Generative Terrain Authoring with Mid-air Hand Sketching in Virtual Reality

Yushen Hu
New York University
New York, NY, USA
yh3856@nyu.edu

Keru Wang
New York University
New York, NY, USA
keru.wang@nyu.edu

Yuli Shao
New York University
New York, NY, USA
ys3203@nyu.edu

Jan Plass
New York University
New York, NY, USA
jan.plass@nyu.edu

Zhu Wang*
New York University
New York, NY, USA
zhu.wang@nyu.edu

Ken Perlin*
New York University
New York, NY, USA
perlin@cs.nyu.edu

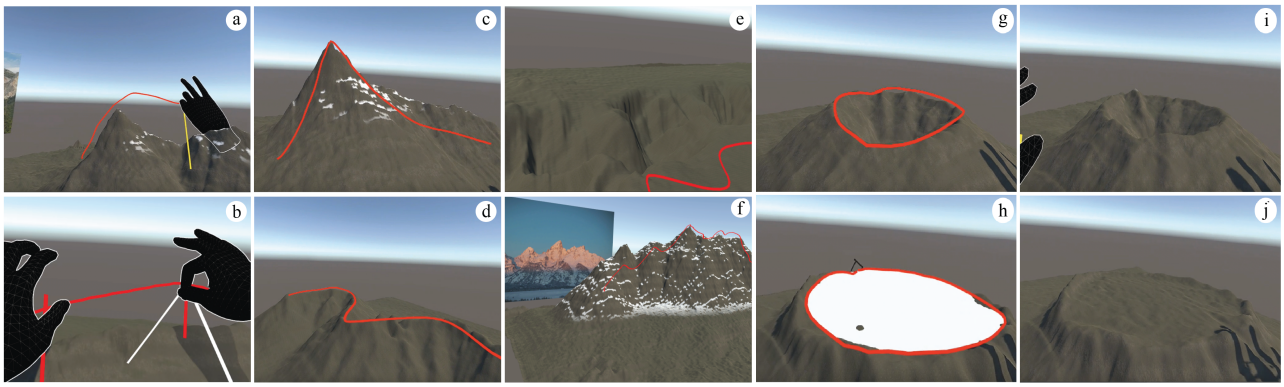

Figure 1: Generative Terrain Authoring with Mid-air Hand Sketching in Virtual Reality: a. Users can sketch terrain outlines using pinch gesture. b. Users can modify the sketch. c-e. The system can generate different types of terrain: summits, ridges, canyons, and mesa. f. User study participants can generate high-quality results with our system in a short time. g-j. Different drawing modes: g. Draw a volcano with a non-filled drawing. i. The saved result of the volcano in g. h. Draw a mesa with filled-polygon mode. j. The saved results of the mesa in h.

ABSTRACT

Terrain generation and authoring in Virtual Reality (VR) offers unique benefits, including 360-degree views, improved spatial perception, immersive and intuitive design experience and natural input modalities. Yet even in VR it can be challenging to integrate natural input modalities, preserve artistic controls and lower the effort of landscape prototyping. To tackle these challenges, we present our VR-based terrain generation and authoring system, which utilizes hand tracking and a generative model to allow users to quickly prototype natural landscapes, such as mountains, mesas, canyons and volcanoes. Via positional hand tracking and hand gesture detection, users can use their hands to draw mid-air strokes to indicate desired shapes for the landscapes. A Conditional Generative Adversarial Network trained by using real-world terrains and their height maps then helps to generate a realistic landscape which combines features of training data and the mid-air strokes. In addition, users can use their hands to further manipulate their mid-air strokes to edit the landscapes. In this paper, we explore this design space and present various scenarios of terrain generation. Additionally, we evaluate our system across a diverse user base that varies in VR experience and professional background. The study results indicate that our system is feasible, user-friendly and capable of fast prototyping.

CCS CONCEPTS

• **Human-centered computing** → **Virtual reality**; • **Computing methodologies** → **Parametric curve and surface models**; **Machine learning approaches**.

KEYWORDS

Virtual Reality, Hand Gesture Control, Hand Sketching, Generative Terrain Authoring

*Equal advising

VRST '24, October 09–11, 2024, Trier, Germany

ACM ISBN 979-8-4007-0535-9/24/10
https://doi.org/10.1145/3641825.3687736

ACM Reference Format:
Yushen Hu, Keru Wang, Yuli Shao, Jan Plass, Zhu Wang, and Ken Perlin. 2024. Generative Terrain Authoring with Mid-air Hand Sketching in Virtual Reality. In *30th ACM Symposium on Virtual Reality Software and Technology (VRST '24), October 09–11, 2024, Trier, Germany*. ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3641825.3687736

1 INTRODUCTION

Terrain authoring, including terrain generation and editing, is the interactive process of shaping and customizing virtual landscapes in a digital space. Terrain generation involves the creation of virtual landscapes by simulating geographic features such as mountains, canyons, volcanoes and mesas. Modern Virtual Reality (VR) platforms, which combine advanced technologies such as hand-tracking and speech input alongside stereo displays, provide unique benefits for terrain authoring, particularly in the context of design, visualization, and interaction. It has been decades since researchers started exploring VR-based spatial interfaces for digital content manipulation and CAD (computer-aided design) modeling [4, 29, 41]. For example, they have used gesture interfaces (mostly achieved by glove-based hand-tracking systems) to replace traditional input (keyboard, mouse, tablet, etc.) [4, 29, 41]. VR platforms inherently empower users with the ability to use spatial and natural interfaces, such as hand tracking, controller tracking, speech input, etc. In addition to new input modalities, VR can benefit perception when authoring terrains. Users can observe models from desired points of view by simply moving their head. This dynamic perspective shift enhances spatial awareness by improving users' perception of scale and spatial relationships between segments. As a result, VR can potentially transform the design process. Users can immersively create, edit, navigate and experience virtual landscapes as if those landscapes were physically present, which can help to maintain a comprehensive understanding of the landscapes on the fly. Additionally, designing and building terrains in VR provides a unique benefit if the terrains are intended for a VR experience: Designers can immerse themselves into their target user's perspective, facilitating a more user-centered design approach.

Yet fast prototyping, terrain fidelity and user-friendly interfaces continue to pose challenges. Landscapes often have diverse features, making it inherently complex to generate realistic terrains while achieving an intended aesthetic style. To mitigate these issues, we present a VR-based terrain authoring system which incorporates CGAN as a generative terrain synthesizer, allowing users to immersively, rapidly and easily prototype terrains using intuitive interactive hand controls.

