Exploring Virtual Reality Interfaces for Modeling Digital Terrains

Ian Scilipoti

March 26, 2019

1 Abstract

Terrains are a central component of convincing 3D graphics. Various approaches exist to model and customize terrains. However, many of these approaches rely on significant background knowledge of geologic processes and complicated user interfaces. To solve this problem, we explore the use of virtual reality for terrain generation. In our system, users customize terrain through hand gestures. We develop two interfaces to explore this approach. The first interface allows users to stretch and pull the terrain surface similar to clay or fabric. The second interface allows users to push terrain material similar to the mechanics of a sandbox. Additionally, we developed an erosion process to smooth out terrains throughout user interactions in the sandbox model. The goal of our research is to determine which interaction scheme is more intuitive and effective for terrain generation. Our user study (N = 25), shows that the sand model is on average easier to use than the mesh model.

2 Introduction

Terrains central visual and structural component of games, movies and simulations. Terrains can be quite complex, however. Realistic terrains have a combination of large-scale features such as mountains and valleys as well as small scale features such as streams and boulders. Geological limitations restrict the domain of natural terrains. For example, Larsen and Montgomery [16] suggest that terrain hill slopes steeper than 35 destabilize quickly, triggering landslide events.

Many terrain generation tools exist that allow users to generate randomized realistic digital terrains. These tools rely on many common techniques to synthesize landscapes. One of the simplest techniques

is procedural patterns. Pseudo-random noise patterns such as Perlin Noise [9] can be used to synthesize random elevation data that resemble landscapes. Although this technique is relatively fast, it allows little customization of the type and layout of landscape features.

A second common technique is example-based modeling. These techniques leverage real elevation data taken from geologic surveys to generate realistic features. User specified low resolution guide terrains are used as input into matching algorithms that cut and stitch pieces of real elevation data together to create high resolution terrains that match the layout specifications of the guide terrain. Although this technique allows users to supply guide images to control layout of features, from what we have seen, this work does not focus on the ability to construct these guide images.

Geologic simulations are also commonly used for terrain generation. These simulations rely on models of geologic principles such as water flow, erosion and hill slope stability. Simulations can generate the most realistic results. However, large-scale simulations are often the most computationally expensive. Similar to example based modeling, guide terrains may be input into erosion models. However, since erosion processes have potential to drastically alter landscapes, specific details provided in the guide terrains will likely not be preserved.

From our perspective, all of the above techniques lack a convenient way for users to customize the shape of landscape features. The techniques do not focus on the direct shape, rather, they focus on speed, realism, or layout. In general, we believe the above approaches work well for generating large scale random landscapes, however, they permit little artist control.

A newer technique used for terrain generation is sketch-based modeling. In this approach, users may draw feature curves using mouse strokes that define large scale features of the terrain [10, 12]. Each curve may define a specific feature such as the silhouette of a mountain or valley. This technique is more suited to fine-tuned artist control. However, in order for a complete terrain to be designed, hundreds of curves must be specified by the user. To address this, systems have been explored that support a mixture of random procedural generation and feature curve sketching [11, 19].

Although terrain sketching provides increased artist control, the technique still comes with an inherent layer of complexity shared by the all of the terrain generation techniques discussed thus far. All of the tools discussed share the familiar windows, icons, menus, pointing (WIMP) interface. In other words, they rely on a mouse and keyboard as the main mode of interaction. Although point and click interfaces can be efficiently leveraged by experienced users to create finely detailed results, some argue that the style of interface does not lend itself to modeling 3D shapes and structures, especially for inexperienced users [14].

In this project, we present a new approach to modeling terrain. The approach relies on the gestural power of virtual reality (VR) systems. Users of our system are presented with a scaled down terrain visual-

ized within a VR headset. Users may articulate gestures using hand-held controllers to sculpt and shape the terrain in real time. Two interaction schemes were developed and implemented in this project. In the first scheme, the terrain is modeled as a stretchable surface. User may stretch, push, and pull the terrain using hand gestures to form mountains and valleys. In the second scheme, the terrain is modeled as a sandbox. Users may push material into hills as well as drag and drop material from one location to another.

The goal of this project is to determine which of these schemes is more effective for generating customized terrains. Additionally, we hope to determine the effectiveness of VR as tool for visualizing and interacting with digital landscapes. To accomplish this goal, we design a user study. In the study, users alternate using the two schemes to accomplish a goal of matching the starting terrain to a goal terrain.

We begin this paper by presenting background information on height fields (Section 3.1). Next, we examine some of the previous work in the field of terrain generation (Section 3). We then discuss methods used in our project beginning with general hardware and technical choices (Section 4.1). Next we present two interactions schemes developed for the project (Section 4.2). Next, we discuss an erosion algorithm used to keep our terrains geologically reasonable (Section 4.2.3). Finally, we discuss a user study that was designed to compare the two interfaces developed (section 4.3).

3 Related Work

We now examine previous work related to our project. We begin with a basic discussion of height fields and polygon meshes.

3.1 Background

Height fields are a common standard for storing terrains [1]. Height fields are rectangular grids of discrete vertical displacement values. In the context of virtual terrains, each cell in the grid may store the elevation of the terrain at that point (Figure 1). Height fields provide a memory efficient and general data structure for storing terrain data. The downside of this approach, however, is their inability to store features such as caves and overhangs. This is because each cell in a height field only specifies one elevation value in the xy-plane where z represents the vertical axis. Most terrains can be sufficiently represented without the presence overhangs and caves, however. In this project, we restrict generated terrains to the subset of terrains that can be stored as a height field without loss of features. This restriction simplifies the domain of possible terrains without significant loss of potential for generating realistic terrains. Additionally, this restriction allows industry standard height fields to be easily exported from our program.

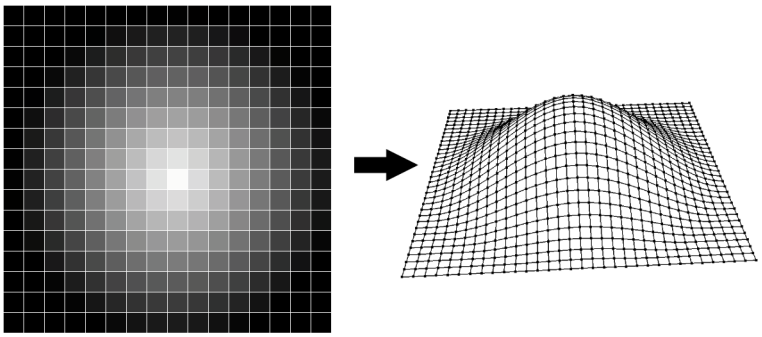

Figure 1: An example height field and the resulting mesh after displacement. Brighter values in the height field indicate higher elevation. Conventionally, height fields store elevations in a 0-1 range.

