Terrain generation using Neural Transfer Style and Fractal Brownian Motion noise with constraints

Rodrigo Andre Cayro Cuadros
and Cláudio Fabiano Motta Toledo
Institute of Mathematics and Computer Sciences (ICMC)
Universidade de São Paulo (USP), São Carlos - Brasil
Email: rodrigo.cayro@usp.br and claudio@icmc.usp.br

Leonardo Tortoro Pereira
Instituto de Geociências e Ciências Exatas (IGCE)
Universidade Estadual Paulista (Unesp), Rio Claro - Brasil
Email: leonardo.t.pereira@unesp.br

***Abstract*—In the gaming industry, recent technological and economic growth has made it one of the most lucrative markets, driving demand for solutions that reduce costs and complexity in development. Procedural Content Generation (PCG) has emerged as a promising resource, particularly in terrain generation, which remains technically challenging. Existing approaches can be grouped into three categories: AI-based methods that provide realistic details, geometric techniques that enable scalability, and rule-based methods that enforce structural constraints. However, issues such as limited variety, quality, and customization remain. To address these challenges, we propose a system that integrates a *Transfer Style Network* with *Fractal Brownian Motion* (FBM) noise under user-defined constraints. This hybrid design generates terrains that are realistic, customizable, and semantically rich. Results show that our method preserves user constraints while producing diverse terrains similar to real-world samples, offering developers a robust and efficient tool for accelerating terrain creation and enhancing variability in game environments.**

I. Introduction

The video game industry has seen rapid growth, surpassing USD 187.7 billion in 2023 and projected to exceed USD 200 billion by 2026 [1], [2]. This expansion has increased the demand for techniques that accelerate development, particularly in complex content creation such as virtual terrains.

Generating large-scale, realistic terrains is challenging and requires expertise as well as significant computational effort. *Procedural Content Generation* (PCG) offers a solution by automating the creation of textures, maps, and landscapes [3]. Common approaches include: **AI-based methods**, which learn from real data to produce realistic structures; **noise-based techniques**, which generate heightmaps efficiently but lack semantic control; and **rule-based systems**, which ensure consistency but risk repetitiveness [4].

While games like *Minecraft* and *No Man's Sky* use these methods to create vast environments, challenges remain in achieving both diversity and coherence. Randomness can produce fragmented terrains, while over-control may yield repetitive results. Moreover, purely stochastic methods often miss realistic features such as rivers or biome diversity.

To address this, our proposed system combines real-world terrain data with stochastic noise generation, refined through neural networks. A genetic algorithm (GA) is employed to automatically tune the parameters of the style transfer stage, reducing manual trial-and-error and adapting the process to different geomorphological contexts. This approach aims to generate diverse, high-quality terrains with semantic structure, respecting user-defined constraints while ensuring consistency and efficiency. Evaluation focuses on visual quality, structural similarity, and adaptability to real terrains.

Results show that our method can produce terrains for multiple geomorphologies (ridge, valley, plain) with structural similarity above 0.79, final loss below 70, and strong adherence to constraints. This demonstrates its potential as an efficient tool for generating realistic, constraint-aware terrains with variety and speed.

II. Related Work

Procedural Terrain Generation (PTG) has evolved through various approaches, primarily falling into two categories: Geometric methods and AI-based.

A. Geometric Methods

Geometric or noise-based methods generate terrain by manipulating elevation, slope, and irregularity using mathematical models such as Perlin noise, fractals, and erosion algorithms [3], [5], [6]. These methods rely on pseudo-random functions and spatial rules to simulate realistic terrain forms. Although computationally efficient and widely used, they often lack semantic structure, realism, or fine control [7], [8].

The Sparse Construction Tree [9] enables terrain simplification by storing elevation features in a hierarchical structure, allowing further refinement. While useful for enhancing visual detail, it lacks support for landmarks, object placement, and consistent morphology.

Thorimbert et al. [10] propose a fast terrain generation technique based on fractal Brownian motion and polynomial interpolation. Their method generalizes terrain dimensions and gradients but lacks control over semantic features or terrain constraints. The AutoBiomes system [11] combines DEMs with climate simulation to generate multi-biome landscapes. Despite producing visually diverse terrains, it lacks support for rivers, erosion features, and points of interest.