Our system integrates the VR platform's build-in hand tracking and gesture recognition functions as its primary input modality. The hand tracking sub-system uses pinching as the action trigger and then tracks the pinch point for mid-air stroke drawing in VR. Similar to prior work [4, 29], we integrate pinch to facilitate direct freehand manipulations with rich controls such as pinch-to-select and two-handed scaling. Our exploration extends to the design space of the hand interface, which provides a comprehensive demonstration of all supported interactions across various terrain generation scenarios. A mid-air stroke, representing the 3D contours of the terrain, captures both horizontal layout and height variations. This helps the system to efficiently generate terrains that match the user-specified height and shape, without the need for extensive adjustments to elevation required by most other terrain generation tools [10, 11]. The generative model for terrain generation is a CGAN module trained on pairs of real-world terrains and their corresponding height maps. The module synthesizes height information extracted from the user's mid-air strokes to generate a realistic landscape that reflects the user's creative intent and aligns with the patterns and features learned from the training dataset. Notice that due to the technical immaturity of Machine Learning-powered 3D voxel content generation, despite utilizing 3D input in VR space, our system still generates a 2D result, which is then interpreted as the height data to be used for the generation of the 3D terrain. In this way we can generate any single-valued terrain, but not multiple-valued terrains such as caves.

In summary, our work makes the following contributions:

(1) A novel contour sketching interface that leverages VR platforms' built-in hand-tracking and gesture recognition to draw the mid-air 3D contours of desired terrains and control editing, enabling the terrain synthesizer to estimate and generate a target terrain that matches the sketched 3D outline.
(2) An algorithm embedded in the terrain synthesizer that dynamically adapts the user-desired terrain 3D outlines and characteristics to the deformation process of the terrain, which overcomes the problems encountered in previous approaches.
(3) Designing and implementing the terrain authoring pipeline that integrates the above user interfaces into CGAN terrain synthesizers. Synthetic noise is added in post-processing to achieve more realistic results. In addition, we conduct preliminary studies to evaluate the effects of added noise and demonstrate the proof of concept.

2 RELATED WORK

As outlined in the survey by Galin et al. [9], Terrain generation techniques can be broadly classified into three technical approaches: procedural modeling, geomorphological simulation, and example-based generation. Procedural approaches incorporate algorithms, such as noise functions [26, 27], faulting [18, 36], and subdivision [6, 20], to mimic natural landscapes. Geomorphological simulation emulates the physical processes that contribute to the formation and transformation of geographical features [21, 28, 30]. Example-based generation synthesizes landscapes by relying on scanned real-world terrain data sets and often leveraging data-driven methods. According to this classification, our work employs an example-based method with a real-world terrain data set and a generative model. However, our work also concentrates on delivering an immersive experience, facilitating fast prototyping, and emphasizing the essential interactions necessary for seamlessly integrating natural hand control into terrain authoring.

2.1 Sketch-based Terrain Generation

Sketch-based terrain generation leverages the intuitive nature of sketching and transforms sketch elements such as contours, lines and shapes into detailed and realistic 3D terrains. Previous approaches have often revolved around 2D user interfaces using flat screens, mice and keyboards, and 2D or 2D-like input modalities

in which users edit altitude cues on terrain surfaces in a view-dependent manner, akin to traditional CAD-style (Computer-Aided Design) interfaces [12, 31, 34, 35, 39]. For example, Guérin et al. [10, 11] proposed two pipelines that allow users to sketch on the terrain surface to generate an initial coarse terrain, while providing various interactive painting brushes for additional height adjustments. Therefore, users might need to perform a number of refinement operations for diversified level sets. Cohen et al. [3] presented their system Harold which creates 3D terrains by projecting 2D silhouette strokes from screen space into 3D curves from a side view. The corresponding terrain is procedurally generated to fit silhouette curves. Gain et al. [7] added to this silhouette approach an improved procedural generation and terrain footprint estimation. In these silhouette approaches, users can directly provide height cues for the intended terrain without incremental level-by-level refinement, but the viewpoint needs to remain fixed as users draw the silhouette since the method projects the silhouette onto a vertical plane and therefore needs to ensure a stable projection while sketching. These methods are therefore inherently 2D or 2D-like approaches.

Several prior works have investigated using handheld controllers for sketching in VR [1, 38, 39]. Additionally, some researchers have explored integrating hand-tracking into their sketching interfaces for immersive 3D modeling by using either camera-based hand-tracking or wearable tracking devices[5, 15, 19, 25, 33]. Specifically, Wong et al. [37] presented their VR-based 3D object and terrain editing System with AI assistance. The system seems to be a VR version of the work presented by Guérin et al. [10]. While these systems have demonstrated innovative approaches to enhancing user interaction for immersive systems, none of them has specifically focused on terrain generation, and most of them achieve 3D modeling by utilizing "Teddy"-like [13] techniques, using 2D sketching or 2D sketches projected from 3D sketching.

In contrast, our work fully utilizes the benefits of spatial interfaces and releases the power of free-hand drawing. Users can draw free-form mid-air strokes in VR from any position and any viewing angle, to effectively communicate elevation and terrain contour lines through 3D sketching.

2.2 Data-driven Terrain Generation

Data-driven methods informed by real-world data sets can closely mimic actual landscapes. These methods are adaptable to user preferences as they integrate with diverse data sources containing specific terrain features. For instance, Zhou et al. [42] introduced a system that creates terrains by merging user-sketched feature maps and height fields to synthesize patches sampled from data sets. With recent significant advancements in machine learning, the use of generative models has become one of the dominant techniques for terrain generation. These models, such as GANs [2, 10, 11, 22, 23, 32, 40], and diffusion models [17], can rapidly generate complex, coherent and realistic terrain patterns. Gain et al. [8] interactively generated terrains from a height field data set using parallel pixel-based texture synthesis. Guérin et al. [10, 11] integrated CGAN with interactive sketching interfaces to enable authoring of various terrain types and features. Perche et al. [24] generated terrain with different styles using improved StyleGAN2

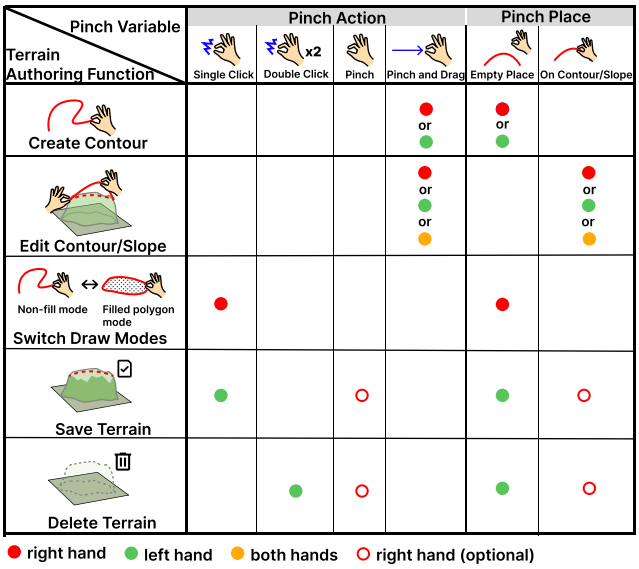

Figure 2: Interaction Design for Pinch-based Terrain Authorization. This chart shows how different pinch variables are combined to create different terrain authoring functions.

architecture. Lochner et al. [17] adapted diffusion models to improve landform variety and fidelity. Our work also uses CGAN conditioned on heightmaps. The height maps are converted from mid-air strokes, with Perlin Noise [26] added prior to the CGAN process.

