Chair for Computer Science 10 (Media Computing and Human-Computer Interaction)

RWTH AACHEN UNIVERSITY

HaptiSculptVR - Virtual Reality Open Source 3D-Design with Haptic Feedback

Bachelor's Thesis
submitted to the
Media Computing Group
Prof. Dr. Jan Borchers
Computer Science Department
RWTH Aachen University

by
Dimitri Zimmermann

Thesis advisor:
Prof. Dr. Jan Borchers

Second examiner:
Prof. Dr.-Ing. Ulrik Schroeder

Registration date: 18.05.2018
Submission date: 16.10.2018

Eidesstattliche Versicherung

______________________________	______________________________
Name, Vorname	Matrikelnummer

Ich versichere hiermit an Eides Statt, dass ich die vorliegende Arbeit/Bachelorarbeit/ Masterarbeit* mit dem Titel

____

____

____

selbständig und ohne unzulässige fremde Hilfe erbracht habe. Ich habe keine anderen als die angegebenen Quellen und Hilfsmittel benutzt. Für den Fall, dass die Arbeit zusätzlich auf einem Datenträger eingereicht wird, erkläre ich, dass die schriftliche und die elektronische Form vollständig übereinstimmen. Die Arbeit hat in gleicher oder ähnlicher Form noch keiner Prüfungsbehörde vorgelegen.

______________________________	______________________________
Ort, Datum	Unterschrift

*Nichtzutreffendes bitte streichen

Belehrung:

§ 156 StGB: Falsche Versicherung an Eides Statt

Wer vor einer zur Abnahme einer Versicherung an Eides Statt zuständigen Behörde eine solche Versicherung falsch abgibt oder unter Berufung auf eine solche Versicherung falsch aussagt, wird mit Freiheitsstrafe bis zu drei Jahren oder mit Geldstrafe bestraft.

§ 161 StGB: Fahrlässiger Falscheid; fahrlässige falsche Versicherung an Eides Statt

(1) Wenn eine der in den §§ 154 bis 156 bezeichneten Handlungen aus Fahrlässigkeit begangen worden ist, so tritt Freiheitsstrafe bis zu einem Jahr oder Geldstrafe ein.

(2) Straflosigkeit tritt ein, wenn der Täter die falsche Angabe rechtzeitig berichtigt. Die Vorschriften des § 158 Abs. 2 und 3 gelten entsprechend.

Die vorstehende Belehrung habe ich zur Kenntnis genommen:

______________________________	______________________________
Ort, Datum	Unterschrift

Abstract

In this thesis, an application is presented for deforming the surface of a 3D model (Sculpting) with haptic feedback in virtual reality (VR). The HTC Vive VR set and the Touch 3D stylus from 3D Systems is the hardware used for this project. The software base for the sculpting is the open source tool SculptGL [1], which was modified to implement VR and a haptic interface.
The haptic feedback of the Touch does not only naturally increase the immersion into the VR even further, it also enables people with visual impairments to use the application by feeling the surface structure.
At the end of the work, a user study was conducted on the usability with and without haptic feedback during Maker Faire Hannover 2018. This study showed that sculpting in VR is much more intuitive and practical than working on a screen and that the use of haptic feedback accelerates this experience.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Digital Sculpting

Digital sculpting or virtual sculpting plays an important role in the creation of detailed high-resolution 3D models and is indispensable for areas such as video game and film industry. Especially in the professional sector, tools like ZBrush and MudBox are firmly established. These tools are very powerful in many ways, but they all have a common problem: Due to the richness of features, the usability suffers from complex keyboard controls and an overloaded user interface, which often comes with context sensitive sub menus.(Figure 1.1). This steep learning curve of the user interface is often one of the biggest hurdles for beginners in digital sculpting. And even after the user interface has been learned and internalized, working with these tools is often tedious. Sculpting usually requires frequent switching between different tools (especially the smoothing tool), adjusting the tool radius and tool strength, switching the extrusion direction, etc. In order to do this, the designer must constantly switch between the 3D viewport (the primary view of the object) and the tool window. Many important key combinations can only be reached with both hands (Ctrl+Shift+O). This, in turn, means that the designer has to constantly switch between

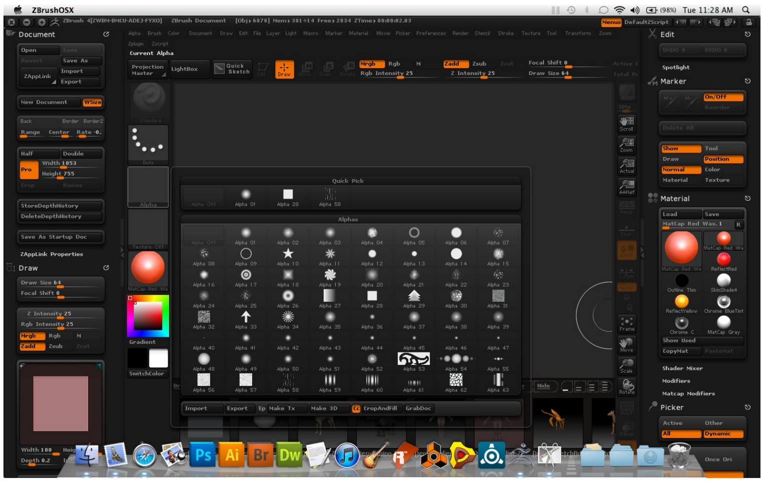
Figure 1.1: The user interface of ZBrush. Like most 3D design programs, ZBrush suffers from a very complex and confusing GUI.

the mouse and keyboard with the main hand.
For this reasons, the professional sector relies on pressure sensitive drawing tablets. Those can simplify the usability immensely, since many features can be reached much quicker using the drawing pen compared to the mouse and keyboard workflow. In addition, the tool strength is implicitly determined by the pressure on the tablet and the pen often has buttons which can be used to change the tool radius easily and quickly. But this does not solve another existing problem. By the projected 2D view of the 3D scene on the screen, the depth perception suffers due to the monocular view.

