UNIVERSITY OF CAPE TOWN

DEPARTMENT OF COMPUTER SCIENCE

CS Honours Project Final Paper 2024

Title: Planetary Generation using Texture Synthesis

Author: Sam Frost

Project Abbreviation: EarthGen

Supervisor(s): Professor James Gain

Category	*Min*	*Max*	Chosen
Requirement Analysis and Design	*0*	*20*	0
Theoretical Analysis	*0*	*25*	0
Experiment Design and Execution	*0*	*20*	15
System Development and Implementation	*0*	*20*	15
Results, Findings and Conclusions	*10*	*20*	20
Aim Formulation and Background Work	*10*	*15*	10
Quality of Paper Writing and Presentation	*10*		10
Quality of Deliverables	*10*		10
Overall General Project Evaluation (*this section allowed only with motivation letter from supervisor*)	*0*	*10*	0
Total marks		**80**	**80**

Planetary Generation using Texture Synthesis

Honours Project

Sam Frost
University of Cape Town
South Africa
frssam005@myuct.ac.za

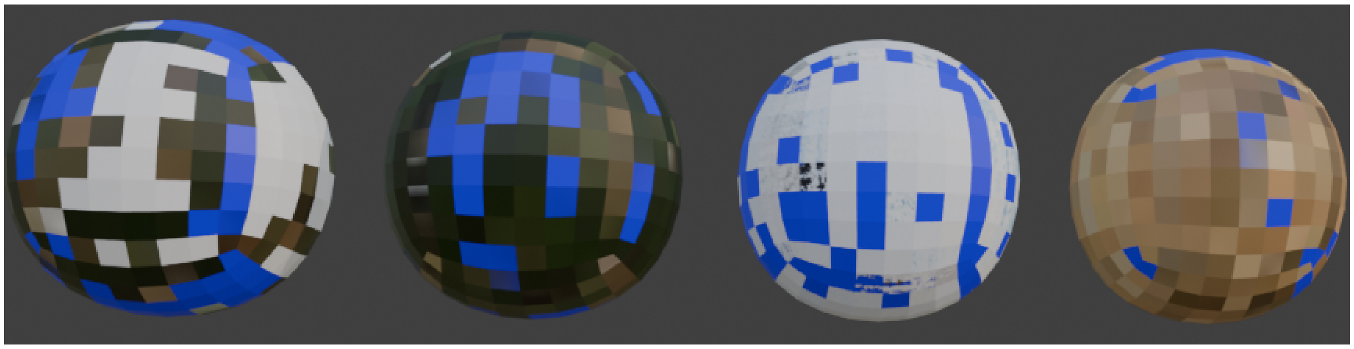

Figure 1: Our Planetary Generation method using Texture Synthesis. From left to right: Earth-like planet, Grassland planet, Snow planet, and Desert planet.

ABSTRACT

Synthetic terrain generation techniques can produce realistic, efficient, and adaptable terrains. These techniques combine elements of existing terrain data to produce synthesised terrain. Researchers classify terrain generation into procedural, simulation, and example-based methods. Procedural terrain generation replicates natural phenomena, simulation-based terrain generation mimics geomorphological processes, and example-based generation uses actual terrain data. We focus primarily on the example-based method of texture synthesis. Texture synthesis algorithmically constructs an output image based on the structural characteristics of an input image. Our texture synthesis method is a parallel pixel-based approach to generate realistic terrain from input satellite data, as shown in Figure 1. In this paper, we utilise example-based methods because they allow for increased realism due to their use of actual terrain data. However, current implementations of example-based methods lack realism when applied to a planetary scale, as the large spatial extents and varying environmental factors introduce complexities. Addressing these issues is crucial to improving the accuracy of planetary terrain simulations. In addition, example-based methods have significant issues regarding polar distortion. We construct a globe interface to evaluate the ability of our texture synthesis method to generate realistic planets. The evaluation process addresses realism, efficiency, scalability, and user control, emphasising the importance of these aspects in terrain generation.

CCS CONCEPTS

• **Computing methodologies → Shape modeling**; *Image manipulation, Texturing, Image-based rendering*.

KEYWORDS

Terrain authoring, terrain generation, and texture synthesis.

1 INTRODUCTION

Films, games, simulations, training systems, and epidemiology research utilise terrain generation techniques [29]. These techniques can provide the large-scale natural landscapes that these applications require. This paper focuses on bare-earth terrain without trees, bushes, and manufactured structures. Due to the underlying geomorphological complexities of bare-earth terrain and its diverse scales, attaining realism in computer-generated terrain can be challenging [29]. In addition, issues around efficiency and availability of useful data are evident. Procedural, simulation-based, and example-based terrain generation methods have attempted to address these issues, as Section 2 explains. Example-based methods use real-world terrain data to synthesise terrain. These methods can provide realism, time and space efficiency, and user control. However, they rely heavily on the quality of their input data and can produce undesirable results if the input data contains anomalies [16]. In this paper, we evaluate an example-based method known as texture synthesis. Texture synthesis generates an output image with characteristics similar to an input image by reassembling areas from the input image, as shown in Figure 2 [15]. Research in terrain generation is expansive. However, researchers need to conduct more substantial studies on terrain generation at the planetary scale. This research could improve scientific exploration, global environmental modelling, gaming, and virtual reality [26]. Addressing planetary scale issues such as large spatial extents and varying environmental factors could improve the accuracy of planetary terrain simulations. This paper evaluates texture synthesis as a method for global terrain generation.

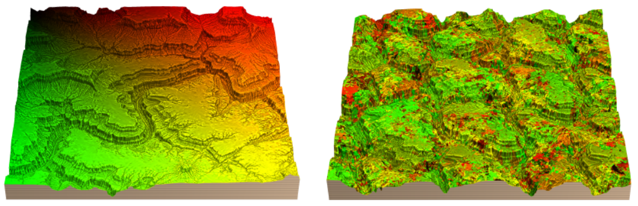

Figure 2: Input terrain (left) mapped to red (x-axis) and green (y-axis). Output terrain (right) coloured according to input coordinates. Gain *et al.* [14] illustrates the reassembling of areas from an input to create an output with similar characteristics. Figure Source: Gain *et al.* [14].

