From Chaos to Continents: Voronoi-Based Procedural Terrain Generation with Hydrology and 3D Visualization

Batsambuu Batbold
Department of Mathematics, Statistics, and Computer Science, Macalester College, St. Paul, MN, USA

Lori Ziegelmeier
Department of Mathematics, Statistics, and Computer Science, Macalester College, St. Paul, MN, USA

Abstract

Procedural content generation often employs grid-based methods to create virtual environments. We present a pipeline that utilizes Voronoi diagrams and Lloyd's Relaxation to construct an irregular mesh for terrain generation. We implement a customizable "Land Anchor" system combined with Perlin noise to determine landmass shapes, distinct from standard radial distribution methods. Furthermore, we simulate hydrology using priority-flood routing on the Voronoi edges and assign biomes via a Gaussian-smoothed Whittaker classification. The full pipeline is exposed through an interactive application that enables real-time parameter tuning and terrain export, and resulting geometric data is extruded in Blender to produce a 3D terrain model.

2012 ACM Subject Classification Computing methodologies → Image-based rendering; Theory of computation → Computational geometry

Keywords and phrases Procedural Content Generation, Voronoi Diagrams, Lloyd's Relaxation, Perlin Noise, Blender

Digital Object Identifier 10.4230/LIPIcs.SoCG.2026.101

Category Media Exposition

Related Version *Blog Post*: `https://basabu1.github.io/Map-Generation/` [2]

Supplementary Material
Software: `https://github.com/BaSaBu1/Map-Generation` [4]
archived at `swh:1:dir:0ce321b8c3e39d9e7d85f1d6855e3c43cf34a476`
InteractiveResource (Web Application): `https://basabu1-map-generation-app-fut9qy.streamlit.app/` [3]

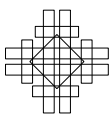
Funding *Lori Ziegelmeier*: Supported by the NSF award BCS-2318171.

1 Introduction

Procedural Content Generation (PCG) creates virtual environments algorithmically [15, 10]. While grid-based approaches are computationally efficient, they often produce rectilinear artifacts. Polygonal map generation based on Voronoi diagrams offers an alternative for constructing irregular meshes; our work is inspired by the framework detailed in [13].

Existing polygonal implementations to generate terrains, such as [13], commonly employ a radial island mask biasing land generation toward the map center and utilize simplified hydrology models. In this work, we extend this polygonal approach to support flexible landmass design as well as 3D visualization. Our approach introduces three primary modifications:

1. **Geometry Generation:** Instead of a radial mask, we implement a customizable "Land Anchor" system combined with Perlin noise, allowing for the generation of non-convex landmasses and archipelagos.



2. **Hydrology Simulation:** We use a priority-flood algorithm to simulate flow accumulation for river generation along the Voronoi edges.
3. **3D Visualization:** We extend the 2D graph data into a 3D pipeline, converting the mesh into heightmaps for rendering in Blender.

We discuss each step of our approach in the following sections.

2 Geometric Foundation

2.1 Voronoi Diagrams and Duality

To create an irregular mesh, we partition the plane using a *Voronoi diagram* [8]. Given a set of points S as sites, the Voronoi region V_p corresponding to a site $p \in S$ contains all points in the plane closer to p than to any other site in S.

We construct this mesh via its dual graph, the *Delaunay triangulation*. We utilize the `delaunator` library [12] to efficiently compute the triangulation. The circumcenters of the resulting Delaunay triangles serve as the vertices of the Voronoi polygons, while the edges between them define the adjacency relationships between cells.

2.2 Lloyd's Relaxation

Voronoi sites are generated uniformly at random, though naive sampling produces uneven distributions with high variance in cell area and edge length. To regularize the mesh, we apply *Lloyd's Relaxation* [11], which iteratively converges to a Centroidal Voronoi Tessellation [9]. This redistributes sites more evenly, producing cells that are relatively uniform in area while retaining irregular boundaries; see Figure 1.