A more flexible approach is proposed by Gasch et al. [12], where Perlin noise is combined with user-defined constraints to control elevation, paths, and terrain zones. This method introduces semantic control while preserving randomness but

can produce discontinuities in large maps. Finally, Le [7] enhances basic Perlin noise with customizable parameters (e.g., scale, persistence, octaves), enabling greater terrain variation. However, this technique still lacks semantic placement, object integration, and support for real-world features.

In summary, geometric methods offer simplicity and speed but typically fall short in semantic richness and customization.

B. AI-Based Methods

Artificial Intelligence (AI) techniques are widely used in terrain generation due to their ability to extract patterns from geospatial data, satellite imagery, and elevation maps [4]. These approaches can be broadly classified into two main types: evolutionary methods and neural networks.

1) Evolutionary Methods: Evolutionary algorithms aim to generate terrains quickly using stochastic operations such as mutation or recombination. Systems like the Auto Terrain Generation System (ATGS) [13] allow users to select preferred maps from a pool of random options, refining them iteratively. Genetic programming (GP), as explored in [14], encodes generation logic into tree structures but lacks diversity and control over real-world features.

Other works use hybrid approaches, combining genetic algorithms with 2D maps to evolve terrain features like elevation and distribution [15]. While promising, they struggle with terrain continuity and object placement. Cellular automata have been applied to generate 2D dungeons [16] but offer limited spatial complexity and are unfit for large 3D environments.

2) Neural Networks: Neural network-based techniques generate terrain procedurally from input data, often using *Deep Learning* (DL) approaches such as GANs and CNNs [4], [17]. These models process 2D satellite images or sketches to produce 3D elevation maps.

In the work of A. Wulff-Jensen [18], a DC-GAN model is used within Unity to generate high-resolution terrains. Similarly, Panagiotou et al. [17] convert satellite RGB images into digital elevation models (DEMs), though the method lacks semantic elements like landmarks or objects. Other GAN-based systems allow user-drawn inputs to guide terrain features like in G. Voulgaris work [19], but regularly trade detail for speed and lack support for infinite terrain generation.

Advanced methods such as StyleTerrain [20] aim to disentangle terrain features and amplify their structure, yet suffer from edge discontinuities and limited customization. In the work of F. Merizzi [21] uses neural style transfer to combine real terrain features with procedural noise, producing morphologically rich terrains at a high computational cost and limited generality.

Overall, while AI-based methods offer high visual fidelity and adaptability, they often struggle with semantic control, performance, or detailed customization.

III. Overview of the Hybrid Method

Prior work in PTG has explored different strategies to balance realism and controllability. Merizzi et al. [21] introduced a neural approach that combines features from real-world terrains with synthetic generation, achieving high visual fidelity but offering limited user control. Gasch et al. [12] proposed a constraint-based noise generation technique, which allows explicit control but lacks the expressive detail of AI-based models. Our work brings these ideas together in a unified pipeline that integrates style transfer, FBM noise, and user-defined constraints. In addition, a genetic algorithm (GA) is used as a preliminary step to optimize the parameters of the style transfer process, helping to identify suitable content and style weights before the pipeline is executed. This design ensures both realism and flexibility, while also reducing manual tuning and highlighting the originality of the approach.

Our system addresses key limitations found in existing methods. To overcome the absence of landmarks and user-defined constraints, we apply constrained FBM noise to generate paths and structured features. To enhance variability and realism, we extract morphological features from selected real terrains and combine them with adjustable noise parameters. The GA contributes by automatically searching for effective parameter configurations, using SSIM as part of the fitness function. Although not depicted in the main pipeline diagram, this optimization step complements the process by providing well-adapted parameters, allowing neural networks to enhance realism and adapt to different geomorphological contexts while smoothing transitions and mitigating discontinuities inherent in noise-based methods.

IV. Methodology

The overall pipeline is illustrated in Figure 1. It begins with constraint-based FBM noise to define paths and structural features, followed by the extraction of morphological patterns from real terrains. These are then integrated through neural style transfer, and finally refined with smoothing operations to reduce discontinuities. This sequence allows the system to balance controllability with morphological realism.

