Final Project: Polygon Graph based Island Generation for "Sea of Thieves" like games

Temur Kholmatov, Innopolis University

October 2020

1 Introduction

1.1 Background

The ground reason to work on the island generation is the author's interest in the anime series "One Piece" where the main characters are pirates who pass through the fascinating journey seeking the hidden treasures of the Pirates King (Wikipedia contributors, 2020). There are several dozen games in the One Piece world. However, most of them are 3D fighting games and only a few are of the action-adventure type. In my opinion, it would be more interesting to play a sort of an open-world online game where players could travel from one island to another and there could be a diversity of different islands like in the anime. For instance, in the series, the crew had trips to quite extraordinary places like a sky island, an underwater island, an island on the back of a large elephant, etc. Later, searching for some similar games, I encountered the game "Sea of Thieves" with multiplayer mode and different beautiful islands to travel to (Wikipedia contributors, 2020). It turned out that the islands in this game are human-created content because there is a fixed number of such locations. This fact led to the idea of island generation for the project of the Procedural Content Generation course, even though without extraordinary sky and underwater islands.

The proposed method is based on the construction of Voronoi diagrams combined with Delaunay triangulation. This method helps to produce less number of separated lands compared to simple noise-based algorithms.

The source code of the project with description is available here: `https://github.com/temur-kh/polygonal-island-generation`.

1.2 Taxonomy

According to the taxonomy (Togelius et al., 2011), one may define the following distinctions of the method:

- Offline. The algorithm itself is quite complex and requires several steps to generate an island. Furthermore, real games usually require such objects to have a large resolution and scale. That is why such an asset should be generated during world creation.

- Necessary. In such games as "Sea of Thieves", an island is definitely one of the most important assets.
- Large Control. Honestly, this factor is quite arguable because it depends on the implementation. For instance, A. Patel (2010) in his online map generator provides a toolbar with lots of different control variables that affect the outlook of a map. Indeed, in a real online game for "One Piece", you would prefer to have the opportunity to change the materials of islands in order to create some extraordinary ones like those located on clouds. In this work, we focus on moving all parameters and variables into the control panel.
- Generic. The player does not affect the process of island generation.
- Stochastic. The method implies the usage of random points and noise generating functions.
- Constructive. The method creates a map in one iteration without any search-based approaches. However, one could combine the method with some evolutionary algorithms to improve the fitness of an object to some criteria.
- Automatic. A human designer is not involved in the process of island generation. However, human intervention could help in the decoration of the terrain with some special assets or in the process of evaluation (discussed in Section 3.1).

2 Method

The method is first proposed by A. Patel (2010) who used the graph structure to model maps by gameplay constraints like elevation, river flow and etc., with noise functions for coastline shapes, river, and tree placement. M. Dangschat and Weweler (2017) in their work make an improvement over the method using it for 3d map generation. The general pipeline of the proposed method is defined to be like this:

1. Generation of random points
2. Calculation of Voronoi and Delaunay polygons based on the points.
3. Apply Lloyd relaxation to make polygons less "clumsy" and makes them closer to quasi-randomness.
4. Generate the elevation map using Perlin noise maps and a falloff map
5. Now, the knowledge about the elevation and a moisture parameter can be used for the assignment of biomes.
6. Finally, we create a mesh and apply the texture to it.
7. In the end, we also evaluate the generated island according to some metrics (see Section 3.1).

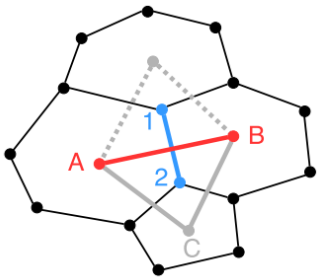

Figure 1: Graphs structure (Dangschat and Weweler, 2017). A and B are nodes of the Delaunay graph, whereas 1 and 2 are nodes of the Voronoi graph. The triangle ABC represents a Delaunay triangle.