This paper addresses the broad research problem of the lack of a complete algorithm for efficiently generating realistic terrain directly onto a globe on a planetary scale. We aim to accomplish this using a texture synthesis method and a globe interface. The texture synthesis method allows for efficiency and realism as synthesising a realistic output terrain requires fewer resources than other example-based methods. The simplistic design of the globe interface will provide an accurate illustration of the different types of generated terrain. Formally, we aim to address the following research questions:

(1) Does a texture synthesis method allow for the realistic generation of Earth-like planets?
(2) Does a mesh cube-based globe interface enable the illustration of Earth-like planets directly onto a globe without distortion?

During the experimental evaluation of the texture synthesis method, we will focus on the realism of its generated planets to answer research question 1. We will assess realism by visually comparing the method's input and output terrains. Section 5 will discuss the reason for this form of evaluation and also include evaluations for efficiency and scalability. Efficiency will be measured by quantitatively evaluating the time and space efficiency of the texture synthesis method. Scalability will be evaluated based on extent and precision. The evaluations will include an evaluation of the globe interface and will address research question 2.

After reviewing related work (Section 2), we will discuss the materials and methods (Section 4) used. The Section on evaluation (Section 5) delves into the experiment design and implementation, which assessed the texture synthesis method and the globe interface. The evaluation results will then be presented and discussed in Section 6. We draw conclusions based on our results in Section 7, and then the paper's limitations and areas for further research are addressed (Section 8).

2 RELATED WORK

We must thoroughly review related prior work to understand the methods discussed in this paper. We discuss traditional terrain generation approaches and planetary terrain generation. Our work will consider these related works to overcome their limitations.

2.1 Traditional Approaches to Terrain Generation

Traditional terrain generation approaches are procedural, simulation-based, or example-based. These approaches encompass many methods with different advantages and disadvantages.

2.1.1 Procedural Generation. Procedural generation approaches replicate natural phenomena using terrain properties such as fractal characteristics [29]. Procedural generation differs from other traditional approaches because it does not simulate actual geomorphological processes or use actual terrain data; instead, it reproduces the effects using algorithms and parametric specifications. Researchers imitate natural phenomena using fractional Brownian motion [13, 25], subdivision schemes [10], and faulting [12]. In a global context, Tripkovic [37] developed a procedural generation tool using a mesh sphere. Although large-scale terrain generation frequently uses procedural methods due to their time efficiency and generality [4], procedural approaches can be highly computationally demanding at larger scales, leading to poor space efficiency and unsatisfactory realism. Therefore, procedural planetary terrain generation would not be viable [15].

2.1.2 Simulation-based Generation. Simulation-based methods imitate actual geomorphological processes, such as erosion phenomena [15]. These methods primarily use hydraulic, thermal, and tectonic erosion. Hydraulic erosion is water movement against bedrock under or on the Earth's surface [2]. Researchers simulate hydraulic erosion using either Eulerian [2] or Lagrangian [20] methods. Gravity and the thermal expansion of water beneath the Earth's surface create thermal erosion [28]. When using tectonic erosion, tectonic plate movement simulates the formation of mountain ranges [5, 6, 27]. Simulation-based terrain generation offers geological accuracy due to the underlying physics of the simulation processes. However, they are computationally expensive and do not scale well [15]. Therefore, simulation-based planetary terrain generation would be impractical for our purposes [15].

2.1.3 Example-based Generation. Example-based methods generate terrain using actual terrain data. The terrain data comes in many forms, including Digital Elevation Models (DEMs) and satellite data. DEMs are clusters of altitude values stored in a two-dimensional grid [15].

Machine learning is an example-based terrain generation method that trains models on extensive input data. Substantial pre-processing time is required, but machine learning methods are highly efficient during terrain generation. Guérin *et al.* [16] used a conditional generative adversarial network (CGAN) to generate terrain. An adaptation from the image-to-image translation of Isola *et al.* [19], the CGAN uses a generator to create new exemplar terrains and a discriminator to compare generated terrain with input DEMs. However, CGAN methods can result in minor grid lines in the output terrain [16]. Denoising Diffusion Probabilistic Methods (DDPMs) [18, 44] or diffusion models generate terrain by iteratively introducing noise and then denoising an input. Lochner *et al.* [24] found diffusion models effective for user-specified terrain generation. Overall, machine learning methods provide a promising avenue for terrain generation, including at larger scales.

Texture synthesis methods reassemble regions of an input image to create an output image with similar characteristics [15]. Texture synthesis methods for terrain generation include pixel-based and patch-based. Patch-based methods combine terrain patches from the input terrain data [9]. A patch-based method by Zhou *et al.* [47], seen in Figure 3, took DEMs and matched them to a user sketch. In this patch-based method, graph cuts and Poisson editing join the patches [21, 31]. Zhou *et al.*'s [47] method has limitations in efficiency, user control, and output, sometimes containing evident patch seams. Tasse *et al.* [35] improved on Zhou *et al.*'s [47] work by using a Shepard gradient interpolation [33] process for patch merging. This removes seams (artefacts) caused by overlapping patches because these artefacts look unrealistic [15, 35]. Lefebvre and Hoppe [22] present a parallel pixel-based texture synthesis method based on neighbourhood matching [43]. They accelerate neighbourhood matching using coherent synthesis [1] and utilise Tong *et al.*'s [36] k-coherence search to increase variety in the output image. Lefebvre and Hoppe's implementation produces efficient and accurate results but with limited exploration of large-scale terrain generation and runs primarily on the GPU. Gain *et al.* [14] adapted the multiresolution stack of Lefebvre and Hoppe [22], improving efficiency and user control [15].

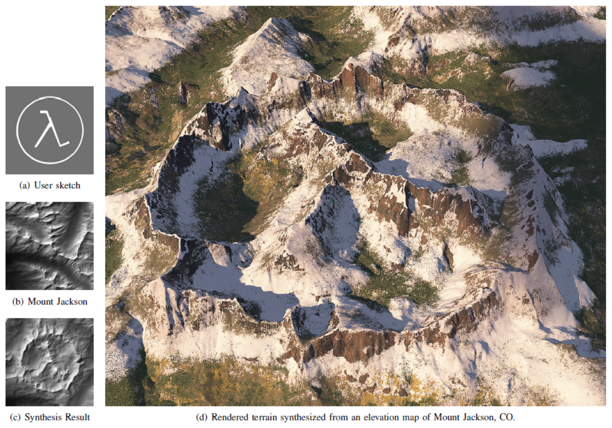
(a) User sketch
(b) Mount Jackson
(c) Synthesis Result
(d) Rendered terrain synthesized from an elevation map of Mount Jackson, CO.

Figure 3: Output using Zhou *et al.*'s [47]'s synthesis algorithm with an inputted user sketch and a DEM of Mount Jackson from the U.S. Geological Survey [38]. Figure Source: Zhou *et al.* [47].