Polygon meshes are a common standard for storing 3D objects. Meshes are composed of a list of polygons (typically triangles) P. Each polygon contains a list of vertices, V, a list of edges, E. 3D objects are defined by a series of vertices connected by edges and polygons to form a surface. Meshes are a common choice because they are relatively lightweight to store. Additionally, meshes are easy to render on modern graphics cards.

3.2 Mesh Sculpting

Next, we will discuss relevant terrain generation work. We begin with mesh-sculpting interfaces. Next, we discuss VR modeling interfaces. Finally, we discuss various artist friendly and customization focused terrain generation techniques.

Polygon meshes are one of the most prevalent techniques to represent 3D objects [4]. The discrete nature of vertexes, edges and faces allows for intuitive manipulation and refinement of shape. This property has been leveraged by several tools that use a sculpting interface to modify meshes [2] [18] [17]. These tools allow users to push pull and stretch 3D meshes to create complex shapes. This is generally achieved using a simple mouse dragging interface. However, some authors have looked into using specialized external control mechanisms to expedite the process such as a trackball for easier navigation [17]. In general, in order to use these sculpting tools, the user specifies a point on the surface of the mesh which is the "origin" of the action. For example, a pull action might be positioned on the nose of a character's face represented using a mesh. As the user drags their mouse to move around the central action point, the action point is moved to match their mouse movements. In addition, any vertexes near the action point are moved similarly based on their distance to the center action point. A *decay function* is used to control the rate in which distance from the central action point effects the amount of change (Figure 2).

Many implementations have multiple variations of the sculpt action available through different user

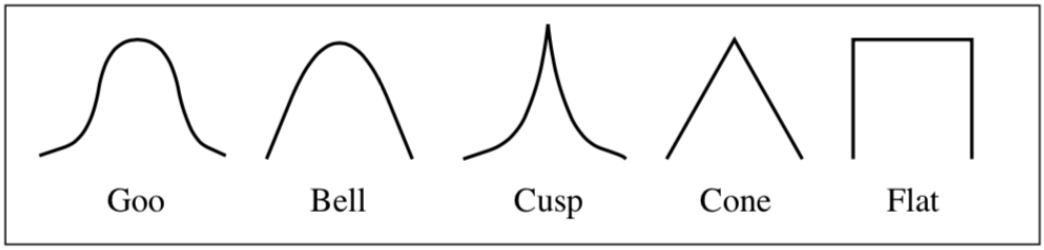

Figure 2: A set of example decay functions used. Each curve specifies how much vertexes should be modified based on their distance to a center action point.

selected tools. One of the driving reasons for this is the fact that mouse drag actions are limited to modifying shape along a plane parallel to the screen. Essentially, relative to screen space, the user may only drag and pull the surface up, down, left or right. This problem opens the door for other types of tools. One example is a tool that is analogous to adding a blob of new material to a clay sculpture. The user may click on a point on the surface of the mesh to slightly extrude vertexes within the region along the normal of the surface. These tools similarly rely on a decay function. The need for these additional tools is largely a matter of limitations of mouse and keyboard interfaces. Our interface relies on a more unified system that supports sculpting with 6 degrees of freedom without need to explicitly specify the type of sculpting action being conducted.

3.3 Virtual Reality Modeling

The challenge of intuitively representing 3D modeling interfaces in virtual reality has been explored by multiple authors [14, 5, 15, 13, 6]. We examine a few relevant works.

In [14], the author discusses a system for modeling 3D shapes using a sketching interface in virtual reality. The technique begins with 2D sketches that are seen projected in the virtual reality environment. Once the sketch appears within the head mounted display, the user may select given lines that form the sketch using hand-held controllers. Using gestures such as pushing and pulling, users may contort they sketch lines into 'rails'. These rails are 3D curves that will define the skeleton of the finished 3D model. Once a collection of 3D 'rail' curves is created, the user may use a sweeping gesture to fill in gaps between curves with a surface. The approach offers simple tools for defining 3D shape. For our purpose, the use of sketch lines is irrelevant, however. Next, we examine the work of Jerald et al. [15]. In this work, the domain of possible actions is significantly restricted. The authors here choose to use more intuitive simplified gestures such as pushing, pulling, and stretching to form shapes from geometric primitives. Additionally, shapes designed may be 'swept' in various directions to form new extruded shapes. Bruno et al. [5] takes a similar approach to virtual reality 3D modeling to [14]. In this work, the author creates a virtual reality interface

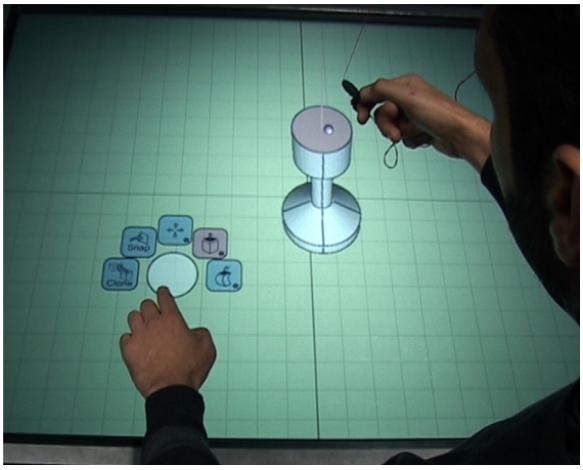

Figure 3: VR interface from [5].

for the 3D modeling program Google Sketchup. The interface follows a 2D sketch first, extrude second approach. Using VR controllers, the user may draw contours of shapes on the ground or on preexisting models. These sketches may then be pushed or pulled to extrude new shapes (Figure 3).

