Dalhousie University
Faculty of Computer Science

VR Voxels – Virtual Reality Voxel Modeler Final Report

Johna Latouf

Date Submitted: August 4, 2017

Abstract

Traditional 3D modeling software requires designers to plot and manipulate 3D meshes in a 2D viewing environment. Virtual Reality hardware offers users the opportunity to design and edit meshes in 3D space. This report outlines the development of a 3D voxel modeling application for the Oculus and Leap Motion systems. It describes the steps taken to create editing tools, the challenges associated with developing for VR, and analyzes gesture based drawing methods. It concludes that the VR environment is ideal for 3D modeling, but some aspects of motion tracking should be improved. It recommends including full-body motion tracking.

Table of contents

List of figures

1 Introduction

Virtual and augmented reality environments offer new tools and new challenges for artists and hobbyists. This application, VR Voxels, was developed for the purpose of studying 3D modeling in a virtual reality environment.

1.1 Requirements

The proposal for VR Voxels included these key requirements:

- The ability to draw, delete, and texture voxel meshes using the Leap Motion hand tracking system.
- A touch-based user interface that allowed the user to apply filters, save, load, and export voxel models and textures.
- Voxel selection and the ability to isolate selections.
- Basic voxel terrain generation.

2 Discussion

2.1 Tools, Libraries, and Tutorials

VR Voxels is written in C# and built using the Unity 3D [1] game engine and Oculus Rift virtual reality headset. It includes several plugins and open source tutorial material. This application uses the Leap Motion hand tracking system for interaction [2]. The UI module is used for user interface interaction, the attachment module is used for brushes and the hand-held menu, and the interaction and detection modules are used for gesture detection. About half way through production of VR Voxels, Leap Motion released its new Orion Beta interaction module [3], but that is not yet incorporated into the application. The Leap Motion camera is mounted on the Oculus headset and uses raw image data to distinguish hands and arms within view [4]. In Unity, the Leap Motion plugin applies the hand positions and gestures to a set of hands built with rigid bodies. The OBJ export functionality is done directly through C# using Stefan Gordon's OBJ parser, which is licenced under the MIT licence and writes mesh data to OBJ text files [5]. The basic voxel construction is modelled in part from AlexStv's Voxel Tutorial [6] which is free for all uses.

2.2 Voxel Mesh

The voxel mesh design is based partially on the AlexStv tutorial for voxel terrain generation in Unity [6]. In this tutorial, voxels are split into chunks and different voxel types are created with pre-assigned texture tiles (a Grass voxel is used throughout the tutorial). Voxels are laid out with a noise algorithm.

The tutorial outlines many ways to prevent excessive triangle creation in the voxel mesh. Whenever the mesh is updated, an algorithm checks the neighbours of each voxel to determine which ones are filled in and which ones are empty. If a pair of neighbours are both filled, there is no need for triangles to be drawn between the meshes, so they are removed. Voxel meshes are drawn in chunks of 163 voxels to accommodate Unity's

triangle limit. Voxels are one cube unit in size. For this application, only two types of voxels are required, full and empty, but textures must be applied dynamically and the user must be able to make changes interactively.

2.3 Voxel Canvas and Platform

Maintaining a high framerate is crucial in a VR environment [8], but framerate suffers as triangles are added to meshes in a scene. Even with meshes that exclude overlapping triangles, a model that is too large can greatly slow performance, so VR Voxels limits the user to 80^3 voxels.

Because voxels represent three-dimensional pixels, the modeling environment resembles a drawing canvas more than an open modeling space. The parent object that contains each voxel chunk is named VoxelCanvas and can be transformed without affecting the scale and position of the child chunks. The canvas itself includes a transparent grid that indicates where each voxel can be placed. The grid prefab is a cube split into three objects consisting of parallel walls. A script is applied to each set of walls that scales the tiled texture to match the number of voxels in its respective dimension.

The Oculus user's head movements control the orientation of the scene camera, but they do not translate the camera. There are many ways to translate a user around a scene, including teleportation (allowing the user to jump from one spot to the next with a button or gesture), and moving the scene around the user. Teleportation would be too sudden and may disorient a user trying to focus on small details. For this project, the ability to move, scale, and rotate a voxel creation around was prefered.