Example-based methods can produce terrains with high realism, user control, suitable scalability, and real-time performance, which lends them to planetary generation. However, the output quality of example-based methods directly depends on the quality of the input data [15].

2.2 Planetary Generation

In planetary generation, the use of procedural methods is frequent. Vitacion and Liu [42], Villa *et al.* [41], and d'Oliveira and Apolinário [11] all used procedural methods to generate planetary terrain. Vitacion and Liu [42] experimented with generation onto a spherical object. Villa *et al.* [41] deviate from standard Earth-like planetary generation and investigate moon-like and asteroid terrains. d'Oliveira and Apolinário [11] apply their multiresolution procedural method to planetary generation for electronic games. Section 2.1.1 discusses that procedural methods can replicate tectonic movements to generate planets. Cortial *et al.* [8] accomplish this by approximating tectonic phenomena using parameters such as crust type, thickness, and elevation. However, as no direct formulae exist to evaluate plate deformation, physical accuracy cannot be guaranteed. In a separate paper, Cortial *et al.* [7] illustrate a subdivision scheme that produces realistic planets through iterative refinement, but user control is limited. Borg [4] uses an example-based machine learning approach to planetary generation with promising results, especially in scalability and user control.

Existing methods for planetary generation fail to achieve the problematic trifecta of realism, scalability, and efficiency. As Cortial *et al.* [7] noted, procedural approaches produce terrains lacking in true geological fidelity as they struggle to simulate complex, multi-layered planetary phenomena such as atmospheric interactions and hydrological cycles. Without consideration of the long-term dynamic evolution of terrain, the resulting planets can become static and unrealistic [28]. Moreover, procedural techniques can struggle with scalability, where ensuring consistent terrain generation at varying levels of detail becomes computationally expensive or yields artefacts, especially when dealing with non-spherical bodies like moons and asteroids, as explored by Villa *et al.* [41]. Existing approaches to planetary generation tend to be computationally expensive with low space and time efficiency. Borg's [4] machine learning implementation highlights this with considerable preprocessing time and high-performance computing resources required.

3 ETHICAL CONSIDERATIONS

We have given ethical consideration to data, code, intellectual property and licensing. All data used is freely available and without copyrights. Additional information regarding data is in Section 4.1. We used original code, open-source libraries, and publicly available software. All intellectual property created is the property of the University of Cape Town (UCT), as set out in the UCT Intellectual Property Policy [30].

4 MATERIALS AND METHODS

This paper centres around three main elements: the input data, the globe interface, and the texture synthesis method. We input the data into the texture synthesis method, which returns an output terrain illustrated using the globe interface. As with all example-based methods, the quality of the input data is crucial in producing good results. Therefore, we examine the data retrieval and processing approaches to illustrate the input data. In addition, we discuss the main features of the globe interface and texture synthesis methods, as well as the standard algorithms and data structures used.

The primary programming language is Python (version 3.10.11) [39], used in the data retrieval and processing, globe interface, and texture synthesis. Each section will state the reasoning behind this choice and its suitability for that particular element. We have adhered to strict documentation standards with detailed README files, in-code comments, and consistent coding conventions, ensuring clarity and maintainability.

4.1 Data Retrieval and Processing

The quality of input data is crucial, so we have carefully considered data retrieval and processing methods. We classify the input data into satellite and elevation data. The data is from NASA's Blue Marble Next Generation [34]. As previously discussed, the data is freely available and without copyrights. The Blue Marble Next Generation data was used for its high quality and global coverage, improving the accuracy of outputs. The data originates from NASA's Terra Moderate Resolution Imaging Spectroradiometer (MODIS). We use its land surface reflectance data. The satellite dataset includes spatial resolutions of 15, 60 and 240 arc-seconds, roughly 500-metre, 2-kilometre, and 8-kilometre spacing at the equator [34]. For the elevation data, the land uses the same spatial resolutions mentioned above, and the ocean uses a 2-kilometre spatial resolution. Therefore, we use the dataset at a spatial resolution of 2 kilometres to preserve consistency across all the data. Stockli et al. [34] apply corrections to the data to remove the distortion caused by molecules, aerosol scattering, and gaseous absorption [40].

After retrieval, we split the data into patches of equal dimensions using a Python [39] algorithm. Due to the equal dimensions of patches, we can have minor data loss, but it should not significantly affect the resulting output terrain. This occurs due to the unequal mapping from an input image resolution of 21504x10752 to patches with dimensions of 64x64. The texture synthesis method and globe interface then use these patches to generate the planet's terrain. We use patch sizes of 64x64 pixels, as this aims to balance space and time efficiency with the input image resolution and, therefore, the realism in the output terrain. The patches can then be inputted into the texture synthesis method to produce a synthesised output terrain.

4.2 Globe Interface

We use a globe interface to illustrate generated terrain accurately on a globe, as shown in Figure 4. We use Blender [3] for the globe interface because its many tools and resources allow for many ways to construct our system. Blender [3] also supports Python [39] scripting, allowing us to remain consistent throughout our systems. The design and implementation of the globe interface minimises distortions and artefacts created by placing the generated terrain onto the sphere. Polar distortion is a significant issue in a planetary generation. In Borg's [4] implementation of a machine learning-based planet authoring model, they experienced distortion due to the mapping from a plane to a sphere. Our globe interface aims to address this issue.

We construct our globe using a subdivided mesh cube, as shown in Figure 5. We start with a cube and apply subdivision, which converts it to a mesh roughly approximating a sphere. Using a mesh cube instead of a sphere disperses the distortion over the subdivided cube's edges. The dispersal prevents the distortion from accumulating at the poles and thus becoming extremely evident. We set subdivisions to 3, creating 384 cube faces to place the generated terrain. As the number of subdivisions increases, the distortion will decrease, and the mesh cube will look more spherical. However, efficiency will decrease due to increased cube faces and patches. The second way we decrease distortion and artefacts is by blending. We blend adjacent patches in the texture synthesis method as it