2.3 Terrain Generation in Virtual Reality

VR inherently has the potential to make full use of spatial user interfaces. Modern VR platforms not only provide stereo views and spatial audio, but also offer untethered solutions with fewer space constraints and advanced functionalities such as hand tracking, gesture recognition and speech recognition. Prior work investigating the use of VR and gesture interfaces for CAD modeling and virtual content manipulation dates back to the late 90s [4, 41]. Zheng et al. presented a VR-based user interface for CAD modeling that used a glove with sensors for hand tracking and gesture detection and a graphical user interface for controls [41]. Coninx et al. demonstrated a hybrid 2D /3D user interface in VR with pinch-gesture-based interaction involved [4]. Pfeuffer et al. explored VR-based interactions combining gaze and freehand gestures such as targeting by gaze and manipulation by pinch [29]. Our work also uses hand tracking and a freehand gesture interface for combining sketch input and a control trigger. However, the focus of our work is to explore the possibilities of a freehand interface that can lets users use simple mid-air sketches for fast terrain prototyping.

3 SYSTEM DESIGN

Our objective is to design data-driven terrain authoring tools that use natural input modalities in VR for simple and fast 3D terrain prototyping. Our emphasis is on preserving artistic control while

lowering effort, to create an intuitive and immersive terrain authorization experience.

Based on these goals, we propose a pinch-based VR terrain authoring system that harnesses users' inherent understanding of 3D space to create and manipulate terrains by using hand gestures to draw contours or filled regions. The system then automatically generates realistic terrains. Users can further refine the terrain by pinching to edit shape, position and slope. In the following sections we elaborate on the pinch variables (the basic design component for our gesture interface), visual representation, and terrain authoring design.

3.1 Pinch Variables

The foundation of our interaction model is the pinch gesture. Pinching was selected because it is one of the most common gestures used for VR user interface design (e.g., Apple Vision Pro and Meta Quest's user interfaces both use pinch for item selection) and it can be reliably detected by most VR platforms (e.g., Unity has built-in pinch recognition). We explore various variables of this gesture, including different pinch actions, different pinch places, and different pinch hands. We combine different pinch gestures to enable various terrain authoring functions. This approach allows for a simple user interface that is easy to learn and fast to use. We classify pinch gestures as follows:

Pinch Actions. Our system recognizes different actions by varying the duration and frequency of pinching. *Single click* (pinch and release) acts as a toggle, which is useful for changing the state of interaction. For more complex commands, *double click* (two consecutive single clicks) is used for features requiring a higher degree of user confirmation. Additionally, *pinching on* interactive visual elements can be used to define the interaction target. For tasks requiring continuous input or 3D navigation, such as drawing or adjusting terrains, *pinch and drag* is ideal.

Pinch Hands. Our system differentiates between left and right hand pinching, which enables distinct interactions based on whether the *left hand*, *right hand*, or *both hands* are used. Left hand only clicking is used for functions such as saving or clearing the contour. Right hand only clicking triggers different actions or mode changes. Pinching with both hands provides two 3D positions that can be interpreted as an area selection and modification. This is particularly useful for tasks such as modifying a terrain area. Also, a two handed pinch can indicate a more complicated modification or state change that requires more user attention. Moreover, by pinching the contour with one hand and clicking with the other hand, users can target specific contours. For example, the user pinches a contour with the right hand and double clicks with the left hand to delete the contour.

Pinch Places. Pinching in different places, such as on pre-existing terrain elements or unoccupied space, helps the system to automatically interpret whether the user intends to edit the current terrain or engage in different interactions. Pinching *in empty spaces* without any pre-existing contours prompts the system to initiate new terrain creation or activate other distinct functionalities. This enables users to seamlessly transition between editing existing features and creating new terrain elements. Conversely, when a user pinches *on existing contours* or editable elements of the terrain, the system interprets this as an intent to modify or refine those terrain features. This allows for precise and targeted adjustments to terrain structures.

3.2 Visual Representation

Our system lets users create and modify the terrain's outlines, change its slope's shape and angle, and switch between creating a ridge, mesa, or canyon. We visualize this information as follows:

Visualization for Terrain Outline. In our system, the terrain outline is displayed as solid red lines. Due to our data-driven approach to generating realistic terrain, the actual terrain does not precisely align with the user-drawn outline, potentially causing the generated terrain to partially obscure the outline. To prevent confusion and ensure clear visibility, we render these red outlines to always appear above the 3D terrain layer. This approach guarantees that the outlines always remain visible to the user, even if the terrain mesh occludes them.

Visualization for Drawing Ridge, Canyon, and Mesa. Users can create ridges or canyons by drawing non-filled lines above or below ground level. To facilitate this, we represent ground level as a semi-transparent plane. The relative position of the user's hand to ground level is indicated by a color-coded vertical line, which turns yellow when the user's hand is above ground level and changes to blue when the user's hand is below ground level.
Users can use a filled polygon tool to create a mesa or flat-topped mountain rather than just drawing non-filled lines. This is shown in Figure 1 g-j. When the user switches to filling mode, an icon resembling a paint bucket appears next to their hand. For visual consistency, the polygon's outlines are rendered in red. This matches the color of non-filled contours, ensuring a cohesive appearance for drawing actions.

Visualization for Slope. As illustrated in Figure 1, we represent the terrain's slope using white lines alongside the red terrain outline. We depict two white lines for landforms such as ridges and canyons to indicate the two sides of the slope. We render a single white line for mesas created in the filled polygon mode. This slope visualization is activated only in terrain edit mode, specifically when the user pinches a contour, and it appears precisely at the point of pinching. We adopt this approach because displaying slope information continuously along the entire contour could lead to visual clutter. Moreover, users generally require slope details only when they are actively editing the terrain.

3.3 Terrain Authoring Functions

As shown in Figure 2, different combinations of pinch variables are applied to different features of the terrain authoring tool. These features include creating and modifying terrain outlines and slopes, switching between creation modes, and saving and deleting terrain. We will discuss this mapping in detail in the following section. Users can create various types of terrains by combining those actions. For example, a Great Basin-like desert mesa landscape can be created by using filled polygons and generating contours to define the mesa edges. Cliffs can be shaped by adjusting the slope lines of a contour via the edit terrain action.

Create Terrain Outlines. The user can pinch and drag in an empty place to draw terrain outlines from their pinching point. Because users might have different dominant hands, this function can be triggered using either the right or left hand.

Edit Terrain. Users can adjust existing contours by pinching and dragging with either hand. For single-point editing, they can pinch and drag a contour using one hand to change point's position in 3D space. To move a line segment on the contour, users can use one hand to pinch and drag the segment's starting point, while using the other hand to pinch and drag the ending point. This dual-hand approach is particularly useful for more involved tasks, such as repositioning an entire mountain or adjusting the elevation of a large terrain area.