1.2 Virtual Reality

VR is used in many different areas besides entertainment, like education, medicine, architecture and military to name a few [2]. With the appearance of the Oculus Rift and the HTC-Vive, VR has attracted a lot of attention again, not least due to excessive media interest. But VR is not a new

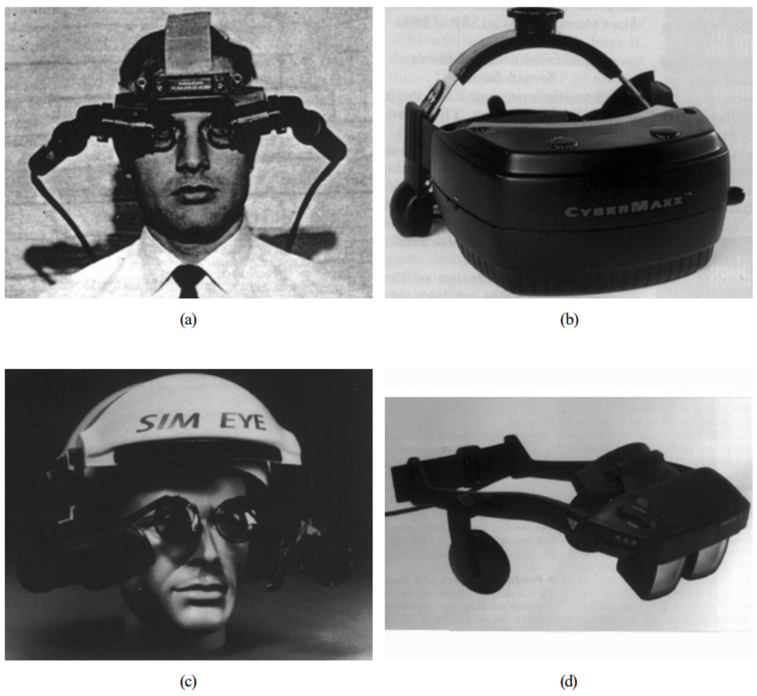

Figure 1.2: (a) Ivan Sutherland's HMD dated from 1968, (b) low cost CyberMaxx HMD, (c) advanced military Sim Eye HMD, (d) low cost see through HMD "i-glasses!" from Virtual I/O. [2]

idea (Figure 1.2). It first came up in 1965 and was mentioned by Ivan Sutherland: *"Make that (virtual) world in the window look real, sound real, feel real, and respond realistically to the viewer's actions"* [3].

In 1968, Sutherland presented the *"The Sword of Damocles"*, a device considered to be the first Head Mounted Display (HMD) with appropriate head tracking. It supported a stereo view that was updated correctly according to the user's head position and orientation.

Research of VR always went hand in hand with research in haptics, since haptics is a part of VR according to Sutherlands conception of a full immersion into a Virtual Environment.

Since then, a lot of research and study were done in the fields of VR, with a high number of results. First prototypes of force-feedback were presented in 1971 *(GROPE)*, a interactive silhouette projector in 1975 *(VIDEOPLACE)*, a stereoscopic monochrome HMD constructed at the NASA in 1984 *(VIVED)* up to the popular DataGlove in 1985 and the EyePhone HMD in 1988 – the first commercially available VR devices. Four years later, CAVE (CAVE Automatic Virtual Environment) was presented, a virtual reality and scientific visualization system. Instead of using a HMD it projects stereoscopic images on the walls of room (user must wear LCD shutter glasses). This approach assures superior quality and resolution of viewed images, and wider field of view in comparison to HMD based systems [2].

1.3 Haptic Feedback

Haptic feedback or just haptics is referred to the sense of touch. For this, a haptic device is required. A haptic device allows the user to interact with a Virtual Environment (VE) by applying and recieving force or pressure feedback. The most common haptic device is the smartphone, which gives the end user a touch feedback when a button is pressed or a new notification occurs. Other devices like force-reflecting joysticks or a force-feedback robotic arm give the user a mean to navigate in the VE through the sense of touch. To describe the amount of freedom of navigation, the term degrees of freedom (DoF) is used. A n-dimensional rigidbody has n translational and $\frac{n(n+1)}{2}$ rotational degrees of freedom. In Euclidean 3D space, there is a total of 6 DoF (Movement on X-Y-Z axes and rotation in Euler Angles). The Project *GROPE* [4] (Figure 1.3) was the first prototype for a 6 DoF force-feedback system realized at the University of North Carolina (UNC) in 1971. It was a haptic display used to interact with virtual protein molecules. In 1985, VPL release the DataGlove, a wearable device with fiber-optic bundles to track movements and orientation (Figure 1.3). Bilateral Master-Slave Manipulators (MSMs) – functionally no different from today's desktop haptic feedback systems, are used in the nuclear industry for save and remote interaction with irradiated material [5]. *Lahav et al* presented a research paper in 2003 with a case study. They created a VE in which a blind user can roam freely using a force feedback joystick, applying force whenever the user hits a virtual obstacle. The result showed that within a short period of time,

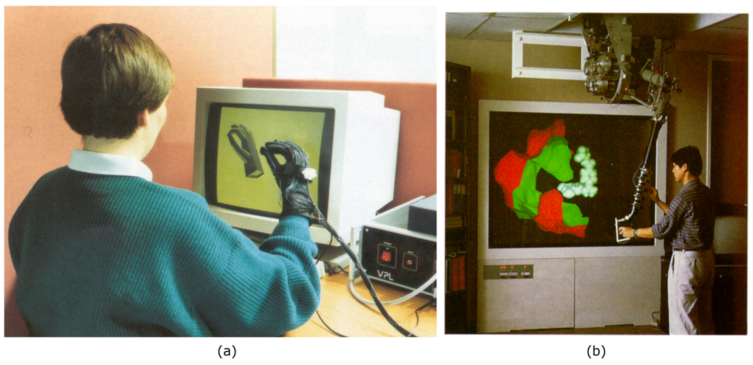

Figure 1.3: (a) The DataGlove from VPL released in 1985. (b) GROPE-III haptic display system in use.

the user could walk freely in the VE by using a cognitive map of the room [6]. Haptics is still an active research area and is used in fields such as medicine, entertainment/VR and assists people with seeing disabilities.

Chapter 2

Related work

After a brief introduction to the individual topics of this thesis in Chapter 1, it's time to look at some work which combines those topics. Unsurprisingly, there are already a lot of drawing and sculpting tools for VR. In 2016 Oculus VR released Oculus Medium, a sculpting and painting tool. The user can deform and paint over a clay like 3D object in VR using the Oculus Touch hand tracking controllers as input device. In the same year, Google released Tilt Brush for the Oculus Rift and the HTC Vive. In 2017, MasterpieceVR was released for both the HTC Vive and the Oculus Rift. Also in 2017, Gravity Sketch, a VR sculpting tool for car and shoe designers was released for HTC Vive, Oculus Rift and Windows Mixed Reality. An open source tool for VR Painting was released by Mozilla VR Team named A-Painter. It uses WebGL for rendering and WebVR for interfacing the VR devices and is build on top of the open source VR Web Framework A-Frame [7].

It is important to note though, that there are different ways of sculpting and the difference is how the 3D object is represented and deformed. One method to deform an object is called free form deformation. It uses control points to deform the surface with spline curve functions. The advantage of this method is, that it will work on both polygonal and solid models. However, using control points can be cumbersome since it's hard to know for sure what mesh deformation

3D objects can be represented as a set of vertices and indices that define a polygonal structure of the surface.
Other methods like spatial occupancy enumeration uses cells (voxel) to store solid mesh information.