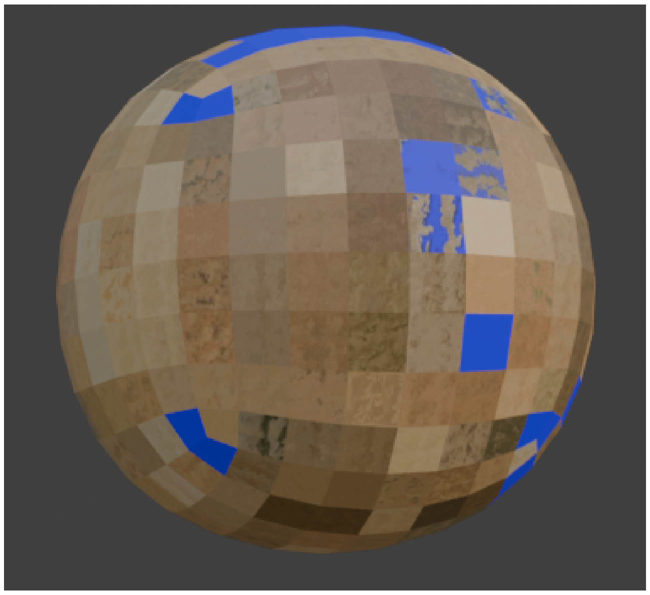
Figure 4: Sand planet generated using our texture synthesis method and globe interface.

allows for greater cohesion when completed during the synthesis process. The tiling method will be elaborated on in section 4.3.4. Our globe interface, as shown in Figure 5, contains four buttons used to set the type of planet that will be generated. There is an *Earth*-like planet, a *Snow* planet, a *Grassland* planet, and a *Desert* planet. An option to load a pre-generated planet or run the texture synthesis on new input data is available.

The globe interface is highly reliable due to its simplicity. We achieve high maintainability for the globe interface as changes and additions are straightforward using Blender [3]. We have designed the globe interface to ease portability through its simplicity. However, we recommend using Blender [3] as it has various useful tools such as UV editing, texture painting, and scripting.

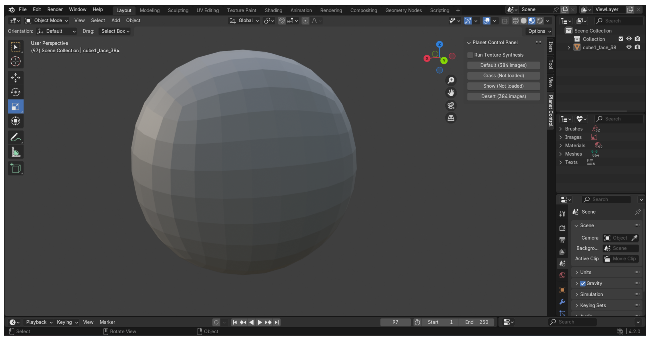

Figure 5: Our globe interface in Blender [3] utilising the subdivided mesh cube.

4.3 Texture Synthesis

The texture synthesis method is crucial in our planetary generation process. We base our method on Lefebvre and Hoppe's [22] implementation because it has successfully generated high-quality terrain. Gain et al. [14] also based their synthesis framework on this implementation, which yielded high realism, efficiency, and scalability. The texture synthesis method takes in an input terrain (exemplar) and creates a synthesised output terrain with characteristics similar to the input terrain. We run our texture synthesis method in parallel to improve efficiency. For an exemplar size of 64x64, six processes running in parallel is optimal. Our texture synthesis method consists of four parts: *upsampling, jitter, correction*, and *tiling*. An important point is that the exemplar comprises colour values corresponding to the colour of each pixel, shown here as u. In contrast, the synthesised texture contains reference coordinates (p) to a pixel in the exemplar. Therefore, the synthesised texture itself does not store its colour values. As Lefebvre and Hoppe [22] did, we apply upsampling, jitter, and correction algorithms to each level in a Gaussian stack to achieve high-quality synthesis. A Gaussian stack is a series of progressively blurred images at the same resolution, created without subsampling, to preserve fine details across different scales. The Gaussian stack is created by applying Gaussian filtering to an augmented exemplar to form the coarser levels. We double the exemplar's dimensions to make the augmented exemplar [22]. Figure 6 shows our Gaussian stack for a common exemplar for texture synthesis.

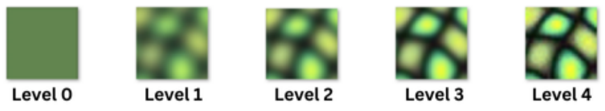

Figure 6: Successive levels of our Gaussian stack illustrating decreasing filtering.

Our synthesis method follows the same general process for all inputs. While traversing through the coarse-to-fine levels of the synthesised terrain and the exemplar in the Gaussian stack, we:

(1) Upsample the parent coordinates
(2) Randomise the synthesis coordinates using jitter
(3) Correct pixel values through several passes

Upsampling increases the resolution of the synthesised image. Jittering introduces variability and disrupts uniformity. Correction visually refines the output image to smooth inconsistencies created by the jitter. h_l, used below, refers to the standard output spacing of the exemplar coordinates. We set $h_l = 2^{L\text{-}l}$ for our Gaussian stack implementation, where L is equal to the number of levels in the Gaussian stack and l is equal to the current level.

4.3.1 Upsampling. Upsampling is an algorithm used to improve image quality by repeatedly increasing resolution by adding new data points between existing ones. We create a finer image from the coarser level below by upsampling the coordinates of the parent pixels. Four children coordinates are assigned the scaled parent coordinates plus a child-dependent offset, as in Lefebvre and Hoppe [22]:

S_l represents the synthesised texture at the level l. p is the synthesis coordinates, m is the input image dimensions and $\Delta \in \left\{\begin{pmatrix}0\\0\end{pmatrix}, \begin{pmatrix}0\\1\end{pmatrix}, \begin{pmatrix}1\\0\end{pmatrix}, \begin{pmatrix}1\\1\end{pmatrix}\right\}$.

$$S_l[2p+\Delta] := \left(S_{l-1}[p] + \left\lfloor h_l \left(\Delta - \begin{pmatrix}0.5\\0.5\end{pmatrix}\right)\right\rfloor\right) \mod m.$$

For our purposes, we do not increase the output size beyond the input size and maintain a 1:1 ratio between input and output resolution.

4.3.2 Jitter. Jitter is an algorithm that perturbs pixel values in the synthesis image to increase variance and originality. Using a hash function ($\mathbb{H}: \mathbb{Z} \to [-1,1]^2$), we introduce spatially deterministic randomness to the image at each synthesis level. Per-level randomness, r_l, is used to set the strength of the jitter function. It ranges from 0 (no jitter) to 1 (complete jitter). The less jitter used, the closer the output image will be to the exemplar. Conversely, the more jitter used, the more unique the output terrain is. As in Gain *et al.* [14], we set $r_l = 0.4$. We base our jitter method on the following formula by Lefebvre and Hoppe [22]:

$$S_l[p] := (S_l[p] + J_l(p)) \mod m, where J_l(p) = \left\lfloor h_l \mathbb{H}(p) r_l + \begin{pmatrix}0.5\\0.5\end{pmatrix}\right\rfloor$$

The output-spacing factor, h_l, decreases the jitter at finer levels [22]. We use a constant hash seed for consistent outputs, aiding comparisons.