Users can also modify the steepness and shape of slopes. As noted earlier, if the user pinches an existing contour with one hand, the slope line will be displayed as white lines intersecting the user's contours at the pinched point. The user can then use their other hand to adjust these slope lines, reshaping the slope by pinching and dragging points along the white lines. They can modify the slope's steepness by gripping the endpoint of the white line and moving it up or down, which rotates the entire white line, altering the angle between the slope lines and the corresponding red contour, thus changing the slope's steepness.

Switch Drawing Mode. Our system offers two distinct drawing modes: non-filled lines and filled polygons. Users can utilize non-filled lines to represent ridges when drawn above ground level or to form canyons if drawn below ground level. Drawing a filled polygon allows users to create a surface suitable for designing mesas or basins. To switch between these drawing modes, users simply need to do a single click action as it is shown in Figure2 with their right hand.

Save Terrain. If users are satisfied with the current terrain and don't require further modifications, they can specify which contour to save by pinching the contour with their right hand while single-clicking with their left hand in the empty place. Users can also save the entire terrain by via a single left hand click in the empty place. This action is particularly beneficial as it prevents unnecessary re-computation of the terrain, thereby enhancing the system's performance. Additionally, when users save the terrain, its corresponding red contours will vanish. This helps to tidy up the interface, reducing visual clutter and minimizing potential confusion as to which parts of the terrain are editable for the user.

Delete Terrain. If a user wishes to remove a specific contour, they can pinch it with their right hand while simultaneously double-clicking in an empty place with their left hand. Similarly, they can double-click with the left hand in an empty place to clear the entire terrain. We use a double-click pinch action to minimize the risk of inadvertently activating the delete function. This method serves as a confirmation step, ensuring that users intentionally commit to such significant actions.

4 IMPLEMENTATION

In general, our system generates terrains via two steps: terrain synthesis from user input, and terrain post-processing.

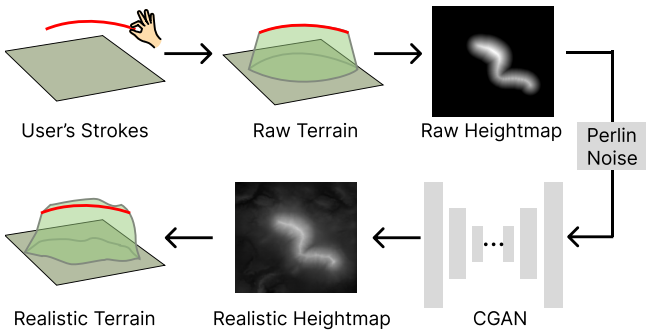

Figure 3: System pipeline.

The system collects user pinch positions, and uses this data to procedurally synthesize the terrain height map. The system interprets the height map data as a 256×256 gray scale image, with each pixel value representing normalized height at that location on the height map. To add realistic details on the generated height map, a conditional GAN (i.e. *Pix2Pix* [14]) performs image-to-image translation and synthesizes a height map image that incorporates features trained from actual terrain from satellite images. Using this synthesized realistic height map, the system then translates the gray scale value of each pixel to its corresponding elevation during generation of the 3D terrain.

Our system uses Unity3D for VR integration and terrain rendering. In addition, we incorporate a procedural texture module to dynamically generate textures based on terrain height and gradient. This helps display a natural and visually appealing representation of the terrain, but is not the primary focus of our system.

4.1 User Interface

Our system integrates the Oculus Interaction SDK package in Unity 3D with built-in features in the Meta Quest 3, which is our main platform for tests and study, to enable hand tracking for VR. Leveraging the Quest 3's camera-based hand-tracking, the SDK provides essential hand tracking functionality, including position and orientation tracking of hands, configuration tracking of fingers, and hand gesture recognition. The system recognizes pinch gestures and records their positions to construct and edit contours.
For editing, the system uses linear interpolation to diffuse the change at a single point to the rest of the unedited section. When editing with both hands as mentioned in previous section, the segment between both hands is unaffected by the interpolation. Rather, the two hands serve as the anchor to transform the segment. The visualization of this process is described in Figure 4.

Users can also edit the slopes on both sides of ridges by manipulating the slope line that controls terrain steepness. The result is then stored in the contour and is further used in later steps.

4.2 Terrain Synthesis

4.2.1 Synthesis Method. In traditional terrain authoring applications like the in-editor terrain tools for Unity and Unreal, terrain is stored as a 2D array of floats representing the height at each corresponding vertex, and is usually deformed by applying a brush mask onto the terrain based on user's input. The brush mask is an

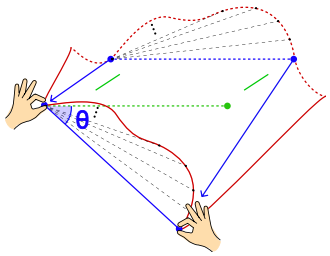

Figure 4: Visualization of modifying contour with both hands.

RGBA image kernel with embedded alpha information. By painting alpha values of the mask brush onto the terrain 2D array, the terrain is deformed.

We use a similar approach to this brush mask-based terrain reconstruction method by marching over each point on the 3D contour and applying the brush deformation based on the elevation of the points. Previous brush mask methods usually used only one uniform instance of the brush kernel, which made it hard to reconstruct diversified terrain characteristics, such as when the two sides of a mountain range have different slope steepness, or when there is a sharp cliff drop along the mountain range. The failure case of a steep drop-off is shown in Figure 5.

Gain et al. [7] used a sketching algorithm that maps the contour to a vertical plane as the silhouette and computes the boundary of the contour to generate 3D terrain. However, their algorithm places constraints on both silhouette and boundary. For example, the silhouette and the boundary may not fold back on or intersect itself. This solution is not suitable for our system, which emphasizes fast prototyping and using fewer interactions to create plausible results. Moreover, in VR-based sketching, users in have more freedom of movement compared to the traditional 2D interface. This freedom of control can lead to unstable sketches and irregular strokes when drawing in mid-air. To tackle this problem, we enhanced the traditional brush mask-based method by changing the alpha values of the brush mask along each vertex on the contour based on slope steepness and the contour gradient. The default brush kernel is defined by

$$a_{x,y} = a_0(1 - \sqrt{(x - x_0)^2 + (y - y_0)^2}) \quad (1)$$

Whereas $a_{x,y}$ is the value stored on (x, y), a_0 is the largest kernel value, and (x_0, y_0) is the kernel center. Then, based on the contour steepness and gradient, we adjust the kernel values to match the overall shape of the terrain that the user desires.

Figure 7 demonstrates two examples from the result of our terrain reconstruction approach. Our reconstruction method can preserve the abrupt slope variations that might be smoothed out in [7].