The system integrates optional user inputs, such as paths or shapes, with procedurally generated terrains and refines them through a style transfer neural network. This pipeline enables the generation of 3D terrains that are realistic, visually diverse, and customizable, while maintaining computational feasibility. The model is organized into four stages.

A. Part 1: Input Collection

The first stage receives two types of inputs. The first is an optional user-defined map provided through an on-screen interface. Users may draw paths, boundaries, or custom regions to define semantic structures such as roads, flat zones, or restricted elevation zones. This input offers high-level control over the terrain layout but can also be omitted for fully procedural generation, as discussed in [12].

The second input is a procedurally generated heightmap based on Fractal Brownian Motion (fBm) noise [22]. By adjusting parameters such as frequency, amplitude, lacunarity, turbulence, and the number of octaves, this technique produces complex terrain patterns with varied and natural-looking formations.

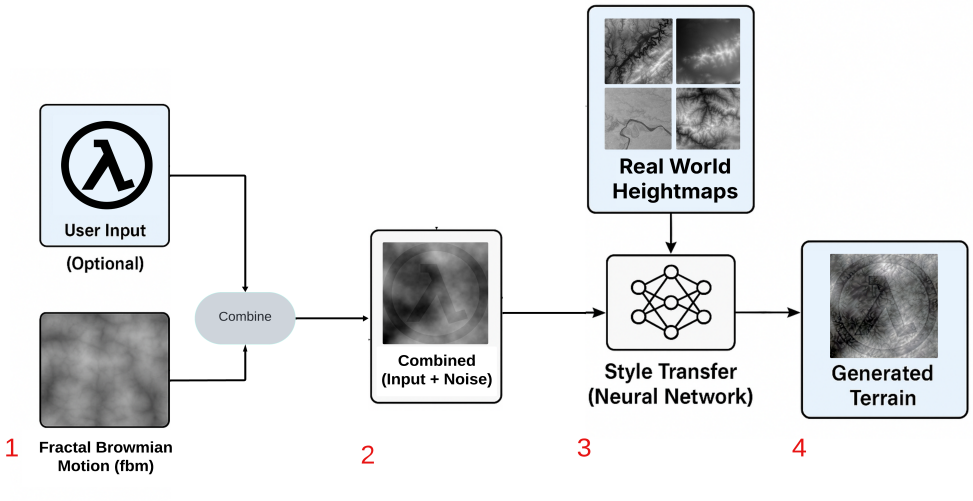


Fig. 1. Hybrid methodology for PTG using restrictions, fractal noise, and neural style transfer.

B. Part 2: Base Terrain Construction

In the second stage, both inputs are combined into a single base terrain following the concept of layered noise [12]. The constraint map is aligned with the fBm noise layer, embedding user-defined features within stochastic variations. This integration provides a compromise between semantic control and natural randomness. The resulting heightmap serves as the foundation for the subsequent stage, where realism is enhanced through neural style transfer.

C. Part 3: Style Transfer Using Real-World Heightmaps

Then, we use a style transfer network to enhance the generated terrain, following the method suggested by [21]. In this stage, we combine a real-world terrain image chosen by the user, serving as the style input, with a procedurally generated heightmap as the content.

To ensure better control and tuning of the neural network parameters, we limit the style inputs to three representative terrain types: *Valley*, *Plain*, and *Ridge*. Each of these requires specific parameter adjustments to achieve optimal results. The goal is to synthesize a final terrain that merges the structural layout of the content with the morphological characteristics of the selected real-world terrain type.

This is achieved using Neural Style Transfer (NST), implemented through a pre-trained VGG-19 convolutional neural network [23], which extracts both content and style features from the input images. Figures 2 and 3 illustrate these inputs and extracted content. The content image defines the overall shape of the terrain, while the style image contributes fine-grained textures such as river branches and erosion patterns.