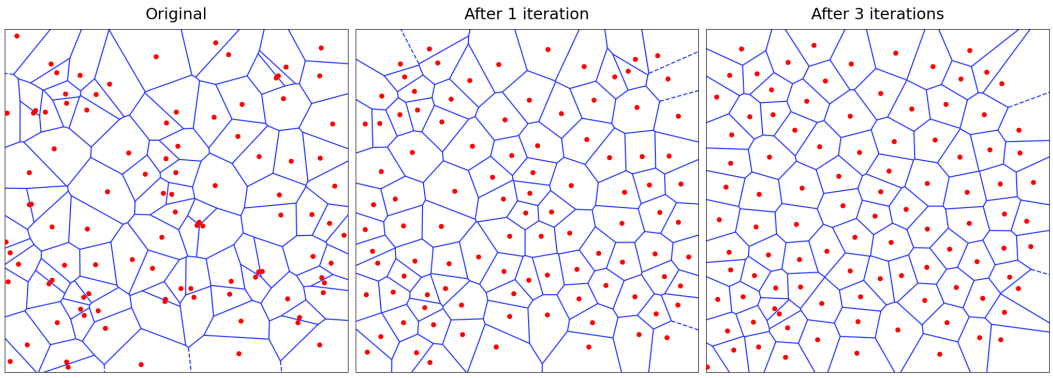


Figure 1 (Left) Voronoi mesh generated from random points. (Right) Uniform mesh resulting from 3 iterations of Lloyd's Relaxation.

3 Procedural Shaping

3.1 Elevation: Noise and Anchors

To generate elevation in our terrain, we use *Perlin noise* [14] implemented in [5], which provides spatially coherent values unlike uncorrelated sampling. We introduce a *Land Anchor* system to control terrain shape. Rather than a single radial mask for island formation, our method generalizes to N arbitrary anchor points. The final elevation E of a cell with Voronoi

site at (x, y) is defined as $E_{final} = \text{Perlin}(x, y) - (d_{min}/R)^2$ where d_{min} is the Euclidean distance to the nearest land anchor, and R is the map radius. This approach concentrates high elevation values around specific focal points while preserving the irregular, jagged nature of the noise along the coastlines; see Figure 2. Subsequently, a piecewise power curve is applied to the elevation data to compress low-lying areas into flat beaches and exponentially scale higher values into steep mountain peaks.

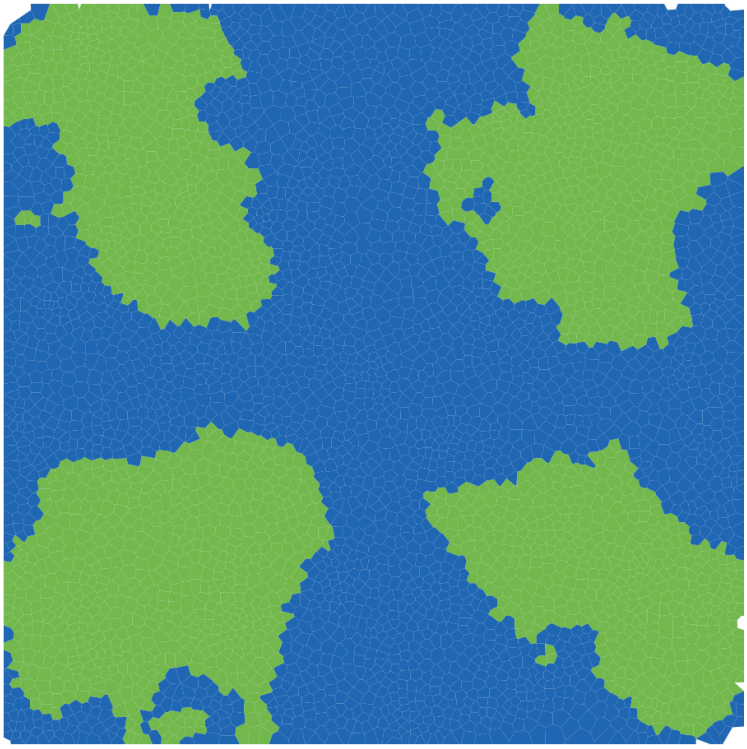


Figure 2 Non-radial archipelago layout generated using a four-land anchor configuration.