4.2.2 Post-processing with CGAN. *Pix2Pix* is a conditional Generative Adversarial Network designed for image translation [14]. We chose this model due to its capability in translating abstract height maps into realistic ones. Our workflow starts with the original height map and aims to obtain a more detailed height map that can closely resemble a real-life height map, which often contains erosion effects.

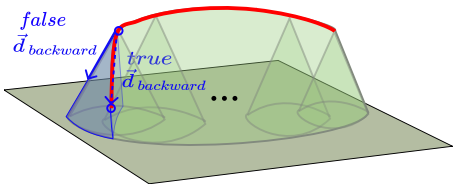

Figure 5: A potential failure case caused by the uniform brush mask overwriting the steep drop-off, which is colored in blue in this figure. Failure cases similar to this case is resolved by adjusting the brush mask per vertex on the contour.

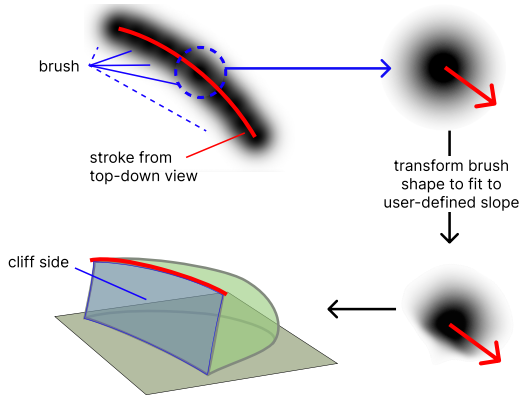

Figure 6: Visualization of adjusting the brush mask to create cliff on one side of the slopes. Notice that the brush alpha has been reversed for better visual representation, in which the darker area stands for higher alpha values.

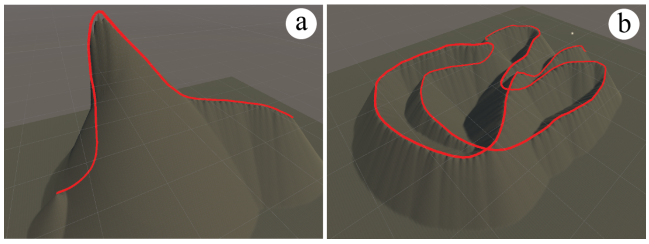

Figure 7: (a) An example cliff reconstructed by our approach. (b) A complex terrain example constructed by our approach preserves the sharp slope drops.

We train our synthesizer using the architecture outlined in *Pix2Pix*. For optimal performance, both input and output sizes are set to $256 \times 256 \times 3$ and batch size is 32. We choose a low resolution for the purpose of fast iteration during system development.

We process the original height map using a level set representation. The *Pix2Pix* model is trained to translate the level sets to the satellite height map images. The method of synthesizing a realistic terrain from level set input is described in [10] and is further elaborated by [22]. Consistent with the trained data set, the height

map deformed and gathered from the previous section will also be processed as a level set representation.

To gather terrain satellite images in real life, we use the Earth Engine API by Google, gathering 10000 height maps randomly cropped from 10 regions chosen based on the criteria of minimal human architecture, high quality satellite height map images, and diversity of geographical characteristics, from the NASA Shuttle Radar Topography Mission data set provided by the United States Geological Survey. After 10000 satellite height map images are sampled and converted to the unified resolution of 256×256 to align with the input and output sizes of the model, we apply the level set quantization on the samples and constructed the data set for training the model with the level set representation as input, and the corresponding satellite height map image as ground truth.

4.2.3 Effects of the added noise. After training the model with the data set that uses the level set representation of the height map as input, we process the height map deformed in the previous section to mimic the level set to get the corresponding output by the trained model, using the *Floor* function by slicing down the pixel values from the image of the raw height map to discrete levels. An additional noise layer is added before we floor the images of raw height map to level sets, as we find that the erosion effect exhibited by the level sets representation resembles the effect of Perlin Noise [26]. By adding Perlin Noise onto the original height map using an alpha mask, we can mimic and control erosion effects in our level set representation of the raw height map, and thus control the input of the *Pix2Pix* model and the strength of the natural erosion effect on the output, which solves the problem presented by previous work [10], in which the user study suggests that users would like a tool that allows them to freely control erosion strength to produce a smooth slope. Figure 8 demonstrates the preliminary study we have conducted to compare two different results from the same Pix2Pix model, with one of the inputs applying noise to produce an erosion effect.

By training the *Pix2Pix* model with the input of the level sets representation of the satellite height map, and the output of the unprocessed height map, we can use the model to translate raw Height maps to realistic height maps that resemble the actual height maps in our dataset. The *Pix2Pix* model in our system is optimized by Unity Barracuda Plugin. With Intel i9-12900KF 3.20 GHz CPU, 32GB RAM and Nvidia 3090 GPU, the average computational time for a 256×256 height map to generate in our system is 2.45 seconds. This result reflects the computation time for height map generation only. Other computation modules such as rendering run in parallel, and their computational costs are trivial.

5 USER STUDY

Given that our system is one of the first VR-based Terrain Authoring Tools, our initial focus is to conduct a feasibility study without a baseline. We evaluate our system with the "usage evaluation" approach as described in Ledo et al.'s HCI toolkit evaluation framework [16]. The methodology helps gather insights about the user experience by investigating the clarity of design concepts, ease of use and significance. Specifically, we aim to gather their feedback and understand whether users could efficiently and easily use our VR terrain authoring system to create various types of terrain. For this purpose, we ask participants to create three distinct types of terrain. Additionally, we provide them with the opportunity to freely explore and create any terrain with the system. At the end, we have an interview with the users and collect their feedback through a customized questionnaire and open-ended questions.

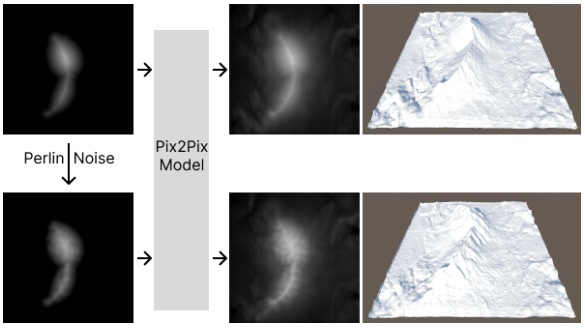

Figure 8: Examples of the use of Perlin Noise on the input. The input on the upper section produces a smooth terrain, the input on the lower section, adding an additional Perlin Noise layer, produces a rough terrain with erosion

5.1 Experimental Setup

Our study uses Oculus Quest 3 for VR display and hand tracking. We use Oculus Quest's Quest Link mode to run our system in Unity3D 2022.3.11f1 on the same PC we have mentioned in section 4.2.3. Each study takes around 40 minutes. We provide each participant with a $15 gift card if they complete the study.