T minimizes a weighted loss function that balances content and style, controlled by parameters α and β. To reduce manual trial-and-error in selecting these values, we employ a genetic algorithm (GA) as a preliminary step. Each individual in the GA encodes a candidate pair of content and style weights, and the fitness is defined by the Structural Similarity Index

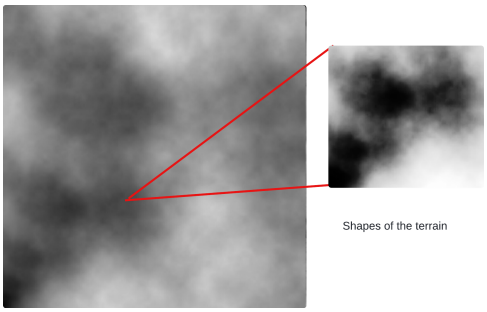


Fig. 2. Content image: a procedurally generated terrain. It provides the base structure for the final result.

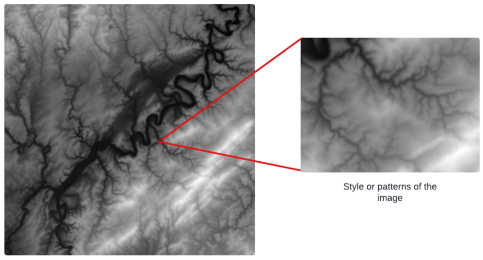


Fig. 3. Style image: a real-world terrain sample, showing natural features such as river branches and textures.

Measure (SSIM) between the generated terrain and the reference morphology. Through selection, crossover, and mutation, the GA converges toward parameter settings that maximize perceptual similarity while preserving structural constraints.

D. Part 4: Texturing and Final Terrain

This is the final stage of the proposed pipeline. Once the terrain has been processed through the Neural Style Transfer (NST) network, the next step involves post-processing the resulting heightmap. This output is loaded into a graphics

engine like Unity [24], where it is rendered by applying a color gradient based on elevation values.

V. Results and Experiments

We evaluated our algorithm with three user-defined inputs: a cross, a circle with a lambda inside, and an M-like curve. Each of these inputs is applied, respectively, to different real-world terrains previously defined in our dataset: ridge, plain, and valley. The inputs and their corresponding outcomes are shown in Figure 4.

The present evaluation highlights the ability of the method to generate coherent terrains that adapt to user-defined constraints. At this stage, the experiments focus on qualitative results and structural similarity metrics (SSIM), which already provide evidence of the method's ability to preserve terrain morphology under user-defined constraints. Since this is a work-in-progress contribution, broader comparisons with alternative methods and the inclusion of further performance measures will be carried out in subsequent stages of the research. Expanding these analyses will allow a more comprehensive positioning of our approach within the state of the art.

a) Case 1 — Ridge terrain.: The first input corresponds to a user-defined constraint that highlights the letter "X", as shown in Figure 4a. This is combined with the Ridge terrain (Figure 4b), and the final result is presented in Figure 4c. Ridge terrains are characterized by elongated elevated landforms with steep sides, often formed by erosion or tectonic activity. The distinctive crest lines and sharp relief patterns of the Ridge terrain have been successfully transferred onto the noise-based content image, preserving the original "X" constraint.

b) Case 2 — Valley terrain.: The second input consists of a shaped-based input, in this case based on the form of the lambda symbol (Figure 4d). This is combined with the Valley terrain (Figure 4e), producing the result shown in Figure 4f. Valley terrains are typically characterized by elongated low areas between hills or mountains, typically containing rivers or streams. The result clearly incorporates these features, with smooth slopes and channel-like depressions appearing within the constrained shape.

c) Case 3 — Plain terrain.: The third input corresponds to a path-based input, drawn manually on a blank canvas to define a specific trajectory (Figure 4g). This is combined with the plain terrain (Figure 4h), and the final result is shown in Figure 4i. Plain terrains are generally flat or gently rolling areas with minimal elevation change. The style transfer effectively reproduces this morphology, resulting in a smooth height distribution along the constrained path while maintaining the original layout of the input.

In all cases, the neural style transfer process successfully integrates the morphological characteristics of the selected real terrain into the user-defined input. This demonstrates the flexibility and adaptability of our approach.