3.4 Sketch-Based Terrain Modeling

Several papers have been written on using sketching interfaces for terrain modeling. The following paper from Gain et al. [10] was appropriately named *Terrain Sketching*. In this work, the user may enter three distinct modes that each provide a different set of tools for sketching. The first technique is *silhouette mode*. This system largely matches the terrain sketching features implemented by Cohen et al. [8]. The endpoints of the user stroke are used to guide ray casts against the terrain surface. These two points are combined with form a vertical plane. The intersection of this plane with the surface creates the *shadow curve* (See Figure 4. This curve may then be modified from a vertical view to allow for curving mountain chains. The system also allows the user to enter *region mode*. This allows the user to have fine control over the surface texture of the terrain. From a top-down perspective, the user may define a region and a corresponding stroke that is used as a template for how rough the surface should be within the region. Finally the user is allowed to finish refinement by choosing to delete certain terrain features they built previously. One major drawback of the system is its inability to support significant modification after a terrain feature is built. In the system, silhouette curves and shadow curves may not be modified after creation, only deleted. The algorithm uses a tree data structure to store the feature curves. A negative consequence of this is that deletion of feature nodes results in deletion of their children as well. Another drawback of the approach is speed. The system is implemented on the CPU, and requires multiple seconds to generate terrain from the created

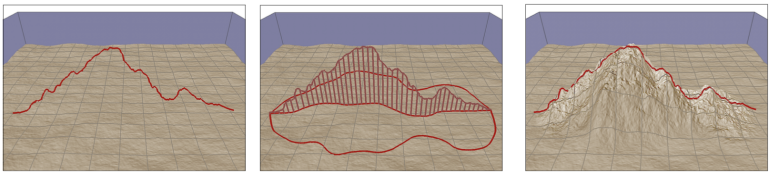

Figure 4: A sketch, shadow curve and footprint, and finally the finished terrain feature. Image from [10].

curves. Thus, interactive modifications to the finished product could not easily be achieved. The authors conclude their paper with discussion about how texture synthesis may be applied to achieve more realistic automated details. This approach, along with faster GPU based techniques [7, 3] are discussed below.

The following work by Houssam Hnaidi delved deeper into how sketch curves are represented, and leveraging the GPU [12]. In this work, users could define *feature curves* and *control curves*. *Control curves* take the place of previous hand sketching tools, but offer similar advantages. The control curves may be used to define the locations of ridges or river beds. Feature curves allow the curvature of the ridge to be changed. Each feature curve is represented with a cubic spline and allows the user to define the smoothness and angle of the ridge (Figure 5). For example, a soft rounded concave-up feature curve may be used with a control curve to define a river bed. The paper describes a method of rasterizing a terrain height field from these curves. Unlike the approach discussed by Gain et al. [10], this approach allows feature curves to intersect and overlap. In regions of intersecting curves, Laplace diffusion is used to fill in gaps that are inconclusive. Another advantage over previous work is the ability to customize curves throughout the editing process. Similar to [10], the final terrain is displaced with a noise layer to provide high frequency details. Progress is made in this work by leveraging the parallelism of the GPU. This allows for fast and efficient generation of terrain maps. The paper reports times less than half a second for large, 1024x1024 terrain maps. One downside of the approach is the amount of user control that is required to see realistic results. For very large scenes, it would take significant time to define the necessary control curves to create a convincing landscape.

A year later, in collaboration with Houssam Hnaidi, Bernhardt et al. [3] made improvements to the efficiency of the approach. This is the first approach we discuss that is capable of truly interactive editing. The approach uses a quad-tree to allow the system to generate lower resolution patches on the terrain in regions that are low in detail or are out of sight from the camera. The paper breaks down the two main user interactions with the system as drawing strokes, and moving the camera. Both interactions force updates to occur within the quad-tree. In the approach, the CPU is used to manage the quad-tree that stores low

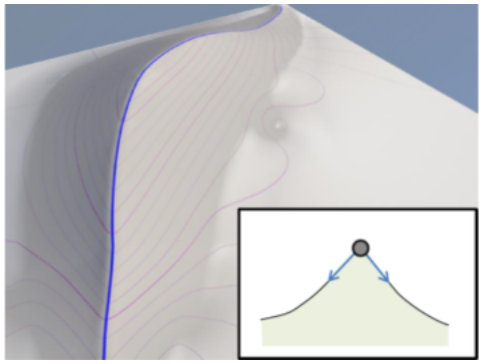

Figure 5: A ridge line and the corresponding feature curve (lower right). Image from [10].

resolution details. Then, the GPU is employed to iteratively simplify the information supplied via the quad-tree and run diffusion steps (similar to [12]). Then, the GPU is employed again to iteratively tessellate the terrain mesh until the desired detail is reached. The control curve user interface heavily resembles the work of Hnaidi et al. [12]. The approach boasts large increases in efficiency using their coupled CPU-GPU approach. For terrain maps as large as 2048 x 2048, they record total terrain generation times as low as 40 ms. These speeds allow for efficient, real-time modification and represents a significant step forward in the field.

The work on sketch based modeling approaches was continued by Tasse et al. [21]. The major difference in this approach is that it is meant for modifying existing terrains as apposed to creating new terrains from scratch. The focus of the system is thus preserving the structure of original features while still allowing the user to intuitively and quickly make modifications through sketching. The user interaction all takes place from a first person perspective. This choice allows modification to be quickly sketched from the standpoint of a player in the virtual world. The approach begins by extracting ridge and silhouette lines from the terrain from a given perspective. This is achieved by finding all edges on the terrain mesh that are adjacent to a polygon that is facing the camera and a polygon that is facing away from the camera. Then once all terrain ridges are identified, similarly to previous work, the user may sketch a number of silhouette lines. In this work, however, sketches are not expected to be limited to single mountain crest or valley. The user may draw many mountain ridge silhouette lines to create layers of mountain range scenery (Figure 6). Next, the sketch is examined using a sweep-line algorithm to determine the relative ordering of each mountain silhouette based on distance to the camera. Once the set of terrain ridge and silhouette lines is known, and the set of desired silhouette lines is extracted from the sketch, the two sets of lines can be paired. The pairing process attempts to minimize the amount of modification required to the terrain while preserving the goal that the final terrain will match the silhouette of the sketch. With this in mind, the authors describe an algorithm that pairs together the individual sketch ridge lines with the pre-existing set of ridge lines

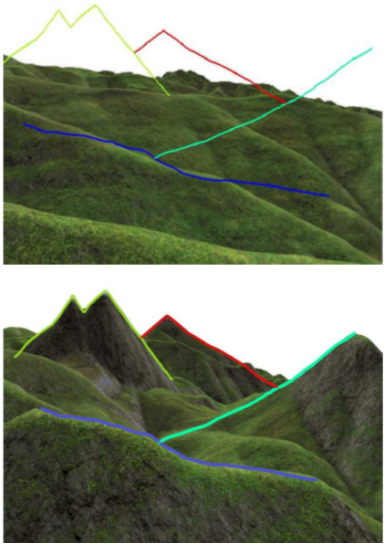

Figure 6: An example sketch and resulting terrain modification. Image from [21].

in the terrain. Once the set of lines are matched, some post processing is applied to the sketches to insure that they extend fully to the ground along the same trajectory that they were drawn with. Next, for each sketch line-terrain ridge pair, the offset is calculated and a displacement map is generated and smoothed using a diffusion step. The final result of a sketch thus preserves the natural structure of the original terrain while closely matching the silhouette of the user's sketch. This approach makes a significant step towards allowing procedurally generated terrains to be modified in an intuitive and simple manor. Unlike previous work in sketch based terrain editing, the approach attempts to preserve geologic features. Thus, geologically correct inputs will result in mostly geologically correct outputs.