4.3.3 Correction. The correction algorithm alters the jittered coordinates to recreate neighbourhoods in the synthesis image similar to those in the exemplar. Our correction algorithm replaces a pixel's value, if necessary, based on the value of its surrounding pixels to make the output more coherent with the input. We conduct two correction passes for improved results as we correct output pixels in parallel and, therefore, cannot simultaneously consider their neighbours' corrected pixel values. Conducting two passes allows the correction to revisit and refine areas where the initial pass lacked sufficient information about the correct values of neighbouring pixels. For all the pixels at the current level in the exemplar and the synthesised texture, we gather their 5x5 neighbourhoods. Specifically, we attain the pixel values for all the pixels in a 5x5 pixel area around u. We project the neighbourhoods into a lower dimensional space using a principal component analysis (PCA), as in Lefebvre and Hoppe [22]. The resulting vectors, $N_s(p)$ and $N_e(u)$ represent the 5x5 neighbourhoods for each pixel in the synthesis texture $(N_s(p))$ and the exemplar $(N_e(u))$. PCA reduces both memory and time requirements [22]. At the end of the correction process, we assign the synthesis pixel, p, with the exemplar pixel value, u, with the most similar neighbourhood.

We implement Ashikhmin's [1] texture synthesis algorithm to improve the time efficiency of our neighbourhood matching technique. Ashikhmin [1] accelerates neighbourhood matching by considering u only when the 3x3 neighbours surrounding p translate to it. In scanline order, each 3x3 neighbour of p in the synthesis image generates a candidate pixel according to its original position (u) in the exemplar, as shown in Figure 7. The acceleration is due to the drastic reduction in the number of comparisons required. We provide additional variation to the output by using a k-coherence search algorithm adapted from Tong *et al.* [36]. The k-coherence search method precomputes exemplar pixels with similar 7x7 neighbourhoods. Our adaptation was to store only the most similar 7x7 neighbourhood coordinates and remove the u that Tong et al. [36] had as the first entry. The u was kept by Tong *et al.* [36] as a coordinate reference. However, we reduce memory by storing the neighbourhoods in a two-dimensional array at the position of their coordinate and, therefore, do not have to store u again. We separated the neighbourhood stored during the k-coherence search by at least 5% from u, as in Zelinka and Garland [45]. We encourage patch formation by penalising jumps, following the approach of Hertzmann *et al.* [17]. We denote the penalty as k and apply it after neighbourhood comparisons. The above techniques create a candidate set, $C^l_{i_{min}}$, comprising of u's using Ashikhmin's [1] coherent synthesis method and their corresponding closest matching 7x7 neighbourhood coordinate using Tong *et al.*'s [36] k-coherence search.

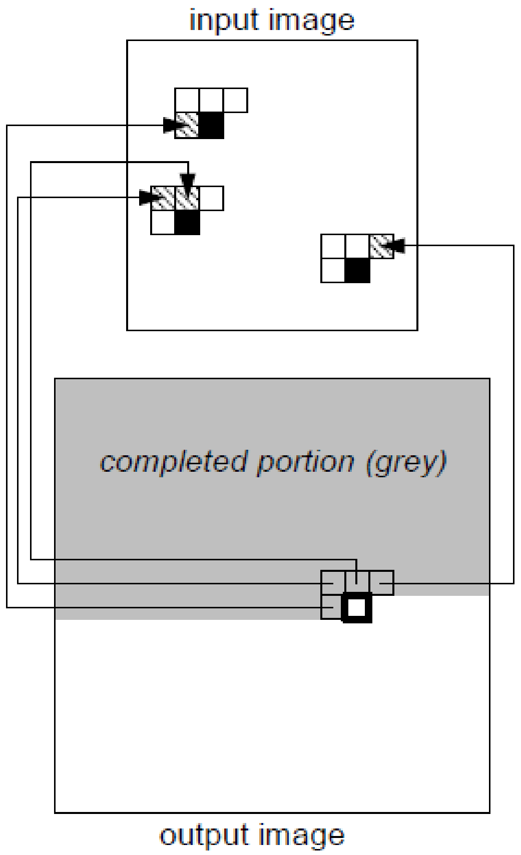

Figure 7: Each pixel in the current L-shaped 3x3 neighbourhood in the synthesis (output) image generates a candidate pixel according to its original position in the exemplar (input) image. Figure Source: Ashikhmin [1].

Lefebvre and Hoppe [22] express this in an equation:

$$S_l[p] := C^l_{i_{min}}\left(S_l[p+\Delta_{min}] - h_l\Delta_{min}\right), where$$

$$i_{min}, \Delta_{min} = argmin_{i\in 1\ldots k}||N_{S_l}(p) - N_{E_l}\left(C^l_{i_{min}}\left(S_l[p+\Delta] - h_l\Delta_{min}\right)\right)||\varphi(i)$$

$$\Delta \in \left\{\begin{pmatrix}-1\\-1\end{pmatrix}, \begin{pmatrix}-1\\0\end{pmatrix}, \ldots, \begin{pmatrix}+1\\+1\end{pmatrix}\right\}$$

$$inwhich\varphi(i) = \begin{Bmatrix} 1, i = 1 \\ 1 + \mathbb{k}, i > 1 \end{Bmatrix}$$

4.3.4 Tiling. The tiling algorithm ensures that the texture synthesis method produces terrain without artefacts and that each patch of generated terrain corresponds with the patches around it. To achieve this, we utilise an adjacency map. The adjacency map consists of six 8x8 grids representing each side of the cubic sphere. A list of adjacent faces are stored in each grid position. In addition, adjacent sides of the cubic sphere are also taken into account. The edges of patches identified as adjacent to a particular patch are used during terrain generation by the texture synthesis method. The tiling interface works with the globe interface to display the generated terrain.

5 EVALUATIONS

We reiterate the research questions as the results found in the evaluations aim to answer these questions:

(1) Does a texture synthesis method allow for the realistic generation of Earth-like planets?
(2) Does a mesh cube-based globe interface enable the illustration of Earth-like planets directly onto the globe without distortion?