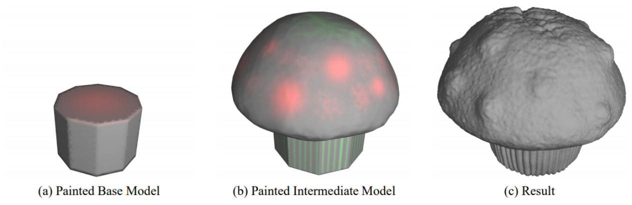

Figure 2.1: (a) The first deformation is painted on top of the base model. Red color means to extrude outwards. (b) The intermediate model is painted again. Green color means to extrude inwards. (c) The final result after two painting steps.

will be obtained when manipulating them. Free form deformation also relies on trivariate Bernstein polynomials, which are costly operations [8].
The other method is to manipulate the vertices of the surface directly by "painting" over it. The information is "painted" on the surface by drawing on it (Figure 2.1). This method has the advantage, that the user always knows directly how and which part of the mesh is getting deformed [9].

Sculpting with haptic feedback is also an active research topic. In 1998, the first publication for interactive sculpting with haptic feedback was released. This research introduces examples of haptic sculpting mediated by a physical model or constraint. Most prior work in haptic drawing and sculpting focused on interacting with a static model and properly simulating contact forces. This work introduced dynamic models for the creative process. These were based on physical models, but present physically impossible scenarios that allow new forms of expression. By focusing on models not realizable in the real world the users showed an expansion of the creative process through haptic feedback [10].
Since then, different approaches were researched to improve haptic force rendering for digital sculpting. In the research paper "Haptic Sculpting of Dynamic Surfaces" from the year 2000, the authors presented a method to directly manipulate physics-based B-spline surfaces without using control points. This method interfaces with a standard haptic device for haptic feedback [11]. In 2001, *McDonnell et al.* presented

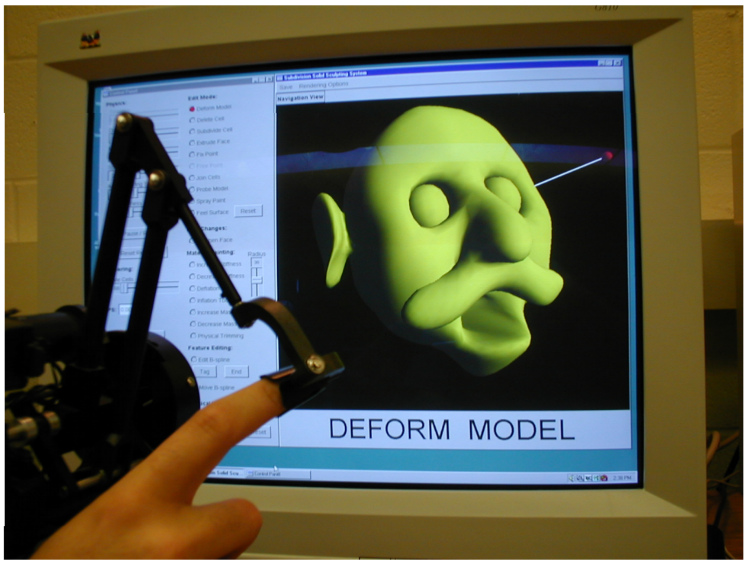

Figure 2.2: A PHANToM haptic device input device is used to sculpt the 3D model with force feedback [12].

a novel, interactive sculpting framework founded upon subdivision solids and physics-based modeling. The framework was able to use a PHANToM haptic device for haptic feedback sculpting [12] (Figure 2.2). A different approach was presented by *Blanch et al.* in 2004. In their work, they extended a virtual sculpting system with haptic feedback. In practice, they use the scalar field defining the implicit surface being modeled to efficiently compute several type of force feedback [13].
3D Systems, a company that engineers, manufactures and sells 3D printers, 3D scanners and haptic devices offers a haptic sculpting tool "Cubify Sculpt" in combination with one of their haptic devices.

Chapter 3

Extending a digital sculpting tool for VR with haptic feedback

$\vec{F} = \vec{D} * s$
$\vec{F}$: Haptic force to be applied
$\vec{D}$: Force direction (unit vector)
s: Stiffness

The goal of this bachelor thesis was to create a digital sculpting tool for VR with haptic feedback (HaptiSculptVR). In order to manage the task in the time constrain, the open source tool SculptGL [1] was used. SculptGL is a browser based sculpting application that can sculpt 3D polygonal meshes and brings an array of the most important sculpting brushes. It is written in JavaScript and uses WebGL for rendering. WebGL is a 3D graphics API based on OpenGL ES 2.0 and with the release of WebGL 2.0 now conforms closely to the OpenGL ES 3.0 API. Practically, WebGL lets the user render GPU accelerated 3D graphics through the browser. Positive about this is, that applications written in WebGL will run on any common operation system where a WebGL capable browser is supported (Table 3.1). In order to extend SculptGL for the HTC Vive, WebVR was used. WebVR is a JavaScript VR API that interfaces with most common VR devices. It is used to sample the HMD and controller tracking. Besides the tracking of the VR device, the rendering needed to be extended to stereoscopic rendering and a user interface had to be designed to use the tool inside VR (Section 3.1).

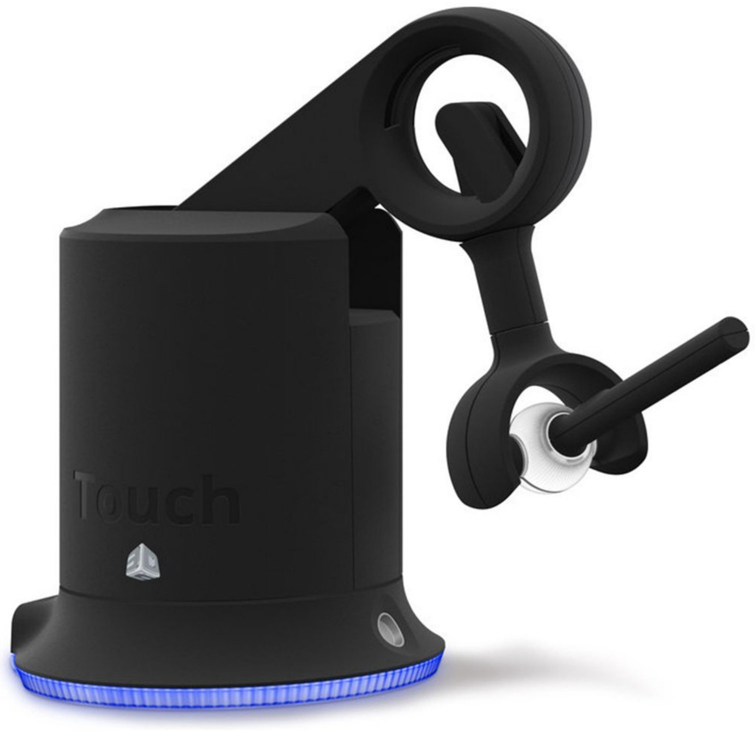

Figure 3.1: The Touch 3D stylus from 3D Systems is a 6 DoF haptic device. It comes with the SDK OpenHaptics to control the force input on the handle motors.