3.5 Example Guided Tools

We take a slight detour to examine an example based technique [20]. These techniques rely on simple sketching to specify locations of basic terrain primitives such as mountains and valleys. In this system, the user supplied an input sketch as a guide for the algorithm. Then, example terrain data is used to synthesize a new set of data that matches the input guide. The input sketch contains a series of lines that will form the ridges and valleys in the finished product (Figure 7). However, the user is not given control of the actual elevation of these features. In other words, they may control the location of mountains, but not the actual height and structure of the ridges. The first major step in the algorithm is to extract ridge and valley information from the example data. This procedure involves finding an initial list of candidate points that

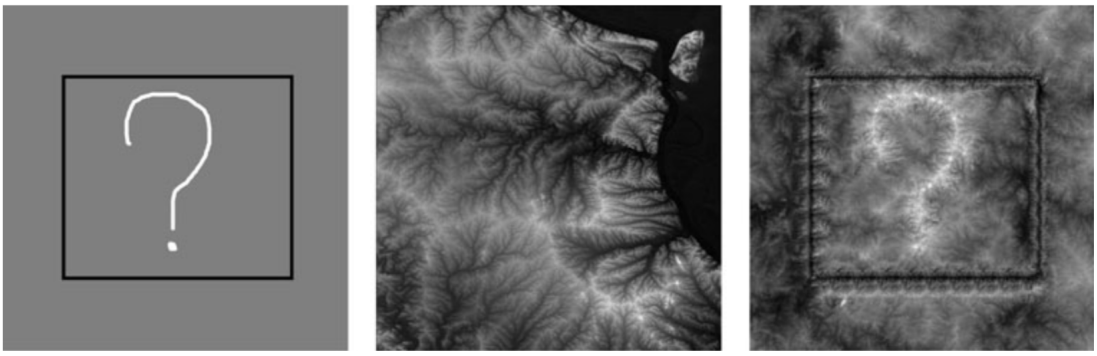

Figure 7: An example input sketch (left), the example data (middle), and resulting terrain (right) respectively. Image from [20].

may be along ridges or valleys. These points are then connected and an iterative process is used to throw away excess connections until a simple tree graph remains that traces the features of ridges and valleys. This graph is then traversed and important terrain features such as branch points and endpoints of ridges are extracted. Then, a similar process is repeated on the input sketch such that locations of branching, etc are located. What remains is to match up the example patches with the guide patches to synthesize a new terrain. The synthesized terrain then possesses the overall structure of the guide while preserving the detail of the example data. To achieve this, the example data is warped to better fit the input guide. Then using a series of seam removal techniques, a polished output is generated. This approach offers quick and easy sketching of large scale features, yet it lacks the ability to customize small scale regions. While the approach generates convincing geologic features considerably faster than running physically accurate erosion simulations, it still lacks the performance for real-time modification

4 Methods

In this project, we explored the use of virtual reality as an interface for quick and efficient terrain modeling. Users of our system use hand gestures to interact with a 3D terrain model displayed inside a VR headset. The terrain model is then output into a standard height field format (Figure 8).

4.1 General

Recall that a consequence of height fields is the limitation placed on the types of terrains that can be represented. In particular, terrains with caves or overhangs cannot be represented using height fields. As a consequence, our system forces terrains made by its users to be represented by standard height fields

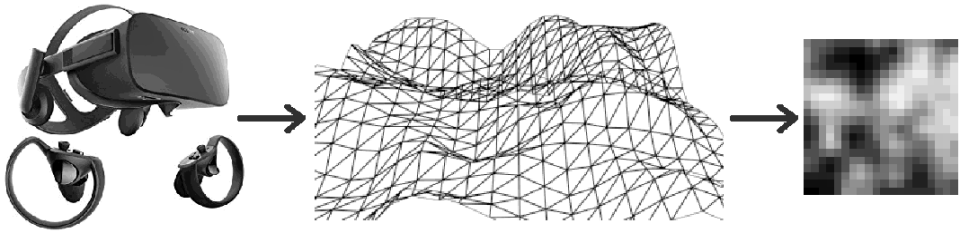

Figure 8: AN overview of the pipeline. A virtual reality user interface is used to model and sculpt the terrain surface mesh. Based on the mesh, a finished height field can be exported

throughout the editing process by filling in overhangs and caves.

The Oculus Rift VR headset was used for this project. The Rift is a widely available consumer VR headset. The Rift is equipped with a set of hand held controllers that are motion tracked by the system. Since the ability to track our users hand movements is a core aspect of this project, the Rift is a convenient choice.

Unity3D was used to develop the interfaces explored in this project. Unity3D comes with drivers and packages that allow for easy integration with the Oculus Rift. In addition, Unity3D possesses a well featured terrain engine that is relied on in this project for rendering terrains. In general, Unity3D is an intuitive tool for prototyping projects making it a useful component in our project.

All interactions made with terrains in our system are through simple gestures. A gesture is formed by pressing the primary trigger on a VR hand held controller, moving the controller, and finally releasing the trigger. This gesture can be equated with the real life action of grabbing, moving, and releasing an object. A gesture is defined only by the positions of the controllers when the trigger is pressed and the position when the trigger is released. The position of the controller between the point of time when the gesture begins and ends are not considered. This is primarily a consequence of performance limitations. Calculating terrain modifications throughout a gesture would cause significant slowdown given our current implementation. Additionally, we believe that gestures defined by only a start position and end position are conducive to simple and easily replicated interactions. The area of effect of a gesture is visualized to the user in the form of a spherical transparent overlay positioned at the location of the user's hand. Users may control the size of the area of effect by pressing forward or backward on a thumb stick mounted on the controller.

4.2 Interaction Schemes

We develop two interaction schemes to explore the effectiveness of VR for terrain generation. Throughout development, we focused on designing intuitive interactions that seem similar to interaction with real world objects.