3.2 Biomes and Gaussian Smoothing

To texture the terrain, we simulate moisture using a secondary Perlin noise map modified by proximity to water. Cells are classified into biomes (e.g., Desert, Forest, Snow) using a *Whittaker Biome Diagram* [17], which maps $(Elevation, Moisture)$ pairs to terrain types.

Discrete biome classification can result in abrupt transitions between distinct zones; see Figure 3 (left). To mitigate this, we implement a smoothing pipeline:

1. Generate a 256×256 lookup table representing the Whittaker classification.
2. Apply SciPy's [16] *Gaussian Blur* ($\sigma = 8$) to the table to generate boundary gradients.
3. Assign cell colors using bilinear interpolation on this blurred table.

This results in gradient transition zones, such as a grassland buffer separating a forest from a beach; see Figure 3 (right).

3.3 Hydrology: Priority-Flood Routing

We simulate hydrology by utilizing the graph structure of the mesh, defining river channels along Voronoi edges. A *priority-flood algorithm* [1] is implemented to model flow dynamics:

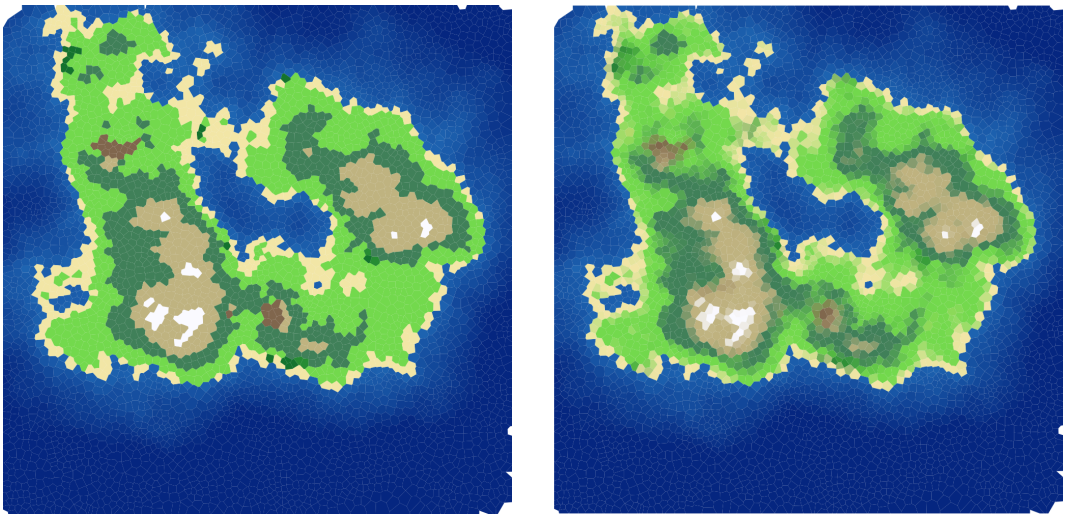


Figure 3 Biome distribution using rigid classification (left) and Gaussian-smoothed gradients between distinct zones (right).

1. **Flow Routing:** We construct a directed graph where each vertex connects to its lowest neighbor, creating a continuous path to the ocean.
2. **Flow Accumulation:** We traverse the graph from local maxima to the coast, aggregating flow values at each step.

The resulting accumulation values determine the width of the river segments, generating a river network structure; see Figure 4.

4 3D Visualization and Results

To visualize the generated terrain in three dimensions, we export the graph data using the Python Imaging Library [7]. We generate three primary texture maps:

- **Heightmap:** A 16-bit grayscale PNG representing the elevation data.
- **Colormap:** An RGB image representing the Gaussian-smoothed biome colors.
- **River Mask:** A binary mask isolating river pixels for shader application.

These textures are imported into Blender [6] to construct the 3D mesh. We apply a displacement modifier to a high-resolution plane to deform the geometry according to the heightmap. A custom shader utilizes the river mask to assign distinct material properties (specularity and roughness) to water surfaces. Figure 5 displays an example final output.