The other part of the work was to implement a haptic interface for the Touch 3D stylus from 3D Systems (Figure 3.1). The Touch is a 6 DoF haptic device and comes with OpenHaptics, a Software Development Kit (SDK) by 3D Systems for C++ with Python bindings. The SDK allows the user to sample the device position and orientation and to apply forces. Since 3D Systems only released device drivers for windows, the haptic device needs to be connected and operated through a windows machine. An architecture was designed that lets the C++ haptic backend communicate with the JavaScript web frontend. After that, a method was compiled to calculate the force feedback for a polygonal mesh (Section 3.2).

Browser	WebGL	WebGL 2.0	WebVR
IE	Version 11, 2013	Not supported	Not supported
Edge	Version 12, 2015	Not supported	Version 15, 2017
Firefox	Version 4, 2011	Version 25, 2013	Version 55, 2017
Chrome	Version 8, 2010	Version 43, 2015	Version 57, 2017
Safari	Version 5.1, 2011	Version 10.1, 2017	Not supported
Opera	Version 12.1, 2012	Version 43, 2017	Not supported
iOS Safari	Version 8, 2014	Not supported	Not supported
Chromium	Version 67, 2017	Not supported	Not supported
Blackberry	Version 10, 2013	Not supported	Not supported
Opera Mobile	Version 12, 2012	Version 46, 2016	Not supported
Chrome Android	Version 69, 2018	Version 69, 2018	Version 69, 2018
Firefox Android	Version 62, 2018	Version 62, 2018	Not supported
IE Mobile	Version 11, 2012	Not supported	Not supported
UC for Android	Version 11.8, 2016	Not supported	Not supported

Table 3.1: A list of browsers that support WebGL, WebGL 2.0 and WebVR with first supported version and release year (from 2018)

3.1 VR for SculptGL

For all readers who are not familiar with the workflow of rendering APIs, this short introduction should help to get a general understanding. To render a 3D model, it first has to be stored on the GPU memory inside so called vertex buffers. This vertex buffers store vertices, basically a float buffer where 3 floats build a vertex, which is a point in 3D space. In order to connect the vertices into triangles, an index buffer is used. This integer buffer stores indices that points inside the vertex buffer where three successive indices inside the index buffer build a triangle inside the vertex buffer. All the vertices of the object inside the vertex buffer are in object space. That means, that their position is relative to the object origin (usually the center of mass of the object). To render the object from the user view and perspective, the object vertices need to be transformed. To do this, the vertices are multiplied with a transformation matrix. This transformation matrix usually composes of this three matrices: the object to world space transformation, the world space to view space transformation and the final perspective transformation. The first transformation is encoded inside the model matrix and its effect is that the object ver-

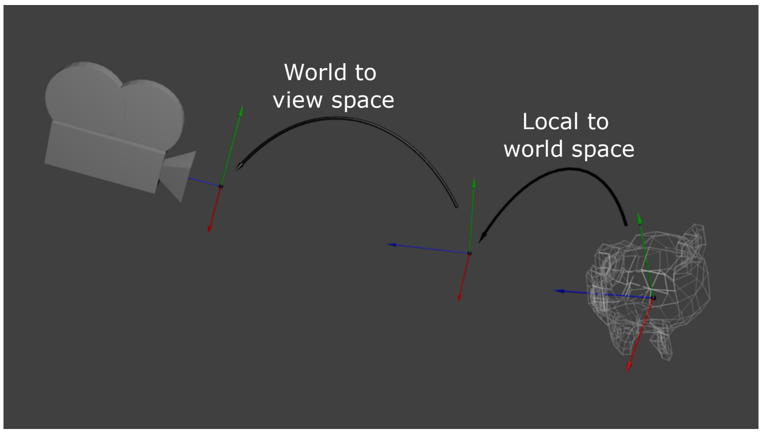

Figure 3.2: This figure shows the two steps to transform the vertices from object (or local) space into view space. From right to left: The object space vertices are multiplied with the model matrix and transformed from local space to world space. After this, the transformed vertices are multiplied with the view matrix and transformed into view space. The model and view matrix can be premultiplied into a single ModelView matrix.

tices are now relative to the world origin. Since translation isn't a linear transformation, the transformation process is extended to affine transformation. Therefore, 4x4 Matrices are used to encode scaling and rotation (linear transformation) inside the 3x3 part of the matrix and the translation inside the last row. The vertices are extended as well using 0 or 1 for the w component, depending whether it is a direction or a point. After the object is in world space, it needs to be mapped to the view space. The view matrix encodes the rotation and translation of the camera, from where the user is viewing the scene, relative to the world origin. Multiplying the vertices with the model matrix and then the view matrix maps the vertices from being relative to the object origin into being relative to the camera position (Figure 3.2). This is required for the last step, the perspective transformation. For this, the projection matrix is used. It stores information like the Field of View (FoV), the aspect ratio of the screen and the distance to the near and far clipping plane, which builds a viewing frustum. The projection maps the vertices from view space into a homogeneous space. Inside homo-

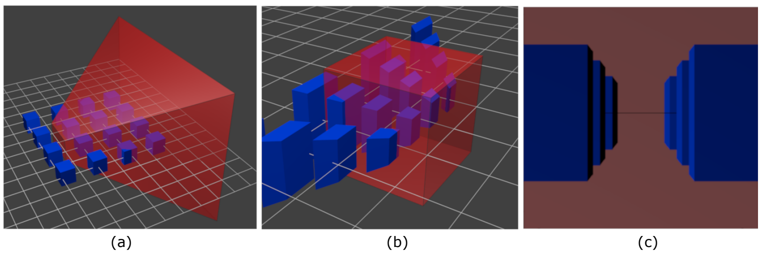

Figure 3.3: This figure shows the projection transformation: (a) The viewing frustum that is build by the projection matrix. (b) The transformation from view space into homogeneous space. The frustum is transformed into a cube. (c) The final image after the depth test, rasterization and fragment shader steps.

geneous space, all vertices are defined inside a small cube. Everything inside this cube is visible on the screen (Figure 3.3). After this, the depth test is performed and only triangles that are visible are passed to the rasterization step. The rasterization step determines which projected triangle is effecting which pixel on the screen. As the last step, a custom fragment shader runs and calculate the color of the pixel based on the result of the rasterizer.