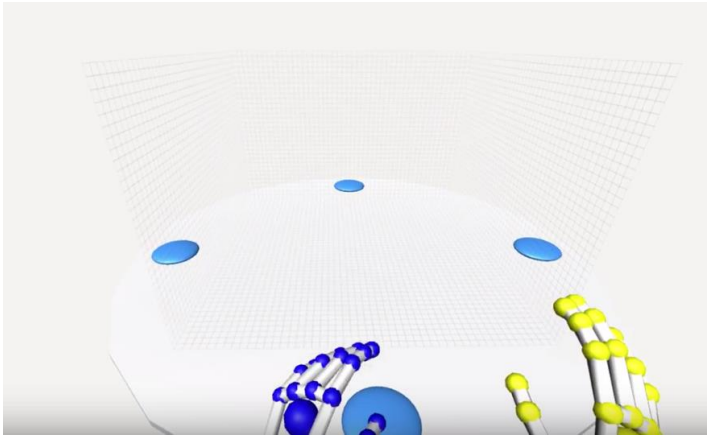

Figure 1 Platform and Voxel Canvas with grid

To make the navigation more intuitive, the voxel canvas is placed on a platform that resembles a potter's wheel which may be familiar to the user. The Platform object is a white, free floating disk with four blue spherical handles. Using a sphere collider, each platform handle informs the Voxel Editor object whether a thumb tip, index fingertip, or palm has entered or exited a handle. The material on the handle turns red to indicate fingertip collisions, yellow to indicate palm collisions. During a fingertip collision, if the user pinches or grabs a handle, the platform will translate along with the hand. If both hands trigger this effect, the platform will scale to fit the distance between the hands. If a palm collision is detected and the palm is opened and facing downward, the platform will

rotate along with the hand. The rotation effect requires measuring the angle between the platform's forward vector and the palm at each frame and applying the difference to the platform's rotation. This affects works well for three of the four handles, but Unity's Vector3.Angle function returns values in the range of 0 to 180, causing one handle to move back and forth repeatedly. A second angle must be measured between the hand and the platform's right vector to detect when that malfunctioning handle is activated so that the angle distance is calculated correctly.

2.4 Editing Voxels with Gestures and Brushes

The Voxel Editor class operates two state machines to facilitate drawing and editing. One state machine handles drawing modes. These include auto draw and auto delete, release draw and delete, texture painting, and selection. The second state machine handles brushes. Because each voxel represents one unit, and each voxel mesh is parented to the voxel canvas, editing individual voxels with game object positions can be done using Unity's InverseTransformPoint function. The position of the drawing object in relation to the voxel canvas is obtained as a vector, each float value in that vector is cast as an integer, and the x, y, z integers are used to edit the corresponding voxel.

The pinching gesture is easy to distinguish making it useful for drawing by hand. While in auto-draw mode, the Voxel Editor class checks both pinch indicators and changes the voxel type and texture tiles for individual voxels in the position of the pinch detector object. In release draw mode, voxels are only updated after a pinch is released. The position of the pinch detector is saved before the pinch is released and used to place the voxel update, to prevent unexpected results from a pinch detector moving while the user's fingers release. Full voxel painting uses these same gestures.

Painting individual faces uses a pointing gesture. A detector indicates when the index finger is extended and draws a raycast [7] along the finger direction. The raycast measures the angle in comparison to the face's normal direction to determine if the user is trying to paint a face within a short distance of the fingertip. The raycast is drawn from slightly behind the fingertip detector, so that the user can touch/intersect with the face.