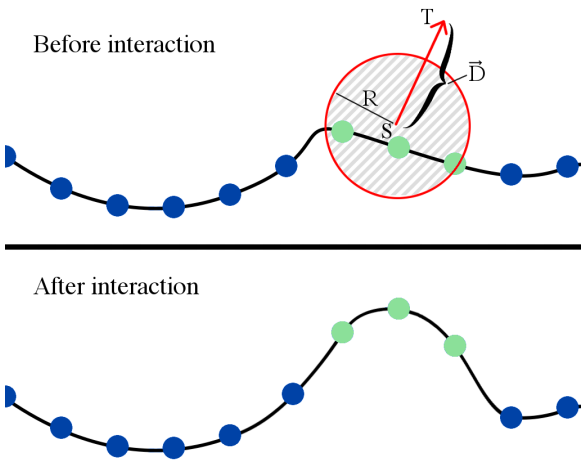

Figure 9: A 2D cross section representing the result of a user gesture in the mesh interaction scheme. Dark (blue) circles represent unmodified control points on the terrain surface. Light (green) circles represent control points effected by the interaction. The striped region represents the area of effect of the interaction, a sphere of radius R. The arrow represents the interaction vector: a vector from the start of the gesture, S, to the end of the gesture, T.

4.2.1 Mesh

The first interaction scheme is similar to the mesh sculpting interface discussed in [2]. In this scheme, interaction gestures stretch the surface of the terrain in the direction of the gesture. The area modified by the interaction is controlled by a grid of control points positioned on the surface of the terrain. Control points are modified only if they are within the area of effect specified by the user at the start of the interaction. Figure 9 shows a 2D cross section of an interaction in this scheme. In the figure, dark (blue) circles represent unmodified points and light (green) circles represent modified control points. The striped region represents the area of effect of the interaction.

A falloff curve is used to control the magnitude of control point displacement. Control points outside of the area of effect are not modified by this interaction. Points inside the area of effect are displaced using a combination of a sigmoidal weighting curve and the interaction vector. The choice of sigmoid weighting curves is motivated by the "goo" curve discussed in [4] (Figure 2). Let $\sigma(t)$ be a sigmoid curve. $\sigma(t) = 6t^5 - 15t^4 + 10t^3$. Let $\vec{D} = T - S$ be the interaction vector, S be the start point of the gesture, T be the end point of the gesture, R be the radius of the area of effect, and p_i be the position of a control point within the area of effect. After an interaction, each control point p_i within the area of effect is displaced such that $p_i = p_i + \vec{D}_i$ where. $\vec{D}_i = \sigma\left(1 - \frac{distance(S,p_i)}{R}\right) \cdot \vec{D}$.In order to ensure that consecutive interactions are consistent, the control point mesh is reset after every interaction. To achieve this, a height field is generated after each interaction. The elevation stored in each cell of the height field is highest point on the mesh

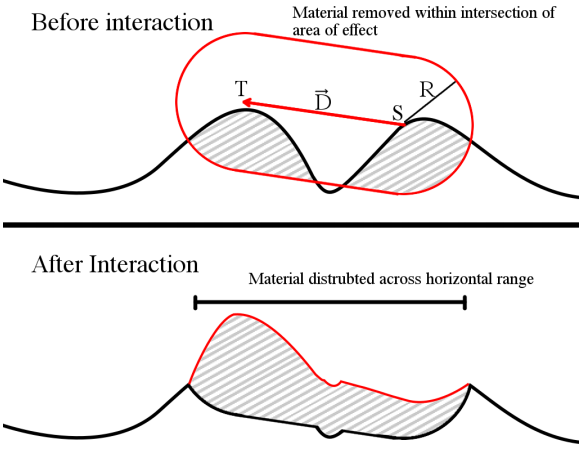

Figure 10: The result of user gesture in the sand interaction scheme. Striped region denotes redistributed material. Capsule in top panel represents the area of effect of the interaction. The arrow represents the interaction vector, $\vec{D}$: a vector from the start of the gesture, S to the end of the gesture, T.

that intersects a vertical line cast from the position of the height field cell center. This process removes any overhangs that resulted from an interaction. Once the height field is generated, the control point mesh is reset to a uniform grid aligned with the height field cell centers and the vertical position of each control point is set to the elevation stored in each corresponding height field cell.

4.2.2 Sand

The second interaction scheme is analogous to a sandbox. As opposed to distorting the terrain surface, interactions in this scheme emulate the process pushing sand around in a sandbox. Figure 10 shows a 2D cross section of an interaction in this scheme. The capsule region shown in the top panel represents the area effected by the interaction. The striped region represents the intersection of the volume of the capsule and the terrain. The material within this intersection is redistributed in the the direction of $\vec{D}$, the vector of interaction.

Interactions in this scheme directly modify the underlying height field. After a gesture is completed, a capsule is formed by tracing the area of effect sphere from the interaction start, S to the interaction end, T. In 3D, this is a cylinder with hemispheres at the end caps. All height field cells such that the position of the center of the cell is within the capsule are considered for modification. Let h_i be the height of cell i. Additionally, let P_i be the xy position of cell i. We label this set of cells such that $0 \leq i \leq n$. Let b_i and t_i be the height of the bottom and top of the capsule at P_i. Let r_i equal the length of the height field column segment which intersects the capsule, $r_i = min(max(h_i - a_b, 0), t_i - b_i)$. The height of the intersection is removed from cell h_i. For all cells within the area of effect ellipsoid: $h_i = h_i - r_i$. The removed material:

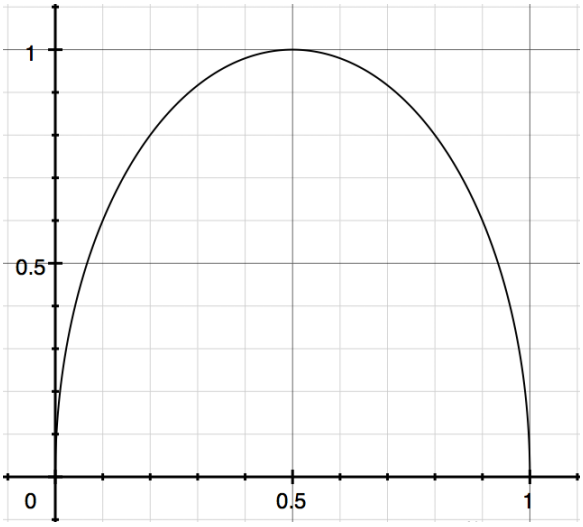

Figure 11: Smoothing function used to even out distribution curve.

m_{total} is summed among all n cells considered: $m_{total} = \sum_{i=0}^{n} r_i$.