5.2 Participants

We recruited 8 participants including 4 males and 4 females, age range 21-24 (mean 22.88, SD=1.36) from our local community. The participants have diverse backgrounds including expertise in 3D art, interactive media art, game design, computer science, film, and more. Their experiences with VR range from first-time users to daily VR users.

5.3 Study Protocol

Upon arrival, each participant is required to provide consent to participate in the study and agree to audio recording during the post-session interview. Then, we give them a brief introduction to our system. For the next stage, they watch a 2-minute tutorial video in which one of our developers demonstrates the features of our system for terrain authorization. This demonstration includes terrain creation and editing, switching between different terrain drawing modes (the surface filling mode and the stroke mode), saving, and deletion. After watching the tutorial video, they will have 5 minutes to get themselves familiar with the features they learned.

The study has two sessions: replication session and free exploration session. The replication session has 3 replication tasks in a shuffled order. Each task requires participants to create one of the three different terrain types: ridge, volcano, and canyon. During the task, the system displays an image of the corresponding terrain type, and participants have a maximum of 3 minutes to finish the

replication. The participants are informed to create a terrain that they believe closely resembles the one in the picture and try to finish the task as fast as they can if it is possible. Participants have the option to inform the experimenters to stop the time counting when they are satisfied with their creation within the 3-minute limit. During this process, the experimenters count the time to complete; the system records hand positions, the time of pinching and release, and the number of actions they perform.

To further understand the user experience, there is a free exploration session after the replication session. In this session, participants have 5 minutes to create anything they desire, with no restrictions on the types or quantity of terrain. They also have the flexibility to conclude the task at any point if they are satisfied with the results.

5.4 User Feedback Collection

After the sessions, the experimenters ask them to fill out a customized 5-point Likert scale about the level of ease-to-use, their preference for each system feature, and satisfaction with the results. At the end, there is a short interview with open-ended questions for their insights on the hand input interface, overall experience, limitations, and what features they expect the system to have in the future. In addition, we logged user behavioral data, such as timestamps for each task and action, hand usages, and positions, to provide objective behavioral evidence for the user's experience of the tool. All data was obtained with user consent.

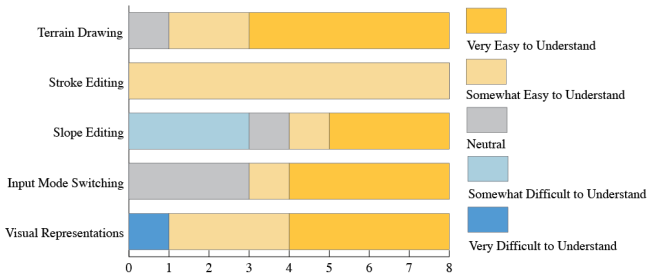

Figure 9: User study results on system's ease to understand.

5.5 Study Analysis

Result. We analyzed the study results from 8 participants. We conducted descriptive analysis for questionnaires and user log data and derived themes from interviews. All data sources based on both users' subjective self-reports and objective behavioral evidence were triangulated to provide an in-depth understanding of the system's efficiency, usability, ease of use, and strengths in 3D environments. By analyzing the data collected from the user study, we found that 19 out of 21 tasks were finished on time. For sketching a ridge, the average time used was 1 minute and 47 seconds with a standard deviation of 63.5 seconds; no participant failed to finish the task. For the task of sketching a volcano, the average time used was 1 minute and 36 seconds with a standard deviation of 45.33; 2 participants were unable to finish the task. For the task of sketching a canyon, it was 1 minute and 23 seconds with a standard deviation of 32.44, no participant failed to finish the task.

In terms of actions taken, the average number of actions to finish all three tasks is 162.5, with a standard deviation of 104.96. The average action of sketching is 65.83 with a standard deviation of 50.23, and the average action of modifying is 96.66 with a standard deviation of 55.54. The average time of a sketching action is 2.09 seconds with a standard deviation of 0.77 second; The average time of a modification action is 1.12 seconds with a standard deviation of 0.94 seconds. The statistics show that users generally tend to take more actions on modifying the terrain than creating the sketch, however, they tend to spend more time on a single action of sketching than that of editing. The total average time users have spent on sketching in a session is 137.58 seconds, compared to 108.26 seconds of time spent on editing, which demonstrates that users tend to take more time on the sketching, but a lot more small adjustments are added after the sketchings are done.

Usability. The interview shows that the system's features were generally adequate for terrain creation. In the questionnaire, half of the participants agreed that our system covers all the necessary tools for basic terrain prototyping. Participants liked the simple and intuitive pinching gesture for terrain authoring. 7 of the participants strongly agreed that the interaction design is efficient while not overwhelming (with 5 of them strongly agreeing on this), and 1 participant held a neutral attitude. The ability to quickly prototype terrains was highlighted by participants as a significant advantage. As **P1** pointed out, the system can help users generate VR terrains directly in VR space within minutes, and the generated terrains seem to be realistic and of high quality. The slope function was praised by **P2**, **P6**, and **P7**, as the function supports more control over terrain shapes. Based on the questionnaire results, half of the participants agree that the system's outputs resemble what they have in mind, and the other half neither agree nor disagree. This might be because our system uses a data-driven method to generate realistic terrain, so the generation results might not fit the participants' drawing, especially if the intended terrain in their minds is unrealistic.

However, participants mentioned that they would expect additional features like curve editing, undo functions, and improved visualization for slope editing. There could be further improvements in the visualization as well. For example, P7 mentioned that sometimes the visual cue of the slope in collided with their hand, making it hard to modify.

Easy to Learn. Based on the questionnaire results, all the participants agreed that they think they can master all the functions within a few minutes in the tutorial session (with 5 of them strongly agreeing on this), indicating a shallow learning curve. The minimalist interface design was appreciated for its focus on terrain creation without distractions (**P2, P3, P5**). **P4** mentioned that they were scared by the complicated interface of traditional modeling systems, but they are confident in using our system to create what they have in their mind. **P6** thinks our system is playful and enjoyable because they don't need to pay much effort to remember how to use each function, and they can put all their attention into the design process.

The interview shows that the pinch-based actions were found to be easy to understand and use by all the participants. Using pinch gestures to create and modify the terrain was intuitive, mirroring

actions in physical life (**P3, P4**), like "how pinching and dragging a rope in real life" (**P4**).

However, some participants accidentally triggered the pinching actions when they were unconscious about their hands. The system occasionally confused drawing actions and editing actions. **P7** suggested using different gestures for stroke creation and editing to avoid undesired triggers.

As shown in Figure 9, participants found most of the functions easy to learn, and the visual cue was clear. Some participants found the slope editing function is not easy to use because sometimes they cannot pinch accurately onto the slope line, making it hard to use. This is because they cannot correctly determine the relative position of their hands and the lines. **P4** mentioned that it is difficult to have a precise spatial sense when doing 3D sketching. For example, they were having a hard time drawing a closed circle because it is challenging to keep the circle's starting point and the ending point at the same height. **P3** mentioned the same issue and suggested adding visual cues for spatial relationships between the hand and the strokes to enhance precision in 3D space.