To complement the qualitative evaluation, we render the final heightmaps in a 3D environment (Figure 5). This post-processing step was implemented in Unity, applying a color gradient from blue to white based on elevation values, where blue tones represent lower altitudes and white indicates the highest peaks. The rendering provides a realistic visual representation of the terrain, enabling an intuitive assessment of its morphology and improving the understanding of how the proposed method performs when integrated into real-time visualization or game engines.

A. Evaluation Metrics

As this is a work-in-progress contribution, the current evaluation relies on SSIM and optimization loss, which already provide meaningful insights into the quality of the generated terrains. In subsequent stages of the research, the evaluation will be extended to include qualitative visual assessments of strokes and morphological details, computational cost analysis in terms of efficiency and processing time, and diversity studies by generating multiple variations under identical constraints.

Among the current quantitative metrics, we employ the Structural Similarity Index Measure (SSIM) to compare the generated heightmap with the corresponding real-world terrain used as style. SSIM is particularly functional in this context, as it captures structural correspondence and perceptual similarity, providing an objective measure of how closely the synthetic terrain preserves the key spatial patterns of the real sample.

The final optimization loss reported in Table I is obtained from the total loss function used during neural style transfer, which combines three terms: the *content loss* (Equation 1), the *style loss* (Equation 2), and the *total variation loss* (Equation 3). The content loss measures the difference between high-level feature maps extracted from the content image (the noise-based heightmap) and the generated image, specifically from the `conv_5_2` layer of VGG-19, capturing the large-scale structure of the terrain. The style loss, computed from Gram matrices of selected convolutional layers, quantifies differences in fine-grained patterns and textures between the style image (real terrain) and the generated output. Finally, the total variation loss acts as a regularizer, encouraging spatial smoothness and reducing artifacts in the output image. Where r is the noise-generated heightmap, t is the real terrain image, and o is the generated output at each iteration.

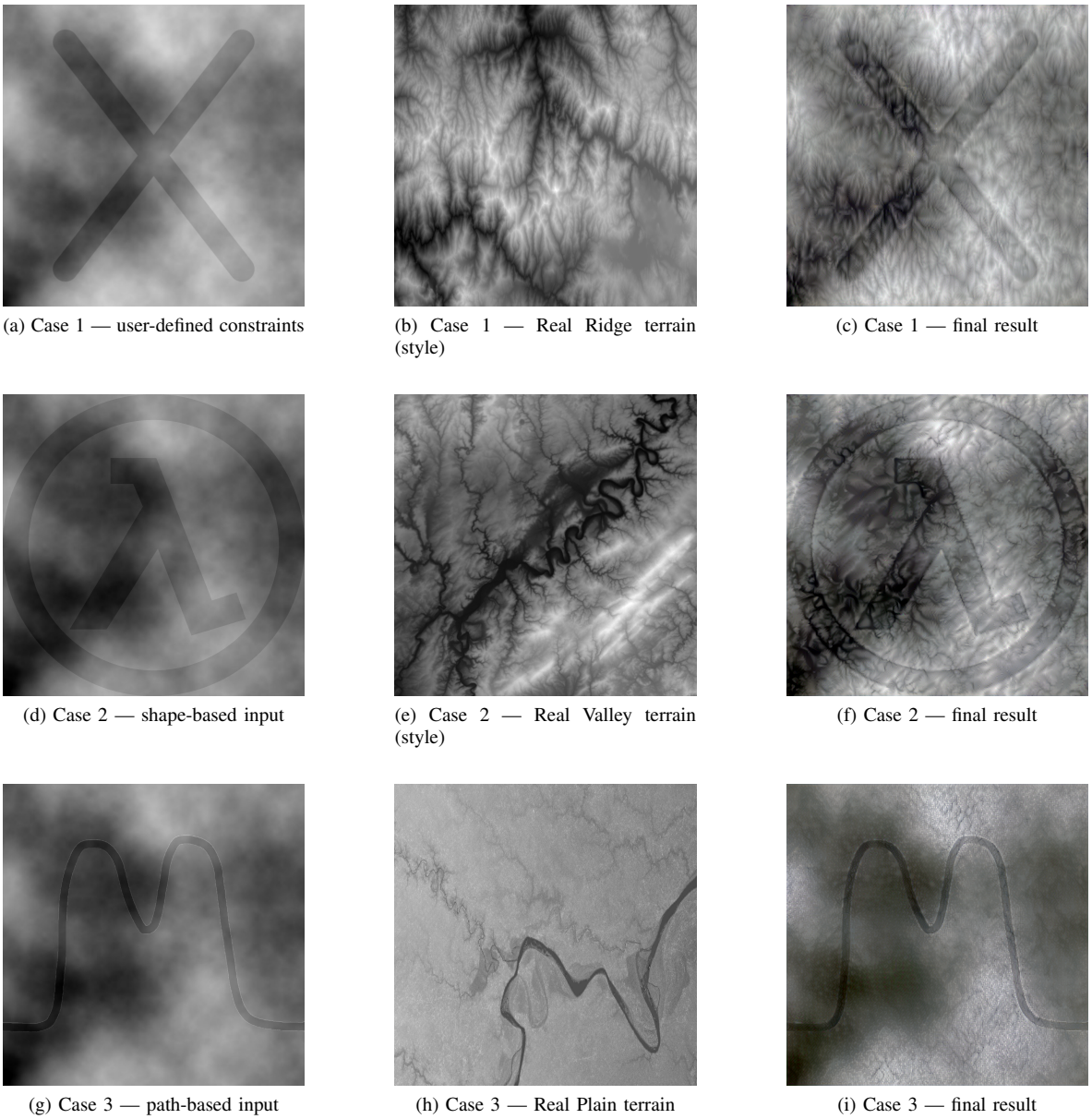
$$L_{\text{content}} = \sum_{l} \left(F1_{ik}^{l} - F2_{jk}^{l}\right)^2 \tag{1}$$