Once all material within the intersection of the terrain and the capsule is removed, the quantity of material removed from each cell is used to generated a distribution curve for the later deposition of the material. The curve is used to control where the removed material is deposited along the xy projection of the capsule. The distribution curve is defined by a discrete set of samples. Let L be the number of samples in the curve. $L = \lceil distance(S_{xy}, T_{xy}) \rceil$. Every height field cell maps to a sample index, j, in the distribution curve. Let $\vec{O_i} = P_i - S_{xy}$. First, the component of $\vec{O_i}$ along $\vec{D}$ is found: $comp_{\vec{D}}\vec{O_i} = \frac{\vec{D} \cdot \vec{O_i}}{|\vec{D}|}$. The index, j mapped to from P_i is: $j = min(max(\left\lfloor \frac{comp_{\vec{D}}\vec{O_i} + R}{|\vec{D}| + 2R} \right\rfloor, 0), L)$

Once the distribution curve sample index has been identified at cell P_i. the quantity of removed material, r_i, at the cell is distributed equally across distribution slices d_{j+1} through d_L. The mapping of cell positions to a distribution curve index is based on how close the cell is to T. Cells closer to the end of the interaction movement are mapped relatively closer the end of the curve. By distributing the material within a cell across all curve samples with a higher index, the distribution will ensure that material is only moved in the direction of the interaction vector, $\vec{D}$.

Once this step has been completed for all height field cells, the distribution curve is smoothed to avoid too much weight at the very end of the curve. To accomplish this, we multiply our distribution curve at each sample, d_j by a smoothing function $s(t)$. $d_j = d_j \cdot s(\frac{j}{L})$. We choose $s(t)$ to be a semi circle scaled $2x$ on the y-axis as seen in Figure 11.

In addition to the distribution curve, a weighting is applied to match the way in which material is removed. This final weighting is based on the original capsule. Let c_i be the height of the capsule at P_i. Finally, let w_i be the final weight calculated for the i^{th} cell. $w_i = d_j \cdot c_i$ (where d_j is the value stored in

the j^{th} sample of the distribution curve. j is calculated above). The combined weight of all n cells is then calculated. $w_{total} = \sum_{i=0}^{n} w_i$. Finally, for cell, the final height is assigned: $h_i = h_i + m_{total} \cdot \frac{w_i}{w_{total}}$. In the first step, material is removed from the height field and summed into m_{total}. Since the sum of all $\frac{w_i}{w_{total}}$ is equal to 1, the entirety of m_{total} will be returned to the height field. Thus, the sum total of all elevation stored in the height field is preserved during the interaction.

In addition to redistributing (pushing) material, user's have the option to drag and drop material. If the user ends a gesture at a position above the terrain surface, a drag and drop is completed instead of a push interaction. Drag and drop interaction function similarly. Material is removed within the intersection of the terrain and the area of effect sphere. However, instead of spreading the removed material out, the removed material is simply added to the height field in a circular region at the xy position of the controller when the gesture was completed. A similar weight function is used to control the deposition distribution. For drag and drop interactions, the weight at cell i is $w_i = distance(P_i, T)$. Similarly, all weights are then summed and the final height assignment is $h_i = h_i + m_{total} \cdot \frac{w_i}{w_{total}}$.

4.2.3 Erosion

In the sand interaction scheme, we simulate an ongoing erosion process as users interact with the terrain. The erosion function is only used in the sand model since it fits the analogy of a sandbox. In contrast, the mesh model does not appear to intuitively share this behavior. This erosion process keeps overly steep cliffs from being formed as well as smoothing the terrain surface. The erosion process is designed after the hill slope stability model discussed in [16]. Hill slopes above a certain threshold angle are collapsed and material is moved downhill. For each erosion iteration, the elevation at each cell, c_i, is compared to its four non-diagonal neighboring cells to determine the lowest neighbor, c_l. The horizontal distance, d, between cell c_i and c_l is calculated based on the size of the terrain and the resolution of the height field. The angle between the cells is then calculated using

$$\theta = \tan^{-1}(\frac{c_i - c_l}{d})$$

If θ is greater than 55, then $\frac{c_i - c_l}{4}$ is added to c_l and subtracted from c_i. This calculation simulates the process of unstable material moving downslope. The calculations made in this algorithm are completed in parallel such that during an erosion iteration, material transfered from c_i to c_l does not effect the erosion calculation made in cell c_l. Material is only swapped between the lowest neighbor of c_i to simplify calculations. The algorithm described is a standard method of simulating simple hill slope erosion.

4.3 Evaluation

We evaluate the success of our project based on multiple metrics. We assessed the intuitive qualities of the interface during development via subjective analysis. Additionally, during development, we received informal feedback from fellow researchers on the intuitiveness and success of our tool. Finally, we created a user study to compare the effectiveness of the two interaction schemes.

4.3.1 User Study

We ran a user study (N = 25) to assess which interaction scheme is more intuitive for users. Study participants were chosen based on interest out of a population of students at Union College. Each study participant tried one of the two interaction schemes in an alternating order. An individual study session begins with a description of the research problem and motivation for my proposed solution. Users are then guided through a tutorial of the interaction scheme they are assigned. In this tutorial, I walk users through the basic interactions available within the scheme as well as other baseline features such as how to modify the size of the area of effect. Throughout the demo, if users are not utilizing a feature that seemed relevant to me, I would remind the user of this capability. This step was done to ensure users were aware of how to use all aspects of the tool.

Once the user appeared to be acquainted with the tool, they are given a test in order to determine their ability to use each their assigned interaction scheme to build a specified terrain. Users begin with a flat, unmodified terrain plane. Superimposed above this plane, users were shown a ghost terrain representing the goal state of their initial flat landscape. The ghost terrain is represented as a transparent overlay as seen in Figure 12.

Users are given 10 minutes to transform the unmodified terrain plane into the goal terrain. The goal terrain is pulled from real terrain elevation data. Additionally, the volume of goal terrain is modified such that it is equivalent to the volume of material given in the initial terrain plane (since the sand scheme conserves volume).

Throughout the editing process in the study, the goal terrain is compared to the users attempts at recreating the goal terrain. The comparison calculates the total difference of each height field cell of the user created terrain to the respective cell of the goal terrain. We use the sum of these differences as a score indicating the success of the interface. By making comparisons of the user's terrain throughout the editing process, we can assess factors such as the final score and the general rate in which the work-in-process terrain converges towards the goal terrain.