5 Conclusion

We present a pipeline for procedural terrain generation (see project webpage [2], interactive application [3] with example in Figure 6, and github repo [4]) that constructs a Voronoi mesh regularized via Lloyd's relaxation. Land shape was determined through a Land Anchor system with Perlin noise, and hydrology was simulated using a graph-based priority-flood algorithm. The mesh was exported to Blender for 3D rendering. The system produced plausible terrain, though the current Python implementation exhibited performance limitations at high polygon counts ($N > 50{,}000$). Future work includes computational optimization and atmospheric extensions (e.g., wind and temperature modeling) to support richer biome variation.

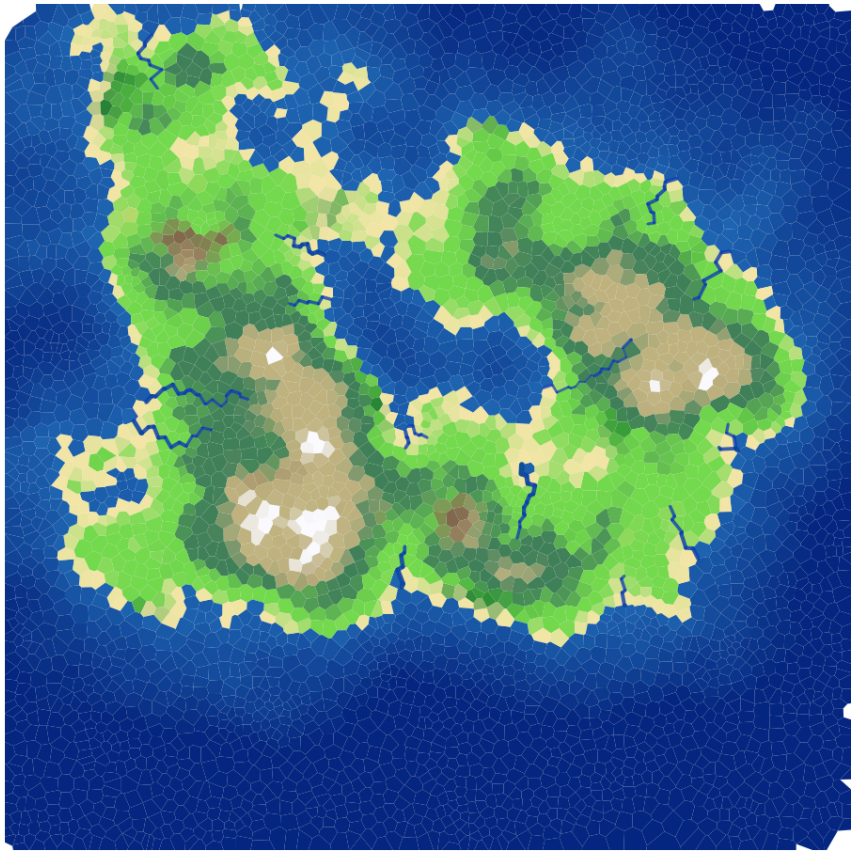


Figure 4 Hydrology simulation with river networks flowing along Voronoi edges.