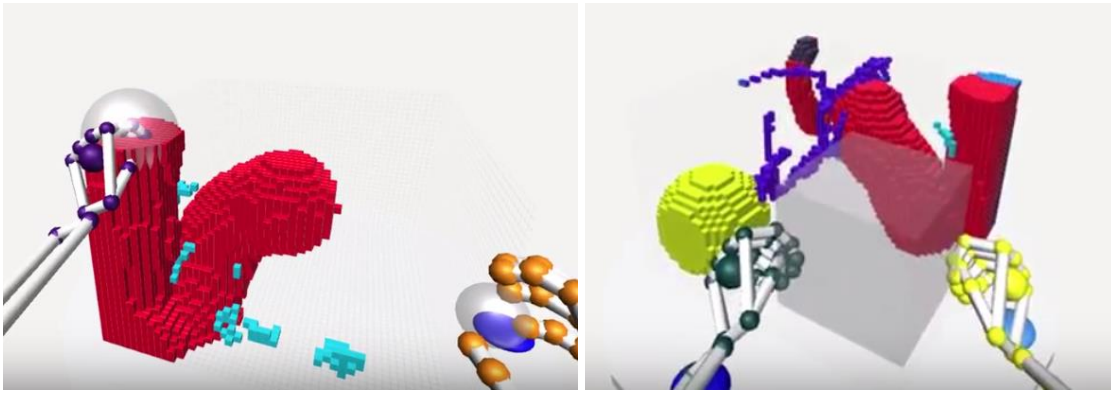

Figure 2 Sphere and cube brushes

There are three available brushes: pen, sphere, and cube. The pen brush draws individual voxels that are touched with the pen tip. Pens are attached to the hands using the Leap Motion attachment plugin, and are activated with the user pinches or grabs them. They are slightly less stable than the pinch gesture because they occasionally wobble when the

hand object rotates unexpectedly, but they provide the user with a clearer visible cue for exactly which voxels are being edited. In release draw mode, the pen brush tips must be held together to edit single voxels.

The sphere brush mode attaches spheres to each Leap Motion hand in auto draw mode. The spheres are resizable with a combination of gestures. The brush hand must be open, and the opposing hand must be pinching within the sphere to scale it. The sphere brush checks each voxel in the canvas and edits those within the radius of the brush's centre. A pinch gesture activates drawing. In release draw mode, there is one sphere attached to the right hand. When both hands are pinching, the brush scales along with them. When both hands release, a single sphere of voxels is edited. The cube brush operates with similar scaling functions, but rather than checking all voxels for those within a radial measurement, the Voxel Editor only edits those between the cube vertices.

Painting with the pen, cube, and sphere brushes uses the same gestures as finger painting, although the pen brush does not require a pointed index finger for face painting. When a sphere or cube brush is used for full voxel texture painting, the texture is updated within the radius of the sphere or within the measurements of the cube. When the face paint option is selected, the sphere or cube brush moves to the fingertip on each hand. For each brush type, a ray is cast from the brush position to the voxel mesh and face in in a single direction within the brush's range are painted.

The selection tool applies instances of a quad with a selection texture to voxel faces. A dictionary collects each voxel face selected. Before edits are made to the mesh, a function is used to check first if that dictionary is empty, and if not, it checks if the voxel is included in the selection. The quad instances have a strong slowing effect on the framerate, so they are currently limited to individual pinch selections. One possible solution to this would be applying vertex coloring to the voxel mesh to indicate selections, but that was also found to negatively affect framerate, and was more difficult to distinguish visually.

2.5 Filters

VR Voxels has four simple filters available. There are two smoothing functions, Flat Smooth and Round Smooth. Each of these accepts a float value as an argument, which can be adjusted using a slider in the handheld UI menu. This value determines the threshold for each function. The Flat Smooth filter checks each voxel's neighbours on all sides to produce a sum and compare that value to the threshold to determine whether to swap the voxel from full to empty or vice versa. A threshold of about 0.5 will remove stray voxels and create a smooth, structural appearance. Similarly, the Round Smooth function checks all neighbours, including diagonal neighbours, and compares the resulting sum to the threshold value. This function rounds out corners and clears stray voxels. In each, the texture assigned to new voxels is chosen from the textures assigned to neighbouring voxels.

The noise function is designed to apply more noise to areas with a high volume of full voxels, while sprinkling full voxels throughout the canvas. For each voxel, a threshold is taken as an argument from a UI slider and combined with a random value is compared against a sum of neighbouring voxels. Then another random number, combined with the

threshold value, is used to fill or empty voxels regardless of neighbours. It is possible to split this functionality up in the future to allow for a contained noise and a random noise. Texture tiles are chosen from neighbouring voxels or from the current drawing tile.

A wave filter uses a combination of the round smooth and a sine wave function to produce a uniform, round wave effect on the top of existing voxel mesh. The sine function builds dips and hills across the mesh in the y direction using the x and z positions. The smooth function softens this wave to produce a rounded set of bumps.