We evaluate the globe interface for user control and regarding research question 2. The texture synthesis method is evaluated based on realism, efficiency, and scalability. The hypotheses for the evaluations are as follows:

- User Control: The mesh cube-based globe interface is user-friendly and displays the generated terrain with minimal visual distortion.
- Realism: The texture synthesis method produces generated terrain that is recognisable and visually coherent, with a LPIPS score of less than 0.4.
- Efficiency: The texture synthesis method completed synthesis of a single 64x64 patch in less than 2 seconds and uses less than 2.5 KB.
- Scalability: The generated terrain has sufficient precision and scale for planetary terrain generation, trading off well between realism and efficiency.

We executed all evaluations four times to ensure accurate results while remaining an efficient evaluation. All methods were evaluated on an AMD Ryzen 7 5700U processor with Radeon Graphics (1.80 GHz) and 8 GB of RAM.

5.1 Globe Interface

5.1.1 Control. We evaluate the globe interface based on two criteria: the ease of user control over the interface and the degree to which its design minimises distortions. Firstly, we assess the user control element. As a user study was out of the scope of this paper due mainly to time constraints, we have opted for a qualitative discussion on the usability of the globe interface. We recognise the potential bias of discussing our system. However, given that the interface requires minimal user interaction, we can confidently evaluate the interface's usability. The second evaluation element for the globe interface is the degree to which the mesh cube-based globe interface can minimise distortions. We visually assess and compare the generated terrain with other implementations of planetary generation, including Borg's [4]. Distortions will be evident if they are present, so it will not be difficult to ascertain whether there are distortions or artefacts. In addition, we will use a metric called Learned Perceptual Image Patch Similarity (LPIPS) to compare the poles of the outputted terrain to the poles of the exemplar. LPIPS [46] is a metric used to find the similarity between two patches. Unlike traditional image similarity metrics like mean squared error (MSE), which measure differences in pixels' values, LPIPS is designed to measure how similar two images appear to a human. LPIPS accomplishes this by applying a deep neural network to extract and compare features rather than pixel values [46]. The evaluations will compare multiple different generated terrain poles to their exemplar poles. The LPIPS scores range from 0 to 1. The lower the LPIPS score, the higher the similarity between the exemplar and generated image. The higher the similarity between the poles, the less likely there are significant distortions or artefacts. We highlight two important caveats. Firstly, we design the texture synthesis method to create a terrain with characteristics similar to the exemplar but not the same. Therefore, an LPIPS score of close to zero is unlikely and undesirable. The texture synthesis method will likely effect the LPIPS score. Secondly, we acknowledge that the LPIPS metric does not directly assess the presence of distortions. However, given our limitations, we believe it can assess distortion.

5.2 Texture Synthesis

5.2.1 Realism. The realism of generated terrains is notoriously difficult to evaluate quantitatively due to the difficulty of capturing human characteristics when assessing terrain visually. Rajasekaran et al. [32] developed a quantitative metric called Perceived Terrain Realism Metrics (PTRM). These metrics derive from a normalised input set of Geomorphons for a DEM terrain. Geomorphons are mixtures of characteristics that describe geomorphic land-form features [32]. Although this metric has seen some success in evaluation, it is complex and well beyond the scope of this paper. Therefore, to evaluate realism for terrain generated by our texture synthesis method, we use the LPIPS metric [46] mentioned in section 5.1.1. We use LPIPS to evaluate the generated terrain compared to that of the exemplar. The caveat mentioned in section 5.1.1 still applies. This method offers a reasonable estimation of realism in our generated terrain. However, the evaluation depends on the exemplar's quality because it is the exemplar with which the generated terrain is being compared. In LPIPS, a lower value indicates a better similarity with the exemplar and likely a more realistic terrain. We will also qualitatively discuss the visual appearance of the generated terrain in our results, as this is standard practice in terrain generation. The terrain evaluation is limited to single patches, so the qualitative visual inspection assesses global cohesion and realism.

5.2.2 Efficiency. We quantitatively evaluate the texture synthesis method's efficiency, assessing time and space efficiency. Once we have assessed it, we will discuss it and compare it to other terrain generation methods. For time efficiency, we measure the performance of preprocessing and synthesising separately. An investigation of the optimal number of processes to be run for each patch size is also conducted. We evaluate memory usage for space efficiency during synthesis and after terrain generation.

5.2.3 Scalability. We evaluate the texture synthesis method's scalability based on extent and precision. The extent describes the real-life distance represented by the entire generated terrain. We assess this by multiplying the patch size by the number of patches on the globe. The precision refers to the real-life distance represented by each pixel. For our implementation, the precision and extent of the input data are as mentioned in section 4.1. We also evaluate the effect of the generated terrain's extent and precision on other evaluation criteria, such as realism and efficiency. Once we have assessed it, we will discuss it and compare it to other terrain generation methods.

6 RESULTS AND DISCUSSIONS

After evaluating the globe interface and the texture synthesis method, we obtained valuable results.

6.1 Control

The globe interface is user-friendly, containing only a few simple user options for generating and regenerating terrain. Due to our mesh cube approach, we minimised distortion at the poles, which was supported by low FPIPS [46] scores when we compared the poles. FPIPS scores ranging from 0.2 to 0.6 indicate a recognisable degree of similarity to the exemplar and a lack of distortion. Average LPIPS score is 0.4, indicating less distortion at the poles. User control of generated terrain is an avenue for future work and should be easily implementable onto the globe interface.

6.2 Realism

The texture synthesis method produced varied realism when evaluated by the LPIPS metric. As we tested on a patch-by-patch basis, the realism results for the texture synthesis method according to LPIPS were across a broad spectrum. LPIPS scores ranged from 0.3 to 0.7, with the average score of 0.5 indicating moderate realism. Upon visual qualitative inspection of the generated terrain, we found that the terrain resembled exemplar terrain but with decreased global accuracy. This decreased accuracy can be seen in mountainous regions where mountain ranges can stop drastically. This is likely due to the absence of a global cohesion constraint between patches during texture synthesis. Alternatives to the LPIPS metric will improve the realism evaluation in generated terrains. However, given time constraints, the LPIPS metric can assess realism reasonably. Elevation data did not successfully merge with globe interface because the *TIF* files the DEMs are stored in do not lend themselves to texture synthesis.

