent on learning and understanding how the codebase functions through direct observation of the code and experimentation. As a result, the scope of the project had to be narrowed. Considering how intricate and inter-connected the codebase is, I had to frequently make sure that the sculpting tool works well with all of the other tools. Unfortunately, integration with Tilt Brush's Mirror tool had to be skipped; however, I am confident that this can be resolved in future implementations.

Tilt Brush was also not designed with any external geometry modifications in mind. This resulted in a need for substantial modifications to fundamental portions of Tilt Brush's codebase. First, the geometry data of every single stroke had to be set from 'private' to 'public'. The second change was much larger in scale:

Tilt Brush never had a need for saving geometry and could retain virtually all of the necessary data by storing only the strokes' control points. As this would discard any sculpting changes made in the sketch, the data that Tilt Brush serializes had to be adjusted. For every stroke, Tilt Brush now reads an integer number which specifies how many vertices a stroke has. If the value is zero, the geometry is created from scratch, assuming that the stroke was never sculpted. Otherwise, Tilt Brush deserializes vertices equal to the integer value and applies them to the re-drawn stroke. Unfortunately, this has caused Tilt Brush to be incompatible with save files from other/previous versions of Tilt Brush.

5 EVALUATION AND RESULTS

Throughout development, I applied the sculpting toolkit in my own creative works to test both its quality and functionality. Even at early stages, when only the basic Push tool was completed, the toolkit already showed major potential in solving the original issue of hull brushes' imprecision, by letting the user correct errors and imperfections in their hull brush strokes. Nevertheless, additional subtools provided better control than the basic push tool in specific situations, therefore the sculpting toolkit became more versatile with each new subtool.

The toolkit has also created the potential for sculpting shapes that would otherwise be impossible to draw in the original version of Tilt Brush. For example, Figure 5.1 shows a drawing of a well, the base of which was made by stretching a toroidal brush stroke using the flattening tool; the original stroke is otherwise restricted to a tube shape.

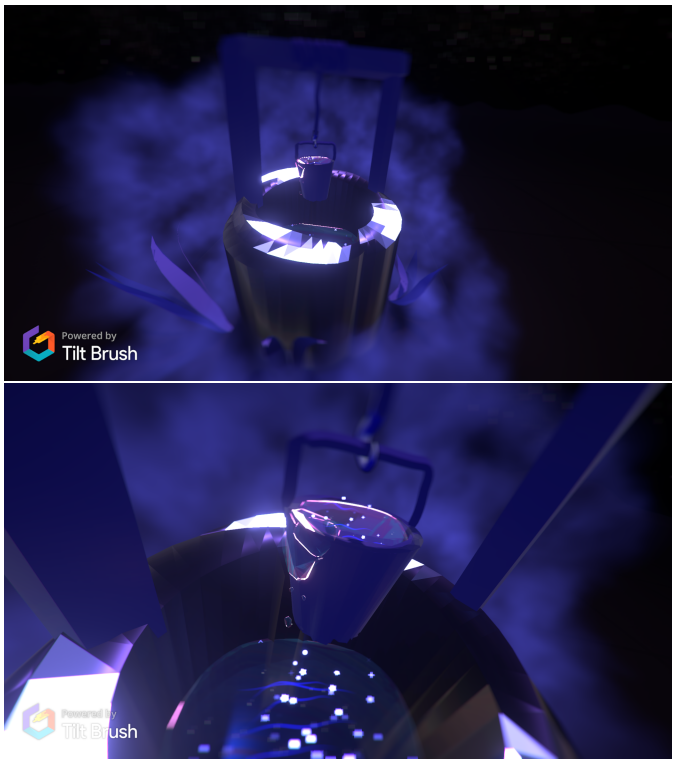

Figure 5.1: An example sketch made with the help of the sculpting toolkit. The well itself, the bucket, and the water flowing out of the bucket were all modified with the sculpting tools.

Close to the end of development, the previously mentioned tester (Chapter 1) was invited once again to evaluate the sculpting toolkit. Following a completely free-form evaluation, the tester reported that they found the sculpting toolkit to have made a noticeable improvement and expressed it as a 20% improvement on a numeric scale (Figure A.6). Also, they reported a considerable learning curve, rated 7 points out of 10. Despite that, the sculpting toolkit is "necessary for working with Hull Brushes" according to them. Overall, they found the Flattening and Push subtools to be the most useful of the four.