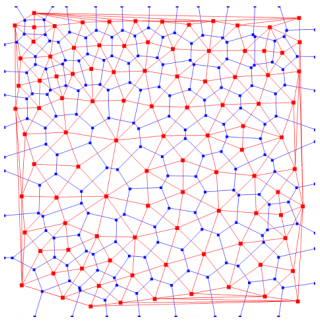

Figure 2: Example of a graph with Delaunay and Voronoi nodes and their interconnections (Dangschat and Weweler, 2017).

2.1 Polygons

The map is constructed using two graph structures that are closely dependent on each other. The first graph represents the Voronoi graph where each node is a polygon corner and edges are sides of polygons. The second graph is the Delaunay graph where polygons form nodes and two polygons having an adjacent side are said to have an edge between them. For each Delaunay polygon, there is a center inside. A visual representation of the graphs can be seen in Figure 1.

We generate random points and use these points to build the graph. After the polygons are constructed, we apply Lloyd relaxation to produce quasi-randomness in points. Lloyd relaxation replaces each random point by the centroid of the polygon. This technique helps to redistribute Voronoi nodes more evenly. An example of the final graph structure is illustrated in Figure 2.

2.2 Elevation

We combine two maps to generate the elevation map.

The first map is the fractal noise map that combines several layers of Perlin noise. Each layer of the Perlin noise, also knows as an octave, is generated with its *frequency* and *amplitude*. Additionally, there are two parameters - *persistence* and *lacunarity*. Persistence determines how much each octave contributes to the overall shape of the map

and is used to update the amplitude on each step. Meanwhile, the lacunarity determines how much detail is added or removed at each octave and adjusts the frequency.

The second map is a falloff map. For each point of the grid, it finds the distance to the center of the map, normalizes it to the unit range, and passes through a curve function. The curve function helps to redistribute the values and maps [0, 1] range values into another [0, 1] range.

We subtract falloff map values from the corresponding values of the noise map. This way, we generate an island-like height distribution. Finally, we clap the values by the lower bound with zero and form the ocean level.

2.3 Map graph

On this step, we use the Delaunay polygons and the height map generated in previous steps to construct a single map graph structure. The map graph consists of Voronoi nodes (points), Delaunay polygons defined as graph nodes, and directed-in-clockwise-order edges that are simply edges between Voronoi nodes. Moreover, we define points to be three dimensional and put the height values according to the elevation map. Additionally, each graph node has a property of node type that is to be filled on the next step with the types of biomes.

2.4 Moisture and biomes

In the current implementation, moisture is to be a single parameter that is defined by the user. The moisture value is a float number in a range [0, 1].

As long as we have height values for each node of the map graph and the value of moisture, we may assign biomes to the nodes.

First, biomes types are distributed according to a modified Whittaker diagram (A. Patel 2010) which is shown in Figure 3. However, there is a slight difference compared to the data provided in the figure. The height values in our map graph are not in the range of [0, 1]. That is why the breakpoints for the biome change are defined manually. In the current implementation, the border values are defined to be 22, 12, 3, 0 units. Usually, the pick of an island has a height of 25-35 units.

Second, graph nodes with low height are reassigned to be either the ocean (salt) water or the freshwater depending on whether the node is outside the island or inside.

Finally, we traverse the nodes neighboring to the ocean cells and assign them to be a sandy beach in case the slopes of the polygons are less than a specified value (in the current implementation it is 0.8 units).

2.5 Mesh and texture

In the final stage of the island generation, we create a mesh and apply a texture to it. The polygons are broken into triangles that are used to form a mesh. This task becomes simple as we have ordered the node edges in the clockwise order. Finally, there is a special library for texture rendering in the Unity3d engine, called GL. We use it to apply colors of biomes to the triangular polygons of the mesh.

3 Evaluation and Results

3.1 Evaluation Metrics

There are several approaches used for the evaluation of islands. These metrics are either computational or survey-based.

One of the computational metrics is the number of disjoint components (separate islands) in the map graph. The idea is that it would be preferable if there was a small number of components if any.