$$L_{\text{style}} = \frac{1}{4N_l^2 N_l^2} \sum_{ij} \left(G_{ij}^{1,l} - G_{jj}^{2,l}\right)^2 \tag{2}$$

$$L_{TV} = \sum_{i,j} |x_{i,j}| - x_{i+1,j} + |x_{i,j} - x_{i,j+1}| \tag{3}$$

The final optimization objective combines three components: content loss, style loss, and total variation loss, weighted respectively by α, β, and γ. These parameters are tuned individually for each terrain type to ensure optimal transfer quality.

$$\text{Loss} = \alpha L_{\text{content}}(r, o) + \beta L_{\text{style}}(t, o) + \gamma L_{TV}(o) \tag{4}$$



(a) Case 1 — user-defined constraints

(b) Case 1 — Real Ridge terrain (style)

(c) Case 1 — final result

(d) Case 2 — shape-based input

(e) Case 2 — Real Valley terrain (style)

(f) Case 2 — final result

(g) Case 3 — path-based input

(h) Case 3 — Real Plain terrain

(i) Case 3 — final result

Fig. 4. Figure 4. Results overview for the three experimental cases: ridge, plain, and valley terrains, with a 2000-iteration loop. Each row corresponds to one case: user-provided input, real-world terrain used as style, and the final result after neural style transfer.

The resulting final loss values (40 for Ridge, 70 for Valley, and 50 for Plain) are not intended to be minimized in absolute terms, but rather balanced to preserve user-defined constraints while effectively transferring morphological patterns from the style image. When interpreted alongside SSIM, these values confirm the trade-off between constraint preservation and style fidelity that is central to our method. In this context, higher SSIM values indicate a stronger preservation of structural patterns from the reference terrain, reflecting greater morphological fidelity. For example, the Plain case achieves the highest SSIM (0.881), showing excellent consistency with the real terrain despite a moderate loss value. Conversely, the Valley case reports the lowest SSIM (0.795), suggesting that while constraints are respected, some fine-grained details are harder to retain. This analysis reinforces that SSIM provides meaningful evidence of the effectiveness of our approach, with higher values denoting more successful morphology transfer.