Based on both my own and the tester's observations, some issues remain in the sculpting toolkit, namely the following:

- The vertex movements are not always intuitive.
- When sculpting changes are made, the normal vectors of each vertex are not updated, which can lead to confusing shading.
- The sculpting tool does not work with the built-in mirroring/symmetry tool.
- On rare occasions, the selection tool can cause some sculpting changes to disappear from the Undo History and become permanent[1].

For the questionnaire and the answers used to make the deductions of this section, see the Appendix (Figures A.6, A.7).

1 This happens due of the fact that the selection tool is not guaranteed to preserve the batch hierarchy of strokes when used.

6 CONCLUSION AND FUTURE WORK

In conclusion, the sculpting toolkit has proven to be an ample solution by successfully enhancing the precision of Tilt Brush, especially with Hull Brushes. Users are now able to make retroactive edits to any brush stroke's mesh, where they can both make small corrections to match their strokes with their initial desired shapes, or completely reshape the stroke into something new, which cannot be achieved any other way.

With all of that in mind, the sculpting toolkit still has room for expansion and improvement. The scalable design encourages future development and the creation of many more tools that modify vertex layouts in new ways. The fundamental design of the sculpting toolkit also leaves room for sculpting tools which modify topology, therefore the biggest limitation of the sculpting toolkit is solvable given more time.

Aside from expanding variety, the sculpting toolkit can benefit significantly from the solution of issues outlined in Chapter 5 and the Appendix (Figures A.6, A.7). In addition, improvements can be made to visual feedback, by highlighting all vertices that are within reach (suggested by the tester), or by displaying wireframe shaders on meshes while the sculpting tool is active.

Following discussions with an Open Brush developer, I believe that the future of this project lies as a part of Open Brush. Therefore, I aim to integrate all of this project's work into the Open Brush codebase in the near future, where developers and artists alike will have much easier access for both usage and improvement.

APPENDIX

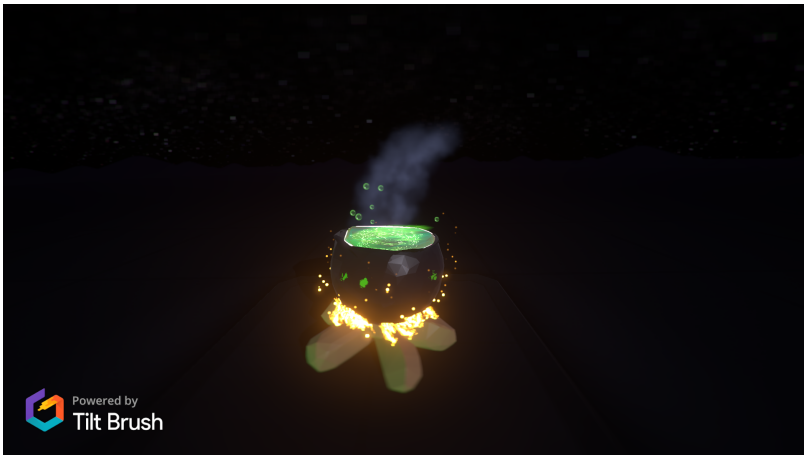

Figure A.1: First sketch made with the prototype sculpting tool, where a sphere was dented to create a witch's cauldron

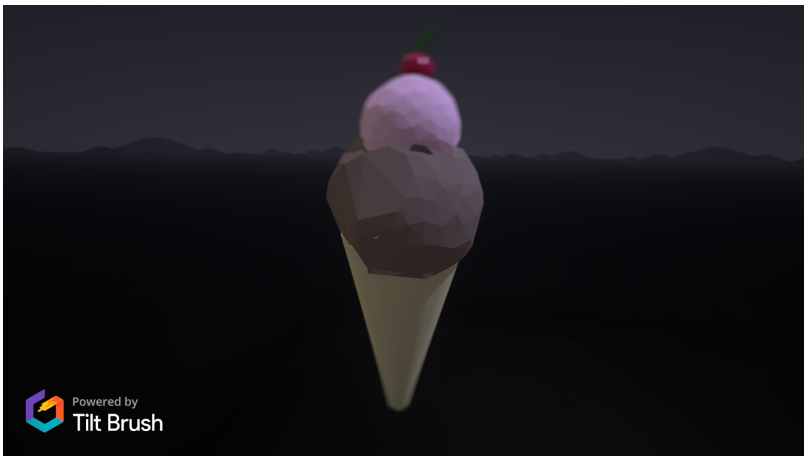

Figure A.2: Sketch of an ice cream cone, where the sculpting tool was used to shape the portions of ice cream.