6.3 Efficiency

We evaluated our texture synthesis method's time and space efficiency with promising results. The improved space efficiency created by procedures like the PCA projection allowed for a relatively small memory requirement for our texture synthesis method. The main memory requirement is storing the generated terrain, which is approximately 500KB for the 384 64x64 resolution patches. Regarding speed efficiency, we have developed a moderately high performance texture synthesis method due to the parallelism and use of Ashikhmin's [1] texture synthesis algorithm. More detail regarding time and space efficiency can be found in Table 1. Synthesis times per patch are 1-3 seconds, resulting in synthesis times across the planet of 2-4 minutes for a 96-faced cube and about 12 minutes for the 384-faced cube. The preprocessing time is significantly longer as it formulates neighbourhoods for all the pixels in the exemplar. This can result in significant preprocessing time when extended to the whole planet. Preprocessing takes 45-55 minutes on a 64x64 exemplar of 384 patches and 10-13 minutes on 96 patches. Parallel execution improved time efficiency with six processes optimal for a patch size of 64x64 and ten processes optimal for a patch size of 128x128. This increase is due to the increased processing needed as a result of the larger dimensions and the decrease in overhead relative to that workload. In Table 1, the entries with "Run Synthesis" at "NO" illustrates the efficiency of the globe interface independent of the texture synthesis algorithm.

6.4 Scalability

The scalability of our texture synthesis allows for consistently high precision and extent. We evaluated the texture synthesis method with varying extents and precisions. Although increasing extent and precision came with decreased efficiency, the texture synthesis handled different extents and precisions well. We ultimately found that the best results were at a patch size of 64x64 with a precision of 60 arc-seconds or 2-kilometre spacing at the equator. The texture synthesis method struggles to manage efficiency with an extent larger than 512x512 per patch and a precision of 500-metre spacing at the equator.

7 CONCLUSIONS

We found that texture synthesis methods can produce realistic terrain on a planetary scale, however, elevation data was unable to be incorporated. However, further development in evaluating realism in generated terrain could improve our understanding of the validity of these results. Our globe interface is user-friendly and significantly reduces distortions at the poles by utilising a mesh cube structure. In conclusion, we provide encouraging work for texture synthesis methods in terrain generation and find that further exploration into texture synthesis as a method for planetary generation is valid.

8 LIMITATIONS AND FUTURE WORK

There have been difficulties in accomplishing this work. Understanding our limitations will help us assess the overall relevance of our results. The main limitation of any example-based terrain generation technique is data availability. At a planetary scale, we only have data on a single planet's terrain, Earth, which must be considered as results are highly dependent on the exemplars. Although texture synthesis handles this issue effectively, the effect of the uniformity of the input data should be considered. Future additions to incorporate elevation data effectively should be explored. Time constraints also prevented us from exploring alternate evaluation procedures that may have yielded more accurate results. Time constraints also meant that the full scope of the texture synthesis and globe interface were not explored, but this allows for avenues for future work. Future work may include applying Lefebvre and Hoppe's appearance-space texture synthesis method [23], as this would significantly increase realism. The globe interface can be extended to include user controls for the specific design of planets. Finally, future work evaluating the realism of terrain will be vital in furthering this field. Rajasekaran et al.'s [32] PTRM takes the first step in this direction.

ACKNOWLEDGEMENTS

We are grateful to our supervisor, Professor James Gain, and our second reader, Professor Josiah Chavula, for their invaluable guidance and feedback throughout the project.

REFERENCES

[1] Michael Ashikhmin. 2001. Synthesizing natural textures. In *Proceedings of the 2001 symposium on Interactive 3D graphics*. 217–226.
[2] Bedřich Beneš, Václav Těšínsk'y, Jan Hornyš, and Sanjiv K Bhatia. 2006. Hydraulic erosion. *Computer Animation and Virtual Worlds* 17, 2 (2006), 99–108.
[3] Blender Online Community. 2023. *Blender - a 3D modelling and rendering package*. Blender Foundation. http://www.blender.org
[4] Oliver Borg. 2023. Planet Authoring with Generative AI. University of Cape Town. https://projects.cs.uct.ac.za/honsproj/cgi-bin/view/2023/borg_brann_hitge.zip/assets/Papers/PlanetAI.pdf
[5] Guillaume Cordonnier, Jean Braun, Marie-Paule Cani, Bedrich Benes, Eric Galin, Adrien Peytavie, and Éric Guérin. 2016. Large scale terrain generation from tectonic uplift and fluvial erosion. In *Computer Graphics Forum*, Vol. 35. Wiley Online Library, 165–175.
[6] Guillaume Cordonnier, Marie-Paule Cani, Bedrich Benes, Jean Braun, and Eric Galin. 2017. Sculpting mountains: Interactive terrain modeling based on subsurface geology. *IEEE transactions on visualization and computer graphics* 24, 5 (2017), 1756–1769.
[7] Yann Cortial, Adrien Peytavie, Éric Galin, and Éric Guérin. 2020. Real-time hyper-amplification of planets. *The Visual Computer* 36, 10 (2020), 2273–2284.
[8] Y. Cortial, A. Peytavie, E. Galin, and E. Guérin. 2019. Procedural Tectonic Planets. *Computer Graphics Forum* 38, 2 (2019), 1–11. https://doi.org/10.1111/cgf.13614 arXiv:https://onlinelibrary.wiley.com/doi/pdf/10.1111/cgf.13614
[9] Leandro Cruz, Luiz Velho, Eric Galin, Adrien Peytavie, and Eric Guérin. 2015. Patch-based terrain synthesis. In *International Conference on Computer Graphics Theory and Applications*. 6–pages.
[10] AR Dixon, GH Kirby, and Derek PM Wills. 1994. A data structure for artificial terrain generation. In *Computer Graphics Forum*, Vol. 13. Wiley Online Library, 37–48.
[11] Ricardo BD d'Oliveira and Antonio L Apolinário Jr. 2018. Procedural Planetary Multi-resolution Terrain Generation for Games. *arXiv preprint arXiv:1803.04612* (2018).
[12] Rae A Earnshaw. 1985. *Fundamental algorithms for computer graphics: NATO advanced study institute directed by JE bresenham, RA Earnshaw, MLV Pitteway*. Springer.
[13] David S Ebert, F Kenton Musgrave, Darwyn Peachey, Ken Perlin, and Steve Worley. 2002. *Texturing and modeling: a procedural approach*. Elsevier.
[14] 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.
[15] 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.
[16] 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.
[17] Aaron Hertzmann, Charles E Jacobs, Nuria Oliver, Brian Curless, and David H Salesin. 2023. Image analogies. In *Seminal Graphics Papers: Pushing the Boundaries, Volume 2*. 557–570.
[18] Jonathan Ho, Ajay Jain, and Pieter Abbeel. 2020. Denoising diffusion probabilistic models. *Advances in neural information processing systems* 33 (2020), 6840–6851.
[19] P Isola, J Zhu, T Zhou, and A Efros. 2017. Image-to-Image Translation with Conditional Adversarial Networks. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. Hawaii, USA* 1125 (2017), 1134.