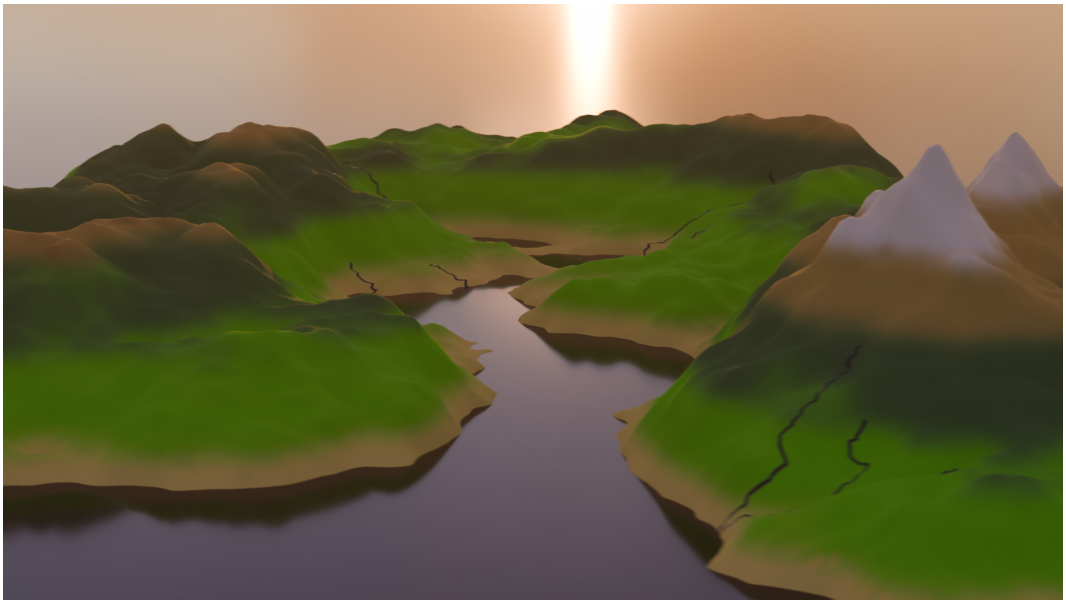


Figure 5 Example 3D terrain rendered in Blender.

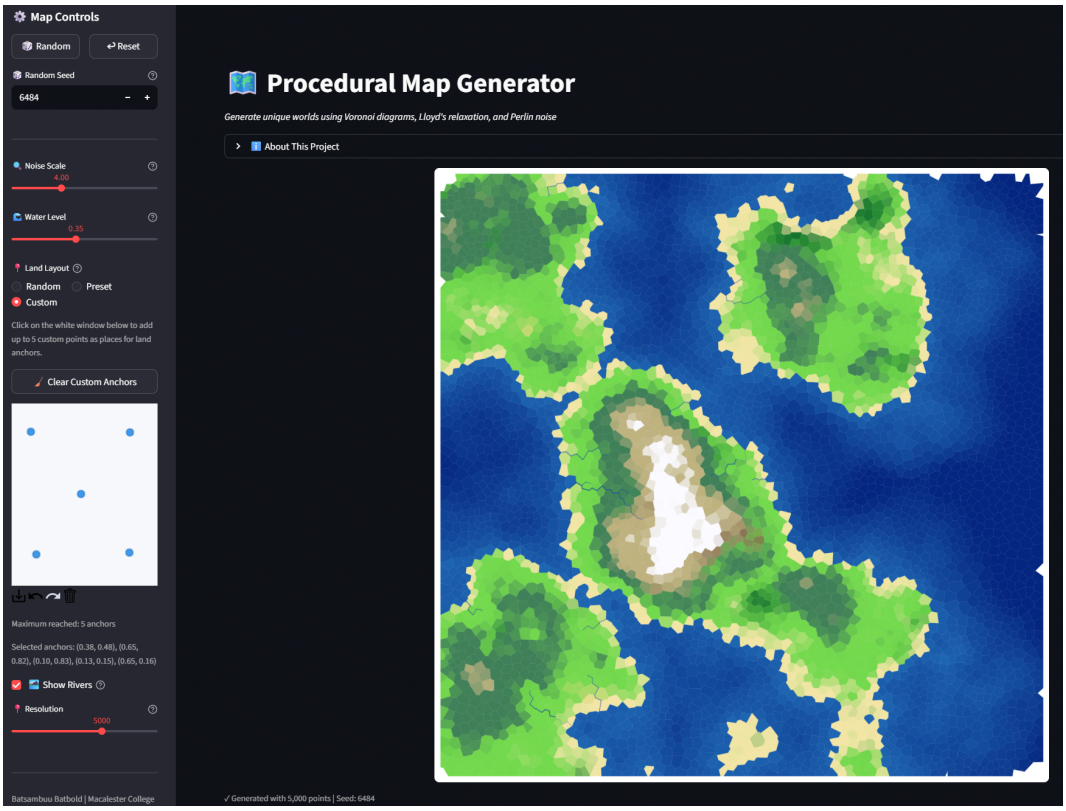


Figure 6 Screenshot of the Streamlit application [3], with sliders to specify parameters.