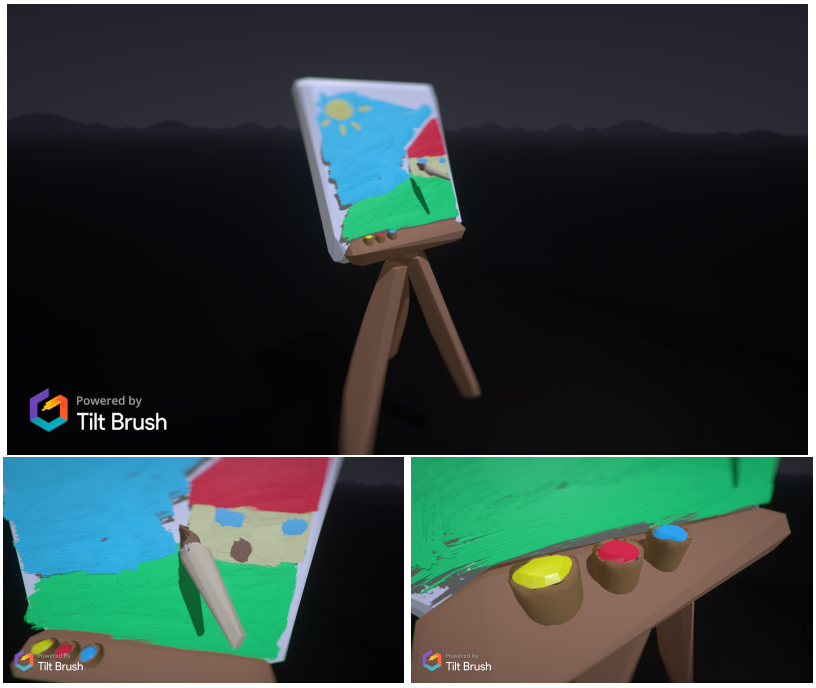

Figure A.3: Sketch of an artist's canvas, where the sculpting tool was used to shape the paintbrush and the paint containers.

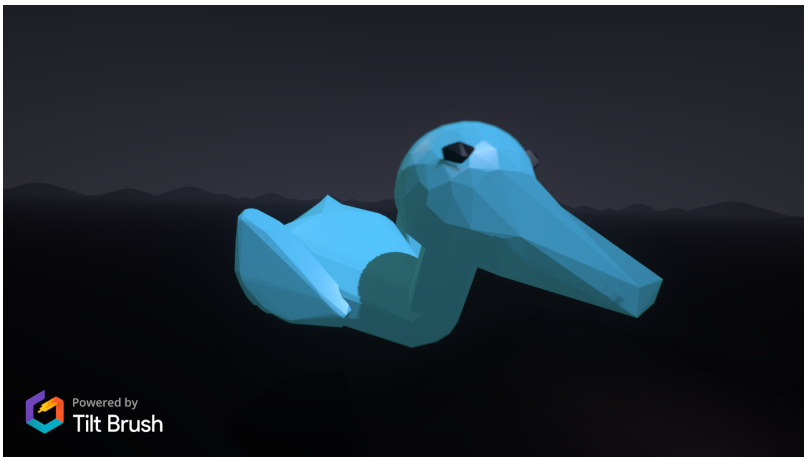

Figure A.4: Sketch of a bird, where the sculpting tool was used to shape the tail and the beak.

Figure A.5: Sketch of a strawberry cake, where the sculpting tool was used to flatten the cake and the candles.