TABLE I

SSIM MEASURES THE STRUCTURAL SIMILARITY BETWEEN THE GENERATED AND STYLE IMAGES. FINAL LOSS CORRESPONDS TO THE WEIGHTED SUM OF THE COMPONENTS DESCRIBED IN SECTION IV.

Case / Terrain	SSIM	Final Loss
Case 1 — Ridge	0.842	40
Case 2 — Valley	0.795	70
Case 3 — Plain	0.881	50

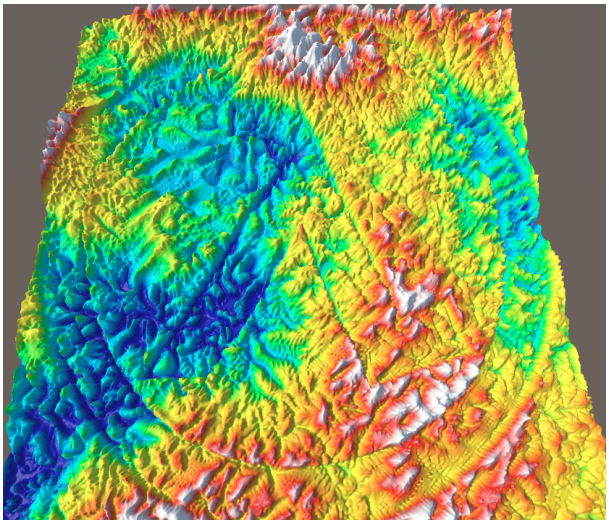


Fig. 5. 3D rendering of the generated terrain with the lambda-shaped constraint. A color gradient shows elevation, from blue (low) to red (high), highlighting the integration of natural features with the user-defined shape.

VI. Conclusion

This work presented a PTG method that integrates user-defined constraints with the morphological realism of neural style transfer. The approach combines the flexibility of design-based methods with the fidelity of AI-based models, generating realistic terrains that respect input constraints.

Results show visually coherent terrains across multiple geomorphologies, with high SSIM values and low loss scores, confirming both the effectiveness of style transfer and the stability of the optimization. A key contribution is the use of a genetic algorithm (GA) as an initial stage to automatically tune content and style weights, reducing manual effort and enabling consistent adaptation to different terrain types. This integration of constraint-based noise, neural refinement, and evolutionary optimization reinforces the originality of the approach.

The method accelerates terrain generation and supports procedural level creation while preserving constraints, with applications in game development and virtual simulations. Although not yet real-time due to computational costs, future optimizations may enable this capability. Current limitations include the absence of broader comparisons and benchmarks; future work will address these through additional metrics such as computational cost, qualitative evaluation, and diversity analysis.

References

[1] Newzoo, "2023 newzoo free global games market report," 2023, accessed on August 03, 2024. [Online]. Available: http://www.daelab.cn/wp-content/uploads/2023/09/2023_Newzoo_Free_Global_Games_Market_Report.pdf

[2] P. Rykała, "The growth of the gaming industry in the context of creative industries," *Biblioteka Regionalisty*, vol. 20, pp. 124–136, 2020. [Online]. Available: 124-136_Rykala_The_growth_of_the_gaming_industry_in_the_context_of_creative.pdf

[3] N. Shaker, J. Togelius, and M. J. Nelson, *Procedural content generation in games*. Springer, 2016.

[4] J. Liu, S. Snodgrass, A. Khalifa, S. Risi, G. N. Yannakakis, and J. Togelius, "Deep learning for procedural content generation," *Neural Computing and Applications*, vol. 33, no. 1, pp. 19–37, 2021. [Online]. Available: https://doi.org/10.1007/s00521-020-05383-8

[5] T. Archer, "Procedurally generating terrain," in *44th annual midwest instruction and computing symposium, Duluth*, 2011, pp. 378–393.

[6] T. Hyttinen, E. Mäkinen, and T. Poranen, "Terrain synthesis using noise by examples," in *Proceedings of the 21st International Academic Mindtrek Conference*, ser. AcademicMindtrek '17. New York, NY, USA: Association for Computing Machinery, 2017, p. 17–25. [Online]. Available: https://doi.org/10.1145/3131085.3131099

[7] T. Le, "Procedural terrain generation using perlin noise," Graduate Project, California State Polytechnic University, Pomona, 2023, scholarWorks.