3.1.1 Stereoscopic rendering in WebGL

The render method described above is called monoscopic rendering. That means, it only uses one camera (matrix) to render the scene from a single position. To use SculptGL with a HMD like the HTC Vive, its rendering needs to be extended to a stereoscopic rendering (Figure 3.4). Stereoscopic rendering renders the same scene twice based on the position where the eyes are supposed to be. It is important to note that it's not possible to use the same frame for the left and right eye. The scene needs to be transformed and rendered twice using two view matrices producing two slightly different looking frames. This two different frames are then stitched together into the final framebuffer. The HTC Vive and Ocolus Rift HMD is just a monitor hardware wise with a default framebuffer OpenGL can render into.

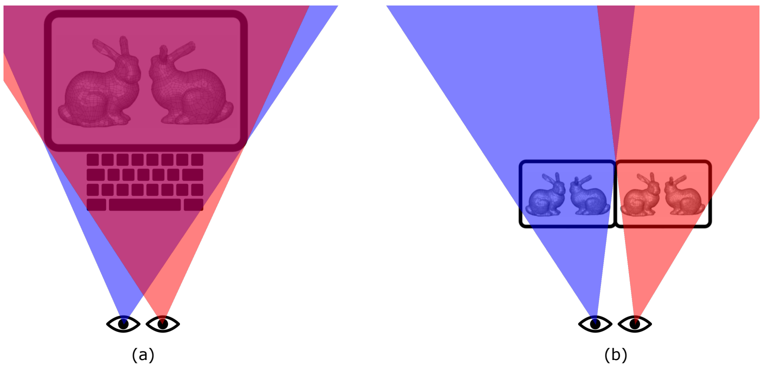

Figure 3.4: (a) Monoscopic rendering. (b) Stereoscopic rendering.
The two render frames in (b) look similar but are slightly different.

On top of the monitor are two Fresnel lenses for the left and the right part of the framebuffer. With WebVR, the HTC Vive and other VR devices can be accessed for information like width and height of the framebuffer, the eye offset, position and orientation of the HMD. With this information, the two required view matrices for the stereoscopic rendering can be constructed. The orientation is a quaternion and can be transformed into a 3x3 rotation matrix. The position is a 3D Vector. The eye offset is a float describing the offset of the lenses to the HMD position. The position and rotation provided by WebVR are relative to some VR world space. The VR origin and orientation is therefore set as the rendering world origin and orientation for the sake of simplicity. The eye offset is added to the HMD position as a scaled left or right view space vector. That means using a unit 3D vector with x component either -1 or 1 for left or right (as is defined by OpenGL), scale the unit vector by the eye offset and transform it by the inverse 3x3 rotation matrix from view space into world space. This is required, since the HMD position is in world space and the offset needs to be done to the left or right with respect to the orientation of the HMD. With the two offset positions, the two view matrices are ready for rendering. The projection matrices are available through WebVR since those can only be calculated knowing the resolution of the monitor of the HMD.

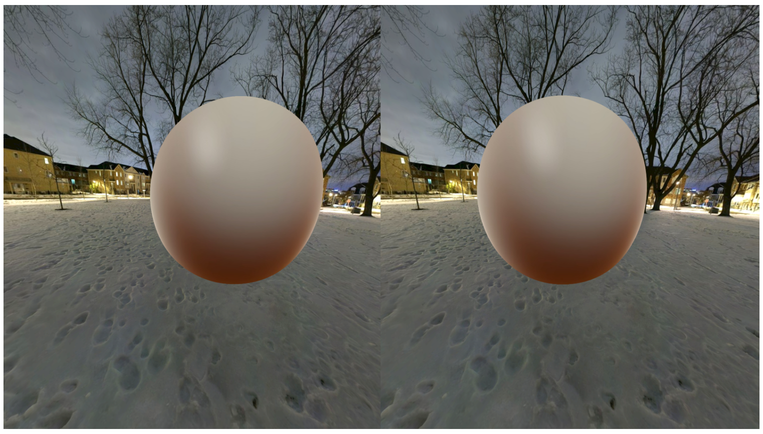

Figure 3.5: The stereoscopic rendering produced by the described method. The view offset is based on the eye position and is distorted due to two different projection matrices.

Therefore, the main render function was extended to draw the scene twice using the two compiled view matrices and restricting the drawing area using `glViewport` to render into the left or the right part of the framebuffer (Figure 3.5).

3.1.2 User Interface and Tooltips for VR

After the rendering was extended for VR, sculpting itself wasn't functioning in VR yet. For this, a proper user interface had to be created (Figure 3.7). The first step is to render the controller with the HTC Vive controller model provided by SteamVR using transformation data provided by WebVR. Input data such as button press or trigger pressure can be polled as well. To make the most use of this, all buttons on the controllers are binded to some functionality (button names Figure 3.6). The most used buttons are the hair triggers on both controllers. The right hair trigger starts a brush stroke. Its intensity is defined by the pressure on the hair trigger. The left hair trigger toggles between the current active brushing tool and the smoothing tool. Since smoothing is a very essential workflow in digital sculpting, the decision was made to bind this quick change on this easy to access button. Undo and Redo was bound to the left and right application menu button, another very easy and

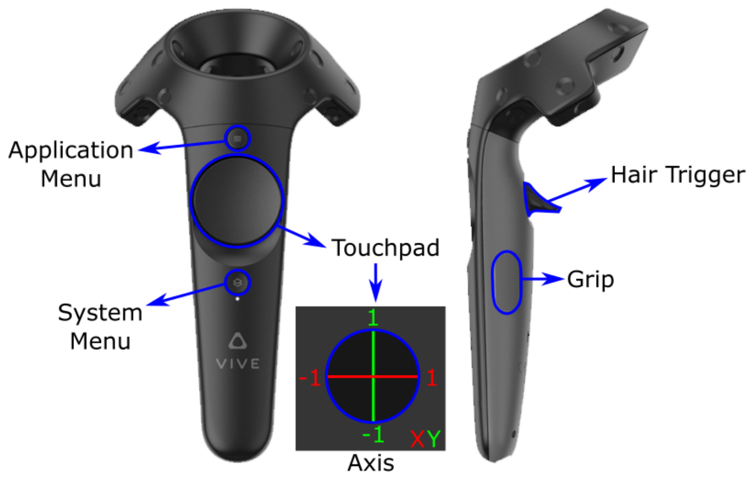

Figure 3.6: The button names on the HTC Vive hand controller.

fast to access button group on the controller, since it's is an other often used functionality in the sculpting workflow. The right grip button is used to move the object. Moving and rotating the right controller while the grip button is pressed will move and rotate the object accordingly. Holding both grip buttons while moving the controllers from or to each other will scale the object. The System Menu button is reserved by Steam. The Touchpad buttons are mostly for utility functions like switching render materials, toggling wireframe rendering, toggle sculpt symmetry etc.
For learning the key bindings, floating tooltips around the controllers are rendered. This tooltips are connected with a line to the according buttons. After learning the bindings, the user can turn of the tooltips by one of the Touchpad buttons. A beam is rendered from the right controller to indicate the pointing direction. A tool panel is rendered above the left controller. Pointing the right controller beam on the tool panel lets the user select a tool. There are color indications on the tool panel for a selected tool and a hovering tool. The design of the user interface with the tool panel mimics the user interfaces of most VR design application, since it is a very intuitive and interactive way to work in VR. The icons for the tool panel were used from the open source application Blender [14].