Sculpting tools feedback

You were able to get the results you wanted when using hull brushes WITHOUT sculpting tools *

	1	2	3	4	5	6	7	8	9	10	
Strongly disagree	○	○	○	◉	○	○	○	○	○	○	Strongly agree

You were able to get the results you wanted when using hull brushes WITH sculpting tools *

	1	2	3	4	5	6	7	8	9	10	
Strongly disagree	○	○	○	○	○	◉	○	○	○	○	Strongly agree

Please rank the tools in terms of how often you see yourself using them *

	Push	Crease	Flatten	Rotate
Most frequently	○	○	◉	○
More frequently	◉	○	○	○
Less frequently	○	○	○	◉
Least frequently	○	◉	○	○

Figure A.6: Questionnaire filled in at the final user evaluation (page 1).

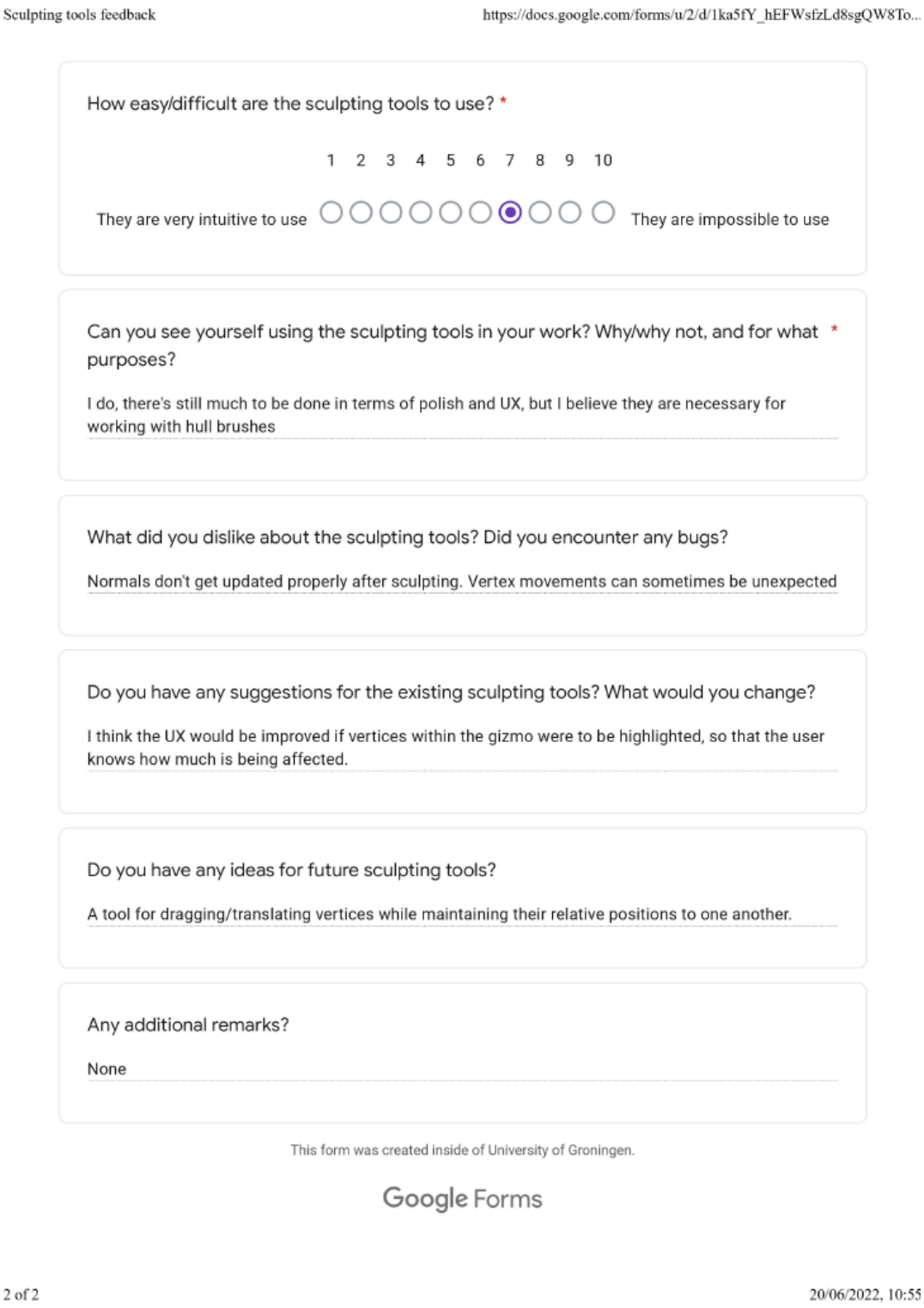

Figure A.7: Questionnaire filled in at the final user evaluation (page 2).