2.6 User Interface

VR Voxels combines floating UI and hand-attached UI elements. The New Model scene uses a minimal UI layout with three sliders to allow the user to set a canvas size. In the drawing scene, a floating Head's Up Display (HUD) object is available in the upper right corner of the Oculus view that indicates the drawing tool options and selected texture tile. Multiple iterations of this display were attempted, including a display that followed the Oculus camera or the drawing platform, but these designs were distracting and difficult to view.

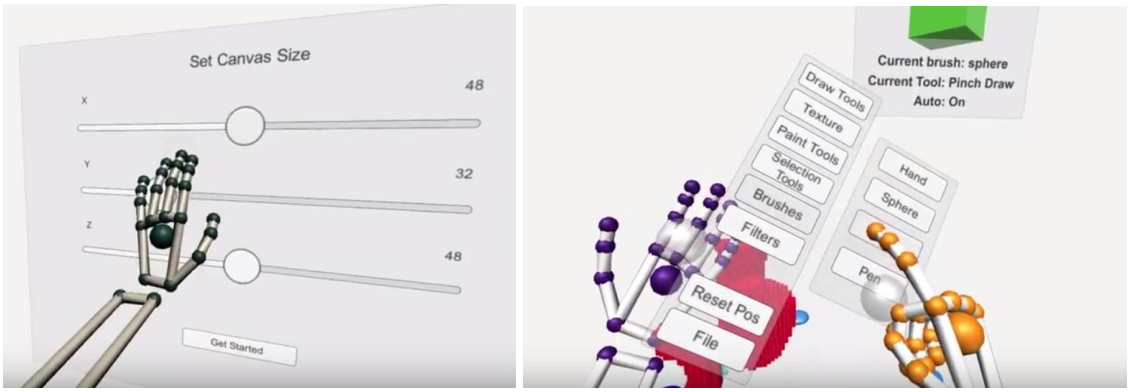

Figure 3 New Model, handheld, and HUD user interface

When the left hand is flat and the palm facing the Oculus camera, a handheld UI display appears. This display combines the Leap Motion UI module with the Leap Motion attachment and detection modules. A palm detector measures finger extension and palm direction and activates attachment game objects. The texture submenu displays a UI panel with the current texture sheet. When this submenu is activated, the TileSelect script instantiates a simple circle button for each tile. This script is designed to be easily adjusted for texture sheets of different sizes.

3 Analysis

3.1 Gesture Control Options

While the pinch gesture is easy to work with, it is important that it is not overused. In some cases, attempting to resize a brush can cause accidental changes to the voxel drawing. Pinch release detection must be deliberate or it will not be detected. The open palm gesture and palm direction detectors work well for displaying hidden items, such as the UI toolbox. The drawing platform's movements are not very smooth. The user must acclimate to the speed of the platforms movements, otherwise the hand moves too

quickly and thus outside of the platform handle. A platform movement system that does not require direct contact may work better.

The Leap Motion's accuracy varies. If there are any reflective surfaces or objects in the environment that could confuse the detection system (such as a dangling Leap Motion cord) the user will experience undesirable results. When the system does not detect a hand, the objects associated with that hand deactivate in the scene and occasionally do not activate correctly. This application attempts to alleviate that by setting attached objects to active whenever the Voxel Editor state machine indicates they should be available.

3.2 Drawing Environment

The Oculus environment is a valuable tool for 3D modeling. Because the user can interact directly with objects in all three dimensions without relying on a 2D viewport, it is easy to place adjustments in desired coordinates. The framerate suffers when meshes are larger than 64^3 voxels, or when more than a few dozen selection highlight textures are visible in the scene. As mentioned above, the platform controls should be more responsive. The pinching gesture is used for drawing and resizing brushes, which occasionally causes unwanted voxels to be drawn. A wider variety of gestures, or a Boolean to turn off drawing when resizing is taking place my fix this. The sphere brush can slow framerates because it must check every voxel. It may be possible to take advantage of faster algorithms, such as the seed-filling algorithm [9], to prevent lag.

3.3 UI and File Options

The UI uses the default Unity appearance. Leap Motion's UI plugin is responsive, but the handheld layout is bulky and may be improved by being spread out around the hand. Anything attached to the hand will deactivate if the hand is not within the field of view, so the size of this UI must remain as compact as possible. The file I/O options can only save files to a hardcoded path. A VR file dialog and keyboard should be added to give users more flexibility.