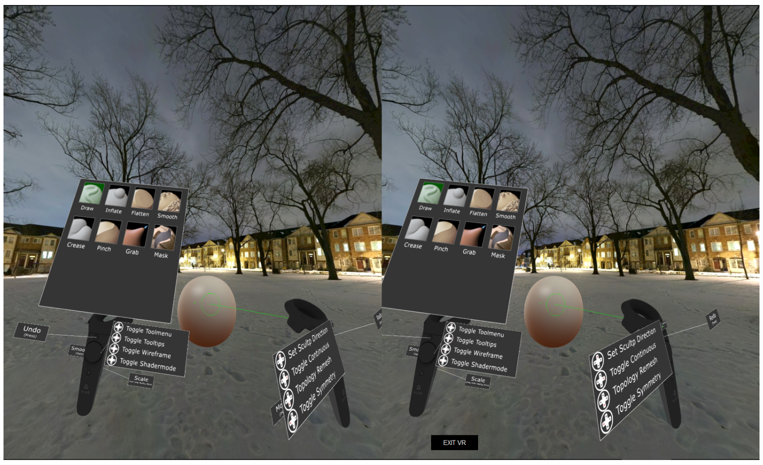

Figure 3.7: The user interface for VR with tool panel and tooltips.

3.2 Haptic interface for SculptGL

The task was to implement an interface for the Touch in SculptGL. The problem was, that device drivers were only provided for windows. Furthermore, the OpenHaptics API has only bindings in C++ and Python. The final problem was to implement a force feedback rendering for non solid surface models.

3.2.1 Implementing the haptic interface

For the haptic backend (windows C++ application) to be able to talk to SculptGL, a WebSocket connection is used for the communication. The small WebSocket daemon *websocketd* was used since working with WinSocket is rather tedious. The general concept is shown in Figure 3.8. The haptic backend opens a WebSocket on port 6001. The JavaScript application then connects to the backend. The communication had to happen in both directions, from sculpting to haptic and other way around. The haptic needs to tell the sculpting constantly about the device position and button state on the pen. This happens 60 times per second. The sculpting tells the haptic when a collision occurs and sends the calcu-

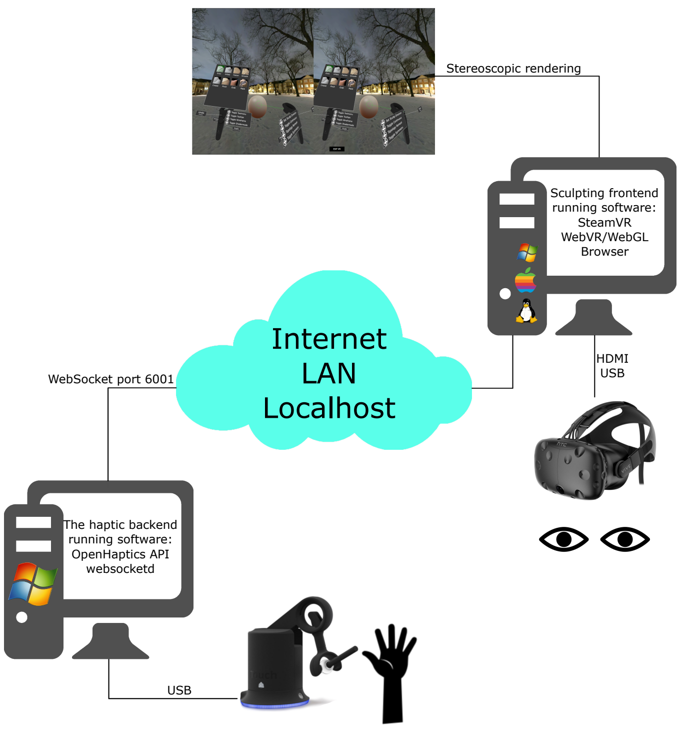
Stereoscopic rendering
Sculpting frontend running software: SteamVR WebVR/WebGL Browser
Internet LAN Localhost
WebSocket port 6001
HDMI USB
The haptic backend running software: OpenHaptics API websocketd
USB

Figure 3.8: WebSocket is used to connect the haptic and the sculpting applications.

lated force vector to the backend. The force feedback rendering is done on the JavaScript side since it's faster to send a force vector rather than the surface information over WebSocket (especially with sculpting, as the surface information is constantly changing). In order to make the WebSocket connection work in both directions, multi-threading is used, because otherwise the listening (happens irregularly on collision) would block the sending (constantly position and button state). Since OpenHaptics itself is highly multi-threaded, some care had to be taken not to clash. To get the haptic device information, a callback function is defined and given to OpenHaptics. The callback is called synchronized and it is save to poll device data like position, rotation and button state. It's also possible to query device information and start a calibration of the device, which is done on startup of the backend. A simple communication protocol was defined (Table 3.2). So for instance, a message which the haptic backend is constantly sending to the sculpting could look something like that: `"P:1.2151231,32.1231341,21.2132411"`. The `P` tells it is a position message, followed by three comma separated floats. On error, the the sculpting tries to initialize the haptic again. All Errors, Warnings and Messages (`E,W,M`) are printed in the browser developer console.

Message index	Message meaning	Message parameters
`P`	Position	Float x,y,z
`W`	Warning	String
`M`	Message	String
`E`	Error	String
`B`	Button	Integer

Table 3.2: The message index for the WebSocket communication protocol.

The last problem was the calibration of the haptic 3D space and the sculpting 3D space, which is basically the VR space. Unless the Touch is rotated in the perfect way, the two orientations are not aligned. Therefore, a simple calibration approach is presented. The origin calibration is done by positioning the VR main controller on the position where the haptic origin should be. The rotational offset is calculated by putting the Touch pen into the resting socket and holding the VR main controller on top of the touch aligned to direct in pen direction. The rotational and positional offset are stored and applied to the haptic position. The scaling of the spaces is adjusted and is constant. To indicate the haptic device pen tip position, a simple sphere is rendered.