In addition to collecting quantitative data on the progress of our users, we created a survey that users

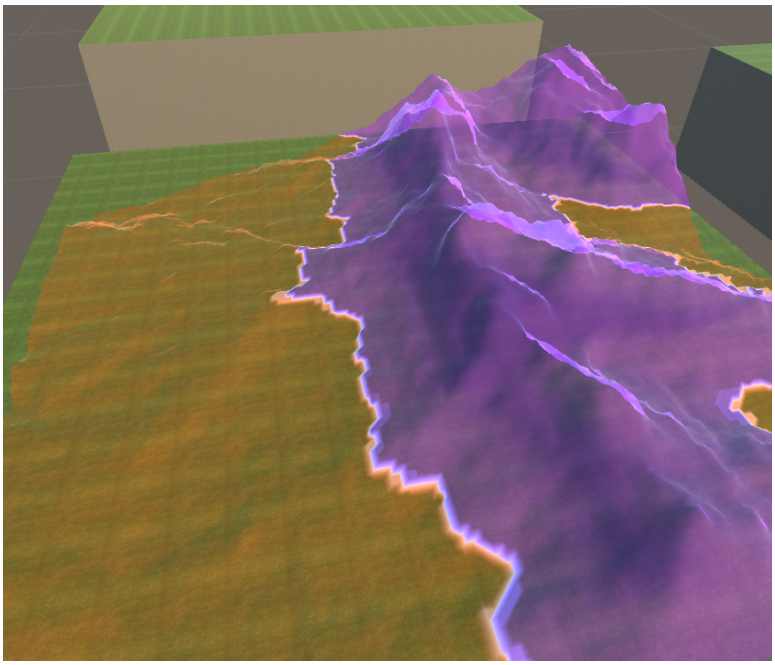

Figure 12: Goal "ghost" terrain represented to user study participants. The ghost landscape is rendered in two colors. The goal terrain regions currently at a higher elevation than the user's terrain are colored purple and the regions lower are colored orange.

filled out following their experience. The survey asks users to rank how intuitive the system felt, how physically uncomfortable the system was, and short answer elaborations on the above questions.

With data from user performance using both the mesh interaction scheme and the sand interaction scheme, as well as qualitative user responses, we were be able to gain insight regarding which interaction scheme is more effective.

We design our two interfaces to promote fair comparison. To achieve this, we ensured that many of the variables impacting the use of the two systems were equivalent. For example, the resolution of the control point mesh used in the mesh scheme is equivalent to the resolution of the height field we use in the sandbox scheme. Additionally, both schemes have identical user interfaces and identical visualizations. We ensure that both schemes have zero delay and lag during interaction.

5 Results

Results of our study show that the sand scheme allows users to achieve higher scores in the test. Figure 13 shows the mean score over time of all participants using the sand model and mesh model. The x-axis represents the time progressed in the 10 minute session. The y-axis is inverted to show the scores increasing over time (low offset between user terrain and goal terrain mean higher scores). The median lines of the

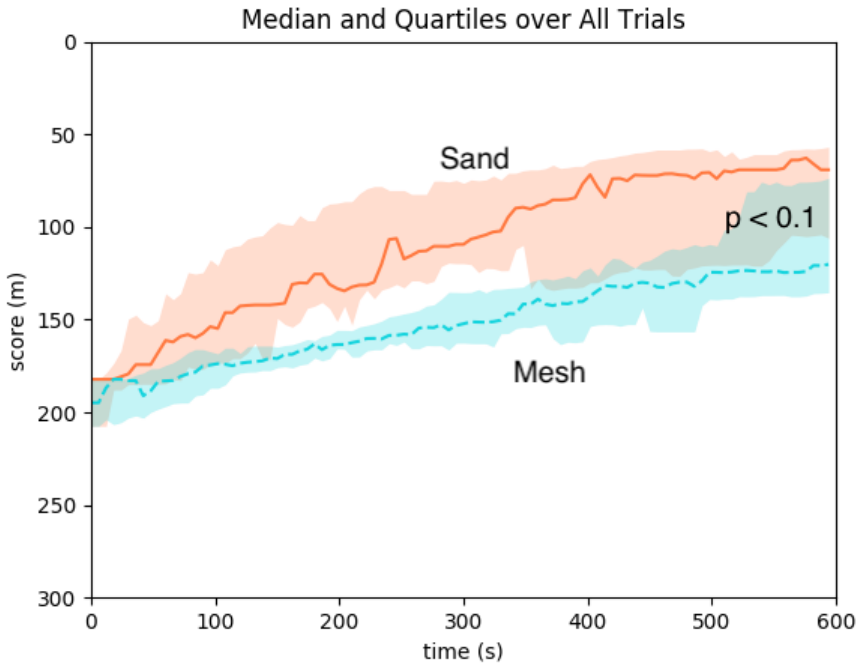

Figure 13: Results of user study. Solid and dashed lines represent the median of all users in the sand and mesh groups respectively. Shaded regions represent the 25th and 75th percentile ranges.

mesh model (dashed line) is significantly smoother than the median for the sand model. This suggests that interactions may be more consistent in the mesh model. Additionally, the presence of the erosion model in the sand model may play a role in this effect. Despite the less consistent score gain in the sand model, overall, the median sand model score climbs faster and reaches a higher final score. The scores in both groups were compared at the 10 minute mark and shown to have an insignificant difference with $p > 0.05$ ($p = 0.083$).

Interestingly, although there was a trend towards higher scores in the sand model, the best two scores in the test were achieved in the mesh model. Based only on qualitative observation and discussion with these two participants, it was evident that they had the most experience using VR hardware. This suggests that experienced VR users may be able to use the mesh model more successfully than the sand model.

Data from two participants were not included in the final analysis. The first participant experienced a major bug in the system which rendered their data useless. The second participant was mistakenly allowed to take the test despite having had prior experience with the system.

Users are asked to rank how natural or intuitive their experience was. The results of this component of the followup survey match the trend in user scores. Figure 14 shows a box plot of user responses provided on a 5 point scale. Median scores are shown as black lines. Boxes represent 25th and 75th percentile ranges.

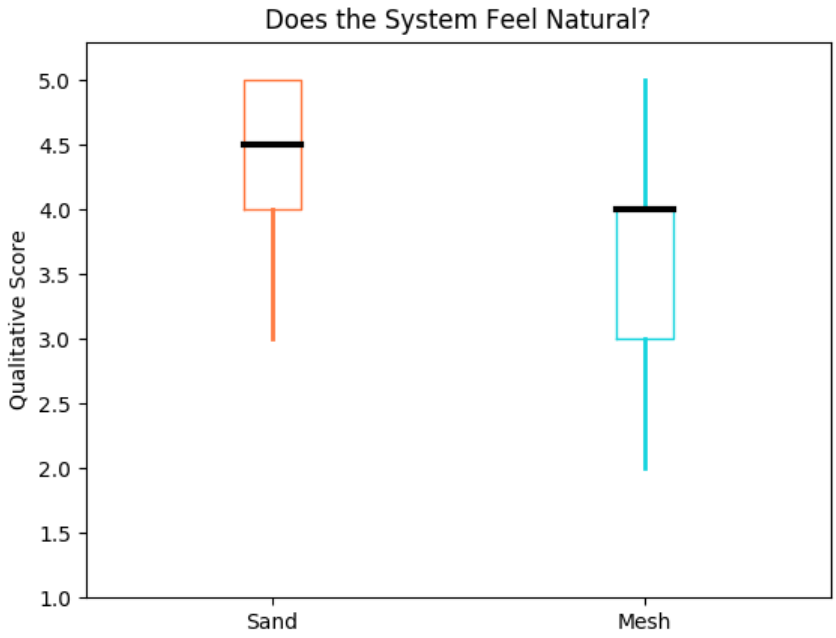

Figure 14: Results of followup survey. Users were asked to rank how natural and intuitive their experience with the system was. Black lines represent the median and boxes represent the 25th and 75th percentile ranges.