Benefits of Terrain Authoring in VR. The immersive VR experience supports a first-person immersive view; therefore, it can help users get a better understanding of the terrain (**P2, P7**). For example, it is easier and more intuitive for the participants to check the terrain from different angles in real-time by physically moving their bodies and rotating their heads while their position and viewing perspective in the virtual space are dynamically synced up with their motions (**P0**). Compared to traditional CAD-like interfaces, users don't have to use 2D user interfaces for 3D content by viewing from different viewports or interacting with a 2D screen, mouse, and keyboard (**P4**). This is particularly valuable in design processes for environments like stages and gardens, where designers can immersively experience the terrain from their end-users' perspective (**P0, P2**). By eliminating the potential detachment imposed by traditional 2D or 2D-like user interfaces, designers can immerse themselves in the spatial context, facilitating a more empathetic and user-centric design approach. Therefore, it could ultimately enhance the quality and user experience of the designed environments. In summary, the system was well-received for its usability in terrain creation, ease of use, and fast prototyping. The immersive experience and intuitive hand gestures were key strengths, although there were suggestions for additional features and improvements in precision and control.

6 DISCUSSION

For users who want to make the terrains with different types of characteristics, they need to train separate synthesizers. The reason behind it is that sketching is an abstract input method that could represent different features in different contexts. Separate CGAN synthesizers could interpret sketches to elicit more accurate characteristics for each terrain type. As a result, the same sketch could be interpreted as different landform styles depending on the user's intention. A potential drawback of our approach is that it requires effort to train different synthesizers with different datasets separately, therefore, it would be impossible for our pipeline to seamlessly generate the terrain for various landform styles without selecting from different synthesizers. In the future, we will try diffusion models [17] and other GAN-based techniques such as the improved StyleGAN2 architecture proposed by Perche et al. [24] We can then interpolate and traverse latent space to seamlessly generate diverse terrain features and styles [22].

From our user study, we noticed that users require significantly more actions to modify rather than to sketch (with 96.66 average actions for modifying, compared to the average of 65.83 actions for sketching). This shows that users may not be satisfied with their initial results and tend to make many minor adjustments, which emphasizes the importance of the editing function. We plan to conduct additional experiments and interviews to identify the underlying reasons behind these extended modification periods, and to explore ways to enhance the interaction features for modification.

The pinch gesture is the primary trigger for interactions. According to our user study, this gesture is easy and natural for participants to use, but brings certain limitations. For example, pinching is the toggle to switch drawing mode or to save terrain, and to draw sketches the user needs to pinch and move their hand. The system might have trouble distinguishing between switching mode and drawing when the user wants to draw a tiny stroke, because the system needs to ignore tiny movements below a certain threshold due to the fact that hands cannot be perfectly still. As a consequence, it would be hard for users to make very small adjustments, since any input action smaller than some threshold will be recognized as a click action. Some users pointed out that they sometimes triggered the pinch action or drew undesired strokes by accident. Furthermore, it is difficult to have stable hand gesture recognition when hands or parts of hands are obstructed. In future work we plan to enhance our design by introducing new gestures and multimodal inputs (such as speech) to improve user control and interaction experience. Also, current VR solutions such as Meta Quest 3 can have inaccurate hand tracking. To better analyze the impact of this inaccuracy, we plan to implement more precise alternatives, such as OptiTrack, for ground truth comparison.

Unlike the Gradient-based method introduced by Guérin [11], generation result from each contour in our system is computed independently, so contours will not affect each other if they are clustered. Instead, one contour may overwrite another. Also, users may not add weight to a contour by drawing multiple contours together. This problem will be addressed in our future work.

Some users reported difficulty in accurately determining the relative position between their hand and their drawings. As a result, those users found it difficult to pinch on the editable lines or sketch in the desired position. In the future, we plan to add more spatial cues to the system to enhance users' spatial perception of the drawings. These could include a spatial grid or highlighting a stroke as the user's hand approaches, enhancing precision and ease of use. Our system introduces interactive methods for using a VR interface to author terrain. In the future, we plan to conduct studies to learn more about the efficiency and learning affordances of our system as compared with previous terrain authoring tools.

REFERENCES

[1] Rahul Arora and Karan Singh. 2021. Mid-air drawing of curves on 3d surfaces in virtual reality. *ACM Transactions on Graphics (TOG)* 40, 3 (2021), 1–17.
[2] Christopher Beckham and Christopher Pal. 2017. A step towards procedural terrain generation with gans. *arXiv preprint arXiv:1707.03383* (2017).

[3] Jonathan M Cohen, John F Hughes, and Robert C Zeleznik. 2000. Harold: A world made of drawings. In *Proceedings of the 1st international symposium on Non-photorealistic animation and rendering*. 83–90.

[4] Karin Coninx, Frank Van Reeth, and Eddy Flerackers. 1997. A hybrid 2D/3D user interface for immersive object modeling. In *Proceedings Computer Graphics International*. IEEE, 47–55.

[5] John J Dudley, Hendrik Schuff, and Per Ola Kristensson. 2018. Bare-handed 3D drawing in augmented reality. In *Proceedings of the 2018 Designing Interactive Systems Conference*. 241–252.

[6] Alain Fournier, Don Fussell, and Loren Carpenter. 1982. Computer rendering of stochastic models. *Commun. ACM* 25, 6 (1982), 371–384.

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

[8] James Gain, Bruce Merry, and Patrick Marais. 2015. Parallel, realistic and controllable terrain synthesis. In *Computer Graphics Forum*, Vol. 34. Wiley Online Library, 105–116.

[9] Eric Galin, Eric Guérin, Adrien Peytavie, Guillaume Cordonnier, Marie-Paule Cani, Bedrich Benes, and James Gain. 2019. A review of digital terrain modeling. In *Computer Graphics Forum*, Vol. 38. Wiley Online Library, 553–577.

[10] Éric Guérin, Julie Digne, Eric Galin, Adrien Peytavie, Christian Wolf, Bedrich Benes, and Benoît Martinez. 2017. Interactive example-based terrain authoring with conditional generative adversarial networks. *ACM Trans. Graph.* 36, 6 (2017), 228–1.

[11] Eric Guérin, Adrien Peytavie, Simon Masnou, Julie Digne, Basile Sauvage, James Gain, and Eric Galin. 2022. Gradient Terrain Authoring. In *Computer Graphics Forum*, Vol. 41. Wiley Online Library, 85–95.

[12] Takeo Igarashi and John F Hughes. 2007. A suggestive interface for 3D drawing. In *ACM SIGGRAPH 2007 courses*. 20–es.

[13] Takeo Igarashi, Satoshi Matsuoka, and Hidehiko Tanaka. 2006. Teddy: a sketching interface for 3D freeform design. In *ACM SIGGRAPH 2006 Courses*. 11–es.