[8] R. r. Spick and j. Walker, "Realistic and textured terrain generation using gans," in *Proceedings of the 16th ACM SIGGRAPH European Conference on Visual Media Production*, 2019, pp. 1–10.

[9] E. Guérin, J. Digne, E. Galin, and A. Peytavie, "Sparse representation of terrains for procedural modeling," *Computer Graphics Forum*, vol. 35, no. 2, pp. 177–187, 2016. [Online]. Available: https://onlinelibrary.wiley.com/doi/abs/10.1111/cgf.12821

[10] Y. Thorimbert, "Polynomial method for procedural terrain generation," *CoRR*, vol. abs/1610.03525, 2016. [Online]. Available: http://arxiv.org/abs/1610.03525

[11] R. Fischer, P. Dittmann, R. Weller, and G. Zachmann, "Autobiomes: procedural generation of multi-biome landscapes," *The Visual Computer*, vol. 36, no. 10, pp. 2263–2272, 2020. [Online]. Available: https://doi.org/10.1007/s00371-020-01920-7

[12] C. Gasch, M. Chover, I. Remolar, and C. Rebollo, "Procedural modelling of terrains with constraints," *Multimedia Tools and Applications*, vol. 79, no. 41, pp. 31 125–31 146, November 2020. [Online]. Available: https://doi.org/10.1007/s11042-020-09476-3

[13] P. Walsh and P. Gade, "Terrain generation using an interactive genetic algorithm," in *IEEE Congress on Evolutionary Computation*, 2010, pp. 1–7.

[14] M. Frade, F. F. de Vega, and C. Cotta, "Automatic evolution of programs for procedural generation of terrains for video games," *Soft Computing*, vol. 16, no. 11, pp. 1893–1914, nov 2012. [Online]. Available: https://doi.org/10.1007/s00500-012-0863-z

[15] I. Antoniuk and P. Rokita, "Procedural generation of adjustable terrain for application in computer games using 2d maps," in *Pattern Recognition and Machine Intelligence*, M. Kryszkiewicz, S. Bandyopadhyay, H. Rybinski, and S. K. Pal, Eds. Cham: Springer International Publishing, 2015, pp. 75–84.

[16] N. M. Husnul Habib Yahya, H. Fabroyir, D. Herumurti, I. Kuswardayan, and S. Arifiani, "Dungeon's room generation using cellular automata and poisson disk sampling in roguelike game," in *2021 13th International Conference on Information, Communication Technology and System (ICTS)*, 2021, pp. 29–34.

[17] E. Panagiotou and E. Charou, "Procedural 3d terrain generation using generative adversarial networks," *arXiv preprint arXiv:2010.06411*, vol. 2, 2020.

[18] A. Wulff-Jensen, N. N. Rant, T. N. Møller, and J. A. Billeskov, "Deep convolutional generative adversarial network for procedural 3d landscape generation based on dem," in *Interactivity, Game Creation, Design, Learning, and Innovation*, A. L. Brooks, E. Brooks, and N. Vidakis, Eds. Cham: Springer International Publishing, 2018, pp. 85–94.

[19] G. Voulgaris, I. Mademlis, and I. Pitas, "Procedural terrain generation using generative adversarial networks," in *2021 29th European Signal Processing Conference (EUSIPCO)*, 2021, pp. 686–690.

[20] Y.-L. Huang and X.-F. Yuan, "Styleterrain: A novel disentangled generative model for controllable high-quality procedural terrain generation," *Computers and Graphics*, vol. 116, pp. 373–382, 2023. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S009784932300225X

[21] F. Merizzi, "Procedural terrain generation with style transfer," 2024.

[22] B. B. Mandelbrot and J. W. Van Ness, "Fractional brownian motions, fractional noises and applications," *SIAM Review*, vol. 10, no. 4, pp. 422–437, 1968.

[23] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," 2015, accessed on July 9, 2024. [Online]. Available: https://arxiv.org/abs/1409.1556

[24] J. K. Haas, "A history of the unity game engine," 2014.