References

1 Richard Barnes, Clarence Lehman, and David Mulla. Priority-flood: An optimal depression-filling and watershed-labeling algorithm for digital elevation models. *Computers & Geosciences*, 62:117–127, 2014. doi:10.1016/j.cageo.2013.04.024.

2 Batsambuu Batbold. From chaos to continents: Procedural terrain generation. https://basabu1.github.io/Map-Generation/, 2025. Accessed: 2026-02-17.

3 Batsambuu Batbold. Map generation interactive app. https://basabu1-map-generation-app-fut9qy.streamlit.app/, 2025. Accessed: 2026-02-21.

4 Batsambuu Batbold. Map generation source code. https://github.com/BaSaBu1/Map-Generation, 2025. Accessed: 2026-02-21.

5 Alexandre Beyeler. Pnoise: A Pure Python Perlin Noise Generator. https://pypi.org/project/pnoise/, 2022. PyPI package.

6 Blender Online Community. *Blender - a 3D modelling and rendering package*. Blender Foundation, Blender Institute, Amsterdam, 2025. URL: http://www.blender.org.

7 Alex Clark. Pillow (pil fork) documentation, 2024. Python Imaging Library. URL: https://pillow.readthedocs.io/.

8 Satyan L Devadoss and Joseph O'Rourke. *Discrete and computational geometry*. Princeton university press, 2025.

9 Qiang Du, Vance Faber, and Max Gunzburger. Centroidal voronoi tessellations: Applications and algorithms. *SIAM Review*, 41(4):637–676, 1999. doi:10.1137/S0036144599352836.

10 Mark Hendrikx, Sebastiaan Meijer, Joeri Van Der Velden, and Alexandru Iosup. Procedural content generation for games: A survey. *ACM Transactions on Multimedia Computing, Communications, and Applications (TOMM)*, 9(1):1–22, 2013. doi:10.1145/2422956.2422957.

11 S. Lloyd. Least squares quantization in pcm. *IEEE Transactions on Information Theory*, 28(2):129–137, 1982. doi:10.1109/TIT.1982.1056489.

12 Mapbox. Delaunator: A fast library for delaunay triangulation of 2d points, 2024. Accessed: 2026-02-17. URL: https://github.com/mapbox/delaunator.

13 Amit Patel. Polygonal map generation for games, 2010. Red Blob Games. Accessed: 2026-02-17. URL: http://www-cs-students.stanford.edu/~amitp/game-programming/polygon-map-generation/.

14 Ken Perlin. An image synthesizer. *ACM Siggraph Computer Graphics*, 19(3):287–296, 1985. doi:10.1145/325334.325247.

15 Noor Shaker, Julian Togelius, and Mark J Nelson. *Procedural content generation in games*. Springer, 2016. doi:10.1007/978-3-319-42716-4.

16 Pauli Virtanen, Ralf Gommers, Travis E. Oliphant, Matt Haberland, Tyler Reddy, David Cournapeau, Evgeni Burovski, Pearu Peterson, Warren Weckesser, Jonathan Bright, Stéfan J. van der Walt, Matthew Brett, Joshua Wilson, K. Jarrod Millman, Nikolay Mayorov, Andrew R. J. Nelson, Eric Jones, Robert Kern, Eric Larson, CJ Carey, İlhan Polat, Yu Feng, Eric W. Moore, Jake VanderPlas, Denis Laxalde, Josef Perktold, Robert Cimrman, Ian Henriksen, E. A. Quintero, Charles R. Harris, Anne M. Archibald, A. H. Ribeiro, Fabian Pedregosa, Paul van Mulbregt, and SciPy 1.0 Contributors. Scipy 1.0: Fundamental algorithms for scientific computing in python. *Nature Methods*, 17(3):261–272, 2020.

17 Wyoming Biodiversity Information System. Whittaker biome diagram guide. https://gveg.wyobiodiversity.org/application/files/7916/4641/2117/Whittaker_Diagram_Guide.pdf, 2020. Accessed: 2026-02-16.