[14] Phillip Isola, Jun-Yan Zhu, Tinghui Zhou, and Alexei A Efros. 2017. Image-to-image translation with conditional adversarial networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 1125–1134.

[15] Hans-Christian Jetter, Roman Rädle, Tiare Feuchtner, Christoph Anthes, Judith Friedl, and Clemens Nylandsted Klokmose. 2020. " in vr, everything is possible!": Sketching and simulating spatially-aware interactive spaces in virtual reality. In *Proceedings of the 2020 CHI Conference on Human Factors in Computing Systems*. 1–16.

[16] David Ledo, Steven Houben, Jo Vermeulen, Nicolai Marquardt, Lora Oehlberg, and Saul Greenberg. 2018. Evaluation Strategies for HCI Toolkit Research. In *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems* (Montreal QC, Canada) *(CHI '18)*. Association for Computing Machinery, New York, NY, USA, 1–17. https://doi.org/10.1145/3173574.3173610

[17] Joshua Lochner, James Gain, Simon Perche, Adrien Peytavie, Eric Galin, and Eric Guérin. 2023. Interactive Authoring of Terrain using Diffusion Models. In *Computer Graphics Forum*. Wiley Online Library, e14941.

[18] Benoit B Mandelbrot and Benoit B Mandelbrot. 1982. *The fractal geometry of nature*. Vol. 1. WH freeman New York.

[19] Masatoshi Matsumiya, Haruo Takemura, and Naokazu Yokoya. 2000. An immersive modeling system for 3d free-form design using implicit surfaces. In *Proceedings of the ACM symposium on Virtual reality software and technology*. 67–74.

[20] Gavin SP Miller. 1986. The definition and rendering of terrain maps. In *Proceedings of the 13th annual conference on Computer graphics and interactive techniques*. 39–48.

[21] F Kenton Musgrave, Craig E Kolb, and Robert S Mace. 1989. The synthesis and rendering of eroded fractal terrains. *ACM Siggraph Computer Graphics* 23, 3 (1989), 41–50.

[22] Shanthika Naik, Aryamaan Jain, Avinash Sharma, and KS Rajan. 2022. Deep Generative Framework for Interactive 3D Terrain Authoring and Manipulation. In *IGARSS 2022-2022 IEEE International Geoscience and Remote Sensing Symposium*. IEEE, 6410–6413.

[23] Emmanouil Panagiotou and Eleni Charou. 2020. Procedural 3D terrain generation using Generative Adversarial Networks. *arXiv preprint arXiv:2010.06411* (2020).

[24] Simon Perche, Adrien Peytavie, Bedrich Benes, Eric Galin, and Eric Guérin. 2023. Authoring terrains with spatialised style. In *Computer Graphics Forum*, Vol. 42. Wiley Online Library, e14936.

[25] Helen Perkunder, Johann Habakuk Israel, and Marc Alexa. 2010. Shape Modeling with Sketched Feature Lines in Immersive 3D Environments.. In *SBIM*. 127–134.

[26] Ken Perlin. 1985. An image synthesizer. *ACM Siggraph Computer Graphics* 19, 3 (1985), 287–296.

[27] Ken Perlin and Eric M Hoffert. 1989. Hypertexture. In *Proceedings of the 16th annual conference on Computer graphics and interactive techniques*. 253–262.

[28] Adrien Peytavie, Eric Galin, Jérôme Grosjean, and Stéphane Mérillou. 2009. Arches: a framework for modeling complex terrains. In *Computer graphics forum*, Vol. 28. Wiley Online Library, 457–467.

[29] Ken Pfeuffer, Benedikt Mayer, Diako Mardanbegi, and Hans Gellersen. 2017. Gaze+ pinch interaction in virtual reality. In *Proceedings of the 5th symposium on spatial user interaction*. 99–108.

[30] Pascale Roudier, Bernard Peroche, and Michel Perrin. 1993. Landscapes synthesis achieved through erosion and deposition process simulation. In *Computer Graphics Forum*, Vol. 12. Wiley Online Library, 375–383.

[31] Kristofer Schlachter, Benjamin Ahlbrand, Zhu Wang, Ken Perlin, and Valerio Ortenzi. 2022. Zero-shot multi-modal artist-controlled retrieval and exploration of 3d object sets. In *SIGGRAPH Asia 2022 Technical Communications*. 1–4.

[32] ryan rs spick and james walker. 2019. Realistic and textured terrain generation using GANs. In *Proceedings of the 16th ACM SIGGRAPH European Conference on Visual Media Production*. 1–10.

[33] Payam Tabrizian, Anna Petrasova, Brendan Harmon, Vaclav Petras, Helena Mitasova, and Ross Meentemeyer. 2016. Immersive tangible geospatial modeling. In *Proceedings of the 24th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems*. 1–4.

[34] François-Xavier Talgorn and Farès Belhadj. 2018. Real-time sketch-based terrain generation. In *Proceedings of Computer Graphics International 2018*. 13–18.

[35] Osama Tolba, Julie Dorsey, and Leonard McMillan. 1999. Sketching with projective 2D strokes. In *Proceedings of the 12th annual ACM symposium on User interface software and technology*. 149–157.

[36] Richard F Voss. 1985. Random fractal forgeries. In *Fundamental Algorithms for Computer Graphics: NATO Advanced Study Institute directed by JE Bresenham, RA Earnshaw, MLV Pitteway*. Springer, 805–835.

[37] Shing Ming Wong, Chien-Wen Chen, Tse-Yu Pan, Hung-Kuo Chu, and Min-Chun Hu. 2022. GetWild: A VR Editing System with AI-Generated 3D Object and Terrain. In *Proceedings of the 30th ACM International Conference on Multimedia*. 6988–6990.

[38] Burkhard Wünsche, Daniel Keymer, and Robert Amor. 2010. Sketch, click, plug and play: accelerated design of virtual environments by integrating multimedia and sketch content into game engines. In *Proceedings of the 11th International Conference of the NZ Chapter of the ACM Special Interest Group on Human-Computer Interaction*. 33–40.

[39] Robert C Zeleznik, Kenneth P Herndon, and John F Hughes. 2006. SKETCH: An interface for sketching 3D scenes. In *ACM SIGGRAPH 2006 Courses*. 9–es.

[40] Jian Zhang, Chen Li, Peichi Zhou, Changbo Wang, Gaoqi He, and Hong Qin. 2022. Authoring multi-style terrain with global-to-local control. *Graphical Models* 119 (2022), 101122.

[41] JM Zheng, KW Chan, and Ian Gibson. 1999. A VR based 3D graphics user interface for CAD modeling system. In *International Design Engineering Technical Conferences and Computers and Information in Engineering Conference*, Vol. 19715. American Society of Mechanical Engineers, 1029–1036.

[42] Howard Zhou, Jie Sun, Greg Turk, and James M Rehg. 2007. Terrain synthesis from digital elevation models. *IEEE transactions on visualization and computer graphics* 13, 4 (2007), 834–848.