3.4 Outstanding and Future Work

The raycast directions are compared to the mesh normal directions for texture painting tools. These measurements are not always accurate, and it can be difficult to paint faces with precision when the voxel canvas is rotated. This system should be refined.
The platform rotates along the y-axis, but options for tilting the canvas along the x and z axis would provide the user with the option to view the top of the model at a more comfortable angle.

Several features remain outstanding. A simple sine wave filter is included, but terrain generation is not yet developed. Isolated views of selections are also not available, making detailed editing difficult. The user cannot upload custom texture sheets or import OBJ files.

In addition to brushes, drawing can be done with stamps. A stamp should be a free-floating object, not attached to the Leap Motion hand, that can be placed on the canvas to

produce a shape. When the canvas is rotated, scaled, or translated, the stamp draws voxels. This could be used to produce round, symmetric images.

4 Conclusions

3D modeling in virtual reality provides users with better views and spatial control than traditional 2D environments, but movement tracking systems like Leap Motion sacrifice precision. Brush attachments and procedural tools alleviate some of these control problems. Users are also limited by the lack of camera translation. Containing models on moving platforms that do not interfere with the mesh helps solve this, but it limits the scale of the model.

5 Recommendations

A more robust drawing platform should take advantage of line of sight or audio cues to rotate and translate a model. Because activities like 3D modeling do not require a great deal of space, this application may benefit from full-body motion tracking hardware that allows a user to walk around a model.

References

1 Unity. Unity Technologies, 2017

2 Leap Motion, 2017. https://www.leapmotion.com, Accessed: 1 August, 2017

3 A. Colgan, 'Getting Started with Unity Modules', *Github*, 2017. https://github.com/leapmotion/UnityModules/wiki, Accessed: 1 August, 2017

3 A. Colgan, 'How Does the Leap Motion Controller Work?', *Leap Motion Blog*, 2014. http://blog.leapmotion.com/hardware-to-software-how-does-the-leap-motion-controller-work/, Accessed: 1 August, 2017

5 S. Gordon, 'A WaveFront Obj Parser and Writer in C#', *Github*, 2017. https://github.com/stefangordon/ObjParser, Accessed: 1 August, 2017

6 AlexStv, 'Unity voxel block tutorial pt. 1', *AlexStv*, 2014. http://alexstv.com/index.php/posts/unity-voxel-block-tutorial, Accessed: 26, June 2017

7 'How can I detect which SIDE of a box I collided with?', *Unity3D.com*, 2012. http://answers.unity3d.com/questions/339532/how-can-i-detect-which-side-of-a-box-i-collided-wi.html, Accessed: 1 August, 2017

8 F. Wong, 'Performance Analysis and Optimization for PC-Based VR Applications: From the CPU's Perspective', *Intel Corporation*, 2016. https://software.intel.com/en-us/articles/performance-analysis-and-optimization-for-pc-based-vr-applications-from-the-cpu-s, Accessed: 1 August, 2017

9 A. Prior. '"On-the-fly" voxelization for 6 degrees-of-freedom haptic virtual sculpting.' *In Proceedings of the 2006 ACM international conference on Virtual reality continuum and its applications (VRCIA '06). ACM, New York, NY, USA, 263-270.* 2006. DOI=http://dx.doi.org/10.1145/1128923.1128966

Bibliography

J.Bloomenthal and B.Wyvill,'Interactive techniques for implicit modeling'. *SIGGRAPH Comput. Graph. 24, 2 (February 1990), 109-116*. 1990. DOI: http://dx.doi.org/10.1145/91394.914273G

S. Jang, W. Stuerzlinger, S. Ambike, K. Ramani 'Modeling Cumulative Arm Fatigue in Mid-Air Interaction based on Perceived Exertion and Kinetics of Arm Motion' *In Proceedings of the ACM Conference on Human Factors in Computing Systems (CHI 2017: 3328-3339), Denver, CO, May 6-11, 2017.* DOI: https://doi.org/10.1145/3025453.3025523

F. Weichert, D. Bachmann, B. Rudak and D. Fisseler. 'Analysis of the Accuracy and Robustness of the Leap Motion Controller', *Sensors 2013, 13, 6380-6393*. 2013 doi:10.3390/s130506380