3.2.2 Implementing the force feedback rendering

After the haptic device is calibrated and its movement is properly rendered in VR, it was time to add the force feedback rendering. So, two main problems had to be solved. First, a collision check with the sculpting object and the haptic sphere must be done. Second, a stiffness factor needs to be calculated in order to render force feedback.
To check if the mesh is colliding with a sphere, the naive

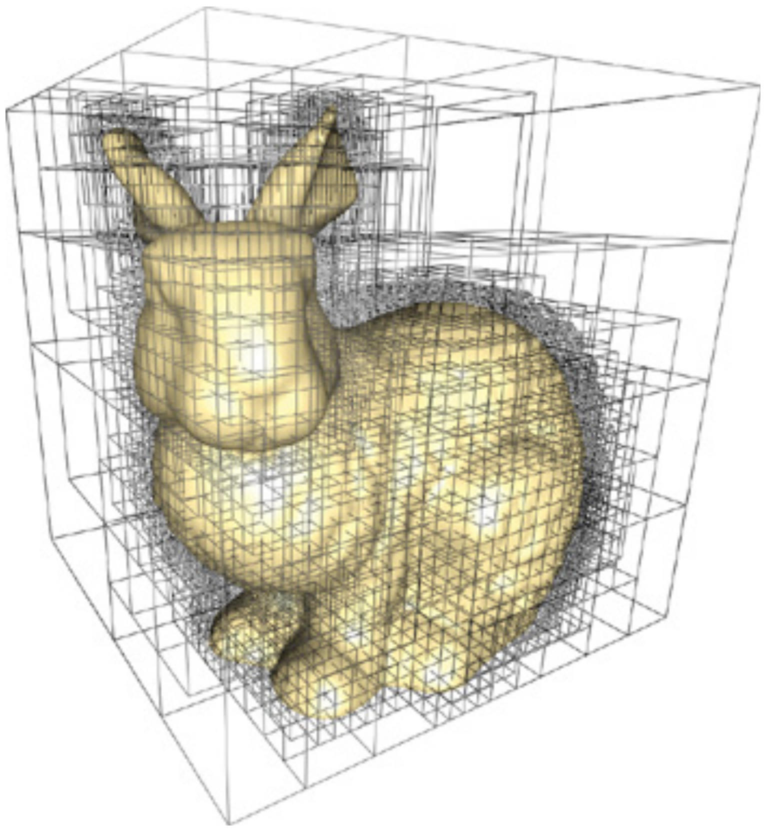

Figure 3.9: A stanford bunny stored inside an octree. This data structure can be processed very fast for collision checks with high resolution meshes (Image by NVIDIA).

approach is to check every triangle of the mesh against the sphere. This would totally destroy the realtime performance, especially since sculpting involves very high resolution meshes. A better way is to use a data structure like an octree for this (Figure 3.9). The octree is used to partition the mesh triangles. It has eight children in each internal node and the leaves store the triangles in its bounds. Therefore, the collision check is now checking the octree boxes against the haptic sphere, traversing the tree down and determent a much smaller fraction of triangles that still needs to be checked as before. With this, checking collision is very fast, but since the mesh is a surface only representation, the case where the haptic sphere is inside the mesh needs to be handled. With the current collision check, this case is not

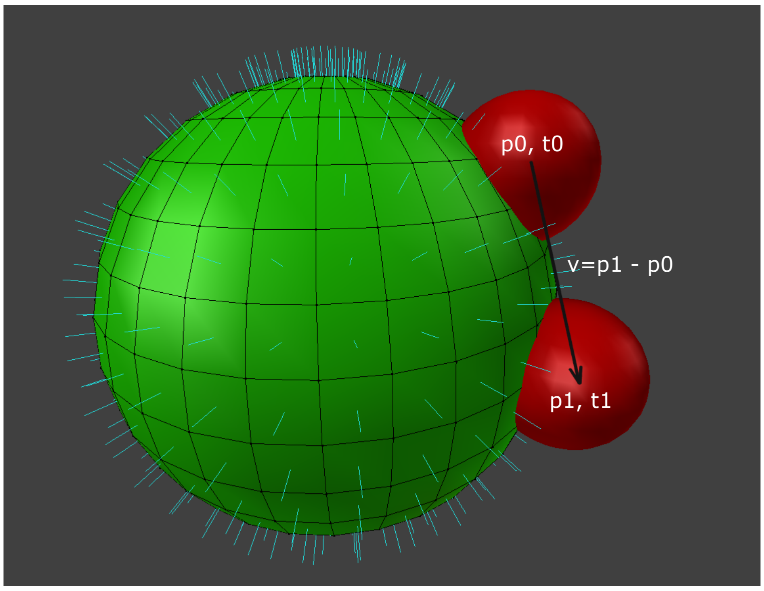

Figure 3.10: This figure shows the first two frames of the haptic simulation. At time t0, the position p0 is received by the haptic backend. At this moment, no frame is rendered. At t1, the position p1 is received. The velocity is calculated and the stiffness is accumulated. At this time, the simulation is rendering the situation at t0.

covered. An additional step needs to be done. To determine, whether the sphere is inside or outside, a raycast is performed from the sphere position into a random direction. If the amount of hit-points with the sculpting mesh is even, then the sphere is outside, if it's odd then the sphere is inside. This collision check is again very fast with the use of the octree. Now, the force direction can be calculated by taking the weighted average of the normals from the collision triangles and normalize the result. The normals are stored inside the normalbuffer of the object. The force itself is calculated by multiplying the force direction with the stiffness factor.
In order to determent a stiffness factor, the simulation is one step ahead of the rendering and it stores the values from the current and the last haptic update cycle. Having the rendering lagging behind one frame isn't noticeable though and is a common method. With the timestamps when the two different positions were measured, a velocity can be calculated (Figure 3.10). If the sphere is outside the mesh, the

stiffness factor is zero. On collision, the application starts to accumulate the stiffness based on the velocity. Since the stiffness should increase the most when moving deeper into the object and not along the side, the velocity is weighted by the dot product of the velocity direction and the hit normal. The dot product is the cosine of the angle between those normalized vectors, and it's a good weight for the stiffness, since moving parallel to the hit-normal (0 degree angle) is where the cosine is 1 or -1 and moving orthogonal to it is where the cosine is 0. This is the same behavior how the stiffness needs to be simulated.

Chapter 4

Evaluation

A user study was conducted to test the implementation of HaptiSculptVR. A large part of the participants were visitors of the Maker Fair Hannover 2018. The user could try HaptiSculptVR using only the HTC Vive Controllers or with the Touch 3D stylus for haptic feedback.
At first, the users were asked the following questions:

1. Age group
   (a) Younger than 14 years
   (b) 14-18 years old
   (c) 19-25 years old
   (d) 26-35 years old
   (e) 36-45 years old
   (f) 46-55 years old
   (g) Older than 55 years
2. Interest and experience in IT
   (a) None
   (b) Just everyday use of computer / smartphone
   (c) Private interest and experience
   (d) Professional interest and experience

3. Interest and experience in VR
    (a) None
    (b) Only tried
    (c) Private interest and experience
    (d) Professional interest and experience

After the first round of questions, the user had time to try the application. The user had a brief instruction about the concept of the tool but no in depth explanation about the handling or the control layout. This was intentional in order to get an overview whether the handling and the UI tooltips were intuitive and helpful. Only if the user was stuck, further instructions were given. After enough time, a second interview was held with the user. In total, 20 people tested HapticSculptVR under the mentioned conditions.