[20] Peter Krištof, Bedrich Beneš, Jaroslav Křivánek, and Ondrej Št'ava. 2009. Hydraulic erosion using smoothed particle hydrodynamics. In *Computer graphics forum*, Vol. 28. Wiley Online Library, 219–228.

[21] Vivek Kwatra, Arno Schödl, Irfan Essa, Greg Turk, and Aaron Bobick. 2003. Graphcut textures: Image and video synthesis using graph cuts. *Acm transactions on graphics (tog)* 22, 3 (2003), 277–286.

[22] Sylvain Lefebvre and Hugues Hoppe. 2005. Parallel controllable texture synthesis. In *ACM SIGGRAPH 2005 Papers*. 777–786.

[23] Sylvain Lefebvre and Hugues Hoppe. 2006. Appearance-space texture synthesis. *ACM Transactions on Graphics (TOG)* 25, 3 (2006), 541–548.

[24] 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*, Vol. 42. Wiley Online Library, e14941.

[25] Benoit B Mandelbrot and John W Van Ness. 1968. Fractional Brownian motions, fractional noises and applications. *SIAM review* 10, 4 (1968), 422–437.

[26] H Jay Melosh. 2011. *Planetary surface processes*. Vol. 13. Cambridge University Press.

[27] Elie Michel, Arnaud Emilien, and Marie-Paule Cani. 2015. Generation of folded terrains from simple vector maps. In *Eurographics 2015 short paper proceedings*. The Eurographics Association, 4.

[28] 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.

[29] Lawrence L O'Boyle. 2018. Agent Based Terrain Generator: Cruthú. Master's Thesis. Grand Valley State University, United States of America.

[30] University of Cape Town. 2011. Intellectual Property Policy. https://uct.ac.za/sites/default/files/media/documents/uct_ac_za/190/Policy_Intellectual_Property_2011.pdf

[31] Patrick Pérez, Michel Gangnet, and Andrew Blake. 2023. Poisson image editing. In *Seminal Graphics Papers: Pushing the Boundaries, Volume 2*. 577–582.

[32] Suren Deepak Rajasekaran, Hao Kang, Martin Čadík, Eric Galin, Eric Guérin, Adrien Peytavie, Pavel Slavík, and Bedrich Benes. 2022. PTRM: Perceived terrain realism metric. *ACM Transactions on Applied Perceptions (TAP)* 19, 2 (2022), 1–22.

[33] Donald Shepard. 1968. A two-dimensional interpolation function for irregularly-spaced data. In *Proceedings of the 1968 23rd ACM national conference*. 517–524.

[34] Reto Stöckli, Eric Vermote, Nazmi Saleous, Robert Simmon, and David Herring. 2005. The Blue Marble Next Generation-A true color earth dataset including seasonal dynamics from MODIS. *Published by the NASA Earth Observatory* (2005).

[35] Flora Ponjou Tasse, James Gain, and Patrick Marais. 2012. Enhanced texture-based terrain synthesis on graphics hardware. In *Computer Graphics Forum*, Vol. 31. Wiley Online Library, 1959–1972.

[36] Xin Tong, Jingdan Zhang, Ligang Liu, Xi Wang, Baining Guo, and Heung-Yeung Shum. 2002. Synthesis of bidirectional texture functions on arbitrary surfaces. *ACM Transactions on Graphics (ToG)* 21, 3 (2002), 665–672.

[37] Filip Tripkovic. 2023. Agent Based Terrain Generator: Cruthú. Master's Thesis. Malmö University, Sweden.

[38] US Geological Survey. 2023. USGS: Science for a Changing World. https://www.usgs.gov. Accessed: 2024-09-06.

[39] Guido Van Rossum and Fred L. Drake. 2009. *Python 3 Reference Manual*. CreateSpace, Scotts Valley, CA.

[40] EF Vermote, N El Saleous, CO Justice, YJ Kaufman, JL Privette, L Remer, Jean-Claude Roger, and D Tanre. 1997. Atmospheric correction of visible to middle-infrared EOS-MODIS data over land surfaces: Background, operational algorithm and validation. *Journal of Geophysical Research: Atmospheres* 102, D14 (1997), 17131–17141.

[41] Jacopo Villa, J Mcmahon, and I Nesnas. 2023. Image Rendering and Terrain Generation of Planetary Surfaces Using Source-Available Tools. In *Proceedings of the 46th Annual AAS Guidance, Navigation & Control Conference, Breckenridge, CO, USA*. 1–24.

[42] Ryan J. Vitacion and Li Liu. 2019. Procedural Generation of 3D Planetary-Scale Terrains. In *2019 IEEE International Conference on Space Mission Challenges for Information Technology (SMC-IT)*. 70–77. https://doi.org/10.1109/SMC-IT.2019.00014

[43] Li-Yi Wei and Marc Levoy. 2000. Fast texture synthesis using tree-structured vector quantization. In *Proceedings of the 27th annual conference on Computer graphics and interactive techniques*. 479–488.

[44] Ling Yang, Zhilong Zhang, Yang Song, Shenda Hong, Runsheng Xu, Yue Zhao, Wentao Zhang, Bin Cui, and Ming-Hsuan Yang. 2023. Diffusion models: A comprehensive survey of methods and applications. *Comput. Surveys* 56, 4 (2023), 1–39.

[45] Steve Zelinka and Michael Garland. 2002. Towards Real-Time Texture Synthesis with the Jump Map. *Rendering Techniques* 2002 (2002), 13th.

[46] Richard Zhang, Phillip Isola, Alexei A Efros, Eli Shechtman, and Oliver Wang. 2018. The unreasonable effectiveness of deep features as a perceptual metric. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 586–595.

[47] 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.

A SUPPLEMENTARY INFORMATION

Table 1: Efficiency Table

Run Synthesis	Patch Size	Number of Patches	Average Synthesis Time	Average Patch Storage Size	Average Total Time
YES	64x64	384	2.01s	1.5KB	1hr 4min 19s
NO	64x64	384	N/A	1.5KB	2 min 33s
YES	64x64	96	2.01s	1.5KB	19min 13s
NO	64x64	96	N/A	1.5KB	2.1s