Median scores are higher for the sand model participants.

Users are asked to justify their ranking on the question above. A variety of responses were submitted to this question. A common trend was users indicating that at first they the system did not feel intuitive however quickly it began to feel natural. This result indicates that our system may be relatively easy for beginners to learn.

6 Conclusions

The goal of the project was to determine which interaction scheme was more intuitive for users with a range of backgrounds. Throughout the development of our system, we attempted to ensure that our system was friendly to non-technically experience users. This meant avoiding reliance on complicated GUI and focusing on intuitive gestures. The results of our user study indicate that users have an easier time using the sand model than the mesh model.

Further work may be done to advance our system. We believe that multi-handed gestures could be leveraged for more advanced interactions. For example, during testing, an interface was considered where users could use two hands to stretch sections of terrain into flat regions. Another area of improvement

would be more responsive interactions. Currently, interactions made with the system are not processed until a gesture is completed (in other words, until the trigger is released). The ability for users to see the impact of a gesture before the completion of the gesture would aid in fine tuned control.

One factor that may have largely influenced the results of our study was the presence of the erosion model. The model essentially amounts to a smoothing filter on the terrain. This may have played a significant role in participants ability to achieve smooth and convincing results in the sand model. In order to validate that the difference in the two interaction scheme came from how users interacted as opposed to the erosion model, farther work could be done to isolate this parameter.

In the author's perspective, the task of modeling terrains has many inherent properties that facilitate natural gestural interactions. For example, it is easy to imagine shaping a landscape within a sandbox or with a pile of clay. Consequently, we believe that virtual reality may be effectively leveraged for this task.

References

[1] Tomas Akenine-Moller, Eric Haines, and Naty Hoffman. *Real-Time Rendering*. AK Peters/CRC Press, 2018.

[2] Jeff B Allan, Brian Wyvill, and Ian H Witten. "A methodology for direct manipulation of polygon meshes". In: *New Advances in Computer Graphics*. Springer, 1989, pp. 451–469.

[3] Adrien Bernhardt et al. "Real-time terrain modeling using CPU-GPU coupled computation". In: *Graphics, Patterns and Images (Sibgrapi), 2011 24th SIBGRAPI Conference on*. IEEE. 2011, pp. 64–71.

[4] James R Bill and Suresh K Lodha. "Computer sculpting of polygonal models using virtual tools". MA thesis. University of California, Santa Cruz, 1994.

[5] R Bruno et al. "Mockup Builder: 3D modeling on and above the surface". In: *Computers & Graphics* 37 (2013), pp. 165–178.

[6] Jeff Butterworth et al. "3DM: A three dimensional modeler using a head-mounted display". In: *Proceedings of the 1992 symposium on Interactive 3D graphics*. ACM. 1992, pp. 135–138.

[7] Giliam JP de Carpentier and Rafael Bidarra. "Interactive GPU-based procedural heightfield brushes". In: *Proceedings of the 4th International Conference on Foundations of Digital Games*. ACM. 2009, pp. 55–62.

[8] Jonathan M Cohen, John F Hughes, and Robert C Zeleznik. "Harold: A world made of drawings". In: *Proceedings of the 1st International Symposium on Non-Photorealistic Animation and Rendering*. ACM. 2000, pp. 83–90.

[9] David S Ebert. *Texturing & Modeling: A Procedural Approach*. Morgan Kaufmann, 2003.

[10] James Gain, Patrick Marais, and Wolfgang Straßer. "Terrain sketching". In: *Proceedings of the 2009 symposium on Interactive 3D graphics and games*. ACM. 2009, pp. 31–38.

[11] Jean-David Génevaux et al. "Terrain generation using procedural models based on hydrology". In: *ACM Transactions on Graphics (TOG)* 32.4 (2013), p. 143.

[12] Houssam Hnaidi et al. "Feature based terrain generation using diffusion equation". In: *Computer Graphics Forum*. Vol. 29. 7. Wiley Online Library. 2010, pp. 2179–2186.

[13] Cathleen E Hughes et al. "Cavecad: Architectural design in the cave". In: *3D User Interfaces (3DUI), 2013 IEEE Symposium on*. IEEE. 2013, pp. 193–194.

[14] Bret Jackson and Daniel F Keefe. "Lift-off: Using reference imagery and freehand sketching to create 3d models in vr". In: *IEEE transactions on visualization and computer graphics* 22.4 (2016), pp. 1442–1451.

[15] Jason Jerald et al. “Makevr: A 3d world-building interface”. In: *3D User Interfaces (3DUI), 2013 IEEE Symposium on*. IEEE. 2013, pp. 197–198.

[16] Isaac J Larsen and David R Montgomery. “Landslide erosion coupled to tectonics and river incision”. In: *Nature Geoscience* 5.7 (2012), p. 468.

[17] André LeBlanc et al. “Sculpting with the’ball and mouse’metaphor”. In: *Proc. Graphics Interface*. Vol. 91. 1991, pp. 152–159.

[18] Richard E Parent. “A system for sculpting 3-D data”. In: *ACM SIGGRAPH Computer Graphics*. Vol. 11. 2. ACM. 1977, pp. 138–147.

[19] Ruben Smelik et al. “Integrating procedural generation and manual editing of virtual worlds”. In: *Proceedings of the 2010 Workshop on Procedural Content Generation in Games*. ACM. 2010, p. 2.

[20] Flora Ponjou Tasse, J Gain, and Patrick Marais. “Enhanced Texture-Based Terrain Synthesis on Graphics Hardware”. In: *Computer Graphics Forum*. Vol. 31. 6. Wiley Online Library. 2012, pp. 1959–1972.

[21] Flora Ponjou Tasse et al. “First person sketch-based terrain editing”. In: *Proceedings of Graphics Interface 2014*. Canadian Information Processing Society. 2014, pp. 217–224.