4.1 Results

The results of the age group evaluation (Table 4.1) shows, that mostly younger people were interested in testing a VR HMD. Older people showed less interest and either didn't want to test at all or didn't want to participate in the survey.

Age group	Amount user
Younger than 14 years	8
14-18 years old	2
19-25 years old	5
26-35 years old	3
36-45 years old	2
46-55 years old	0
Older than 55 years	0

Table 4.1: Evaluation result - Age group

The second question was intended to find out, whether the user had any relation to any IT topics or if any interest existed at all. The purpose was mainly to be able to better assess and communicate with the user. The results (Table 4.2) shows that all of the participants had at least some knowledge about IT, the majority was involved with IT beyond the everyday use of a smartphone.

Interest and experience in IT	**Amount user**
None	0
Just everyday use of computer / smartphone	5
Private interest and experience	9
Professional interest and experience	6

Table 4.2: Evaluation result - Interest and experience in IT

The last question before the user could test HaptiSculptVR was about previous experience with VR. It was interesting to see people having their first VR experience and how they handled it. On the other hand, it was also interesting to hear what other VR sets are used and how they compare to the HTC Vive. The results (Table 4.3) shows that the majority already tried VR and some even own a VR set themselves. One participant uses Unity Game Engine to experiment with VR and plans to create a small game.

Interest and experience in VR	**Amount user**
None	4
Only tried	11
Private interest and experience	5
Professional interest and experience	0

Table 4.3: Evaluation result - Interest and experience in VR

After the user tried HaptiSculptVR, a final interview was held. The length and details of the interview depended on the interest and experience of the user, but this base questions were always asked:

1. What did they like more: Digital sculpting with VR or without VR?
    (a) If they like it with VR more: Do they think that using VR is an overall improvement to digital sculpting?
2. Are the tooltips helpful? Did the user notice or ignore them?

   (a) If not: What is there suggestion to improve accessibility to the control layout?

3. Was the control layout useful or hindering for the sculpting workflow?
4. What did they like more: Digital sculpting with or without haptic feedback?

All users agreed, that digital sculpting with VR is much easier and faster, and that immersing into the scene gives a clearer perception of the 3D object. This is due to the stereoscopic view into the 3D scene. No user had any sculpting experience and still managed to sculpt good results.
Overall, the user had problems with the user interface and the control layout. Experienced users were able to adapt to the workflow very quickly while users with no experience struggled to understand the concept. This can be explained by the excitement of trying VR for the first time and no experience with digital sculpting. Some user had issues reading the tooltips and one user could not see the red tooltip indicators. User, who also agreed to sculpt with haptic feedback told that the immersion was increased and the depth perception was better.

The user study helped to resolve some problems and misunderstanding with the user interface. Some minor requests like a background image and more different materials for the working object have been realized after the user study. Overall, the result of using VR and haptic with digital sculpting was positive.

Chapter 5

Conclusion and future work

5.1 Conclusion

When comparing the results of digital sculpting with and without VR elements in the process, it's pretty clear that using a HMD and a haptic feedback device is a much more intuitive way to interact with a virtual environment. The stereoscopic rendering provides the user with a better depth perception. For example, the depth of field blurring effect is usually a monoscopic rendering post processing step. In stereoscopic rendering, this effect happens naturally. Adding haptic feedback to this experience gives the user a way to feel the object surface. Research showed that haptic feedback is also a great way for visually impaired people to interact with virtual environments. Sculpting is a very fitting work to be done inside VR, not only because brush strokes and object rotation and scaling can be performed by very natural hand movement. Whether it is the rendering or the haptic feedback, the ultimate goal by Sutherland to enable all senses inside a virtual environment.

5.2 Future work

Although the core features of SculptGL were ported to VR and binded to the UI, there is one great feature which would increase the value of this application by a lot. SculptGL comes already with a vertex painter. A port of this feature to VR would make this a great two in one tool. In general, there is room for improvement for the VR user interface. The tool panel could be extended to clear the current scene or import / export meshes. To improve the force feedback simulation, some approaches as presented in Section 2 could be implemented. A voxelization of the collision part or a scalar force field for the object are two reasonable approaches for this.

Bibliography

[1] S. Ginier, "SculptGL - A WebGL sculpting app." `https://github.com/stephomi/sculptgl`, 2013.

[2] T. Mazuryk and M. Gervautz, "Virtual reality - history, applications, technology and future," 1999.

[3] I. E. Sutherland, "The ultimate display," in *Proceedings of the IFIP Congress*, pp. 506–508, 1965.

[4] F. P. Brooks, Jr., M. Ouh-Young, J. J. Batter, and P. Jerome Kilpatrick, "Project GROPEHaptic displays for scientific visualization," *SIGGRAPH Comput. Graph.*, vol. 24, pp. 177–185, Sept. 1990.

[5] R. J. Stone, "Haptic feedback: A brief history from telepresence to virtual reality," in *Proceedings of the First International Workshop on Haptic Human-Computer Interaction*, (Berlin, Heidelberg), pp. 1–16, Springer-Verlag, 2001.

[6] O. Lahav and D. Mioduser, "Haptic-feedback support for cognitive mapping of unknown spaces by people who are blind," *Int. J. Hum.-Comput. Stud.*, vol. 66, pp. 23–35, Jan. 2008.

[7] Mozilla VR Team, "A-Painter - An Open Source VR Drawing Tool." `https://aframe.io/a-painter/`.

[8] T. W. Sederberg and S. R. Parry, "Free-form deformation of solid geometric models," *SIGGRAPH Comput. Graph.*, vol. 20, pp. 151–160, Aug. 1986.

[9] J. Lawrence and T. Funkhouser, "A painting interface for interactive surface deformations," *Graph. Models*, vol. 66, pp. 418–438, Nov. 2004.

[10] A. S. Snibbe S. and V. B., “Springs and constraints for 3D Drawing,” 1998.

[11] F. Dachille, H. Qin, A. Kaufman, and J. El-Sana, “Haptic sculpting of dynamic surfaces,” 03 2000.

[12] K. T. McDonnell, H. Qin, and R. A. Wlodarczyk, “Virtual clay: A real-time sculpting system with haptic toolkits,” in *Proceedings of the 2001 Symposium on Interactive 3D Graphics*, I3D ’01, (New York, NY, USA), pp. 179–190, ACM, 2001.

[13] R. Blanch, E. Ferley, M.-P. Cani, and J.-D. Gascuel, “Non-Realistic Haptic Feedback for Virtual Sculpture,” Research Report RR-5090, INRIA, 2004.

[14] “Blender - Open Source 3D Creation Tool.” `https://www.blender.org/`.

Typeset February 27, 2019