ACKNOWLEDGMENTS

I would like to thank my supervisors, Prof. Dr. Jiří Kosinka and Dr. Cara Tursun for being very responsive and helpful, always providing great feedback that steered my project in the right direction. I would also like to thank Miguel Bartelsman for participating in the evaluation of my work and ultimately shaping my research goal into what it is now.

BIBLIOGRAPHY

[1] T. Aidley and J. Corralejo. *The Future of Tilt Brush*. https://opensource.googleblog.com/2021/01/the-future-of-tilt-brush.html. 2021.

[2] V. Aleksić and D. Politis. "The Characteristics of Virtual Reality Usage in Educational Systems." In: *2020 International Conference on INnovations in Intelligent SysTems and Applications (INISTA)*. 2020, pp. 1–5. DOI: https://doi.org/10.1109/INISTA49547.2020.9194682.

[3] R. Arora, R. Habib Kazi, F. Andedrson, T. Grossman, K. Singh, and G. Fitzmaurice. "Experimental Evaluation of Sketching on Surfaces in VR." In: *CHI '17: Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems*. 2017, pp. 5643–5654. DOI: https://doi.org/10.1145/3025453.3025474.

[4] Blender. *Home of the Blender project - Free and Open 3D creation software*. https://blender.org. 2022.

[5] S. Cornelis. *Seven new things you can do with Tilt Brush*. https://blog.google/products/google-ar-vr/seven-new-things-you-can-do-tilt-brush/. 2018.

[6] CrispyPin. *VR-Sculpter: A WIP VR Sculpting App*. https://github.com/CrispyPin/vr-sculpter. 2022.

[7] T. A. Galyean and J. F. Hughes. "Sculpting: An Interactive Volumetric Modeling Technique." In: *Proceedings of the 18th Annual Conference on Computer Graphics and Interactive Techniques, SIGGRAPH 1991*. Vol. 25. 1991, pp. 267–274. DOI: http://dx.doi.org/10.1145/122718.122747.

[8] M. Donaji Barrera Machuca, W. Steurzlinger, and P. Asente. "The Effect of Spatial Ability on Immersive 3D Drawing." In: *C&C '19: Proceedings of the 2019 on Creativity and Cognition*. 2019, pp. 173–186. DOI: https://doi.org/10.1145/3325480.3325489.

[9] J.K. Moo-Young, A. Hogue, and V. Szkudlarek. "Virtual Materiality: Realistic Clay Sculpting in VR." In: *CHI PLAY '21: Extended Abstracts of the 2021 Annual Symposium on Computer-Human Interaction in Play*. 2021, pp. 105–110. DOI: https://doi.org/10.1145/3450337.3483475.

[10] A. Oti and N. Crilly. "Immersive 3D sketching tools: Implications for visual thinking and communication." In: *Computers & Graphics* 94 (2021), pp. 111–123. DOI: https://doi.org/10.1016/j.cag.2020.10.007.

[11] Icosa Team. *Open Brush homepage*. https://openbrush.app/. 2022.

[12] J. Tham, A. Hill Duin, L. Gee, N. Ernst, B. Abdelqader, and M. McGrath. "Understanding Virtual Reality: Presence, Embodiment, and Professional Practice." In: *IEEE Transactions on Professional Communication* 61 (2 2018), pp. 178–195. DOI: https://doi.org/10.1109/TPC.2018.2804238.

[13] J. Xing, K. Nagano, W. Chen, H. Xu, L. Wei, Y. Zhao, J. Lu, B. Kim, and H. Li. "HairBrush for Immersive Data-Driven Hair Modeling." In: *UIST '19: Proceedings of the 32nd Annual ACM Symposium on User Interface Software and Technology*. 2019, pp. 263–279. DOI: https://doi.org/10.1145/3332165.3347876.

[14] Q. Yuan and Y. Huai. "Immersive sketch-based tree modeling in virtual reality." In: *Computers & Graphics* 94 (2020), pp. 132–143. DOI: https://doi.org/10.1016/j.cag.2020.12.001.

[15] emonimo. *polvorón y anisete*. https://www.flickr.com/photos/23633963@N07/3898673230/. 2009.