Another computational metric is the score of a complex function with the following formula:

$$F_{\text{Island}} = max(F_{\text{beach}}, F_{\text{mountains}}, F_{\text{flatland}}),$$

where each $F_{some\,type}$ is a triangular function (also known as a hat function) of some value:

- F_{beach} is the percentage of a coastline covered with a sand beach passed through its own parameterized triangular function.
- $F_{\text{mountains}}$ is the percentage of an island covered with mountains regions passed through its own parameterized triangular function.
- F_{flatland} is the percentage of an island covered with flat regions passed through its own parameterized triangular function.

Each triangular function has its own parameters - a and b. a is responsible for slope angle of the triangle, whereas b is the shift by the x-axis. A triangular function is defined with the following formula:

$$f(x, a, b) = \begin{cases} 1 - a * |x - b|, \text{if } |x - b| < \dfrac{1}{a}, x \in [0, 1] \\ 0, \text{ otherwise} \end{cases}$$

Additionally, it is possible to apply a survey-based scoring. The evaluation is done by letting several respondents answer a questionnaire and averaging results. Here are some questions that might be used in the survey:

- How attractive is the island for you? (Score from 1 to 10)
- Evaluate how extraordinary the island looks. (Score from 1 to 10)
- Are there any errors on the generated map?

The results of the survey would be used to decide on the best island look. Nevertheless, we will skip this survey-based evaluation in the following section when showing results.

3.2 Results

In this subsection, we may observe two examples of generated islands and their evaluation scores. One may try to generate other variants of islands using the source code of the project.

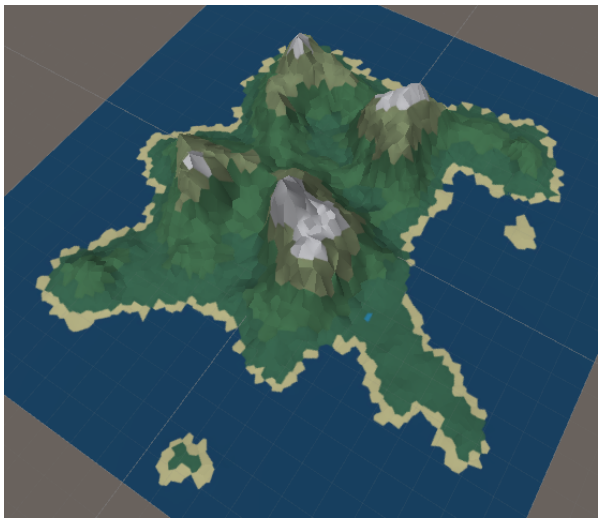

Figure 3: An example of a generated island with *moisture* = 1.0 and *seed* = 42.

In Figure 3, you may see an island generated with the default parameters, *moisture* = 1.0, and random *seed* = 42. There are 3 disjoint components in the graph, thus, there are one main island and two small lands. 80% of the coastline is covered with the sand beach. The a and b parameters for the triangular function are equal to 3 and 0.85, respectively. That is why the $F_{beach} = 0.86$. 18% of the land is covered with mountains. $a = 5$ and $b = 0.15$ for the triangular function of mountains. Thus, we prefer a small but non-zero quantity of land covered with mountains. $F_{mountains} = 0.83$. Finally, 48% of the land is flat, and, as soon as $a = 3$ and $b = 0.6$, $F_{flatland} = 0.64$. The final maximum score of the function F_{island} is equal to 0.86.

There is another example provided with the default parameters, except for *moisture* and random *seed* that are equal to 0.0 and 2, respectively (see Figure 4). Here are the metric scores related to the island from Figure 4:

- There is only one disjoint component.
- 76% of the coastline is covered with a sand beach and $F_{beach} = 0.72$.
- 16% of the land is covered with mountains and $F_{mountains} = 0.95$, which means that the percentage is very close to the ideal value.
- 42% of the land is flat and $F_{flatland} = 0.45$.
- The final maximum score is 0.95.

All parameters of the triangular functions are the same as in the previous example.

4 Further Work

This island generator can be used as the basis for such games as the Sea of Thieves. The islands may be decorated with rivers, stones, vegetation, and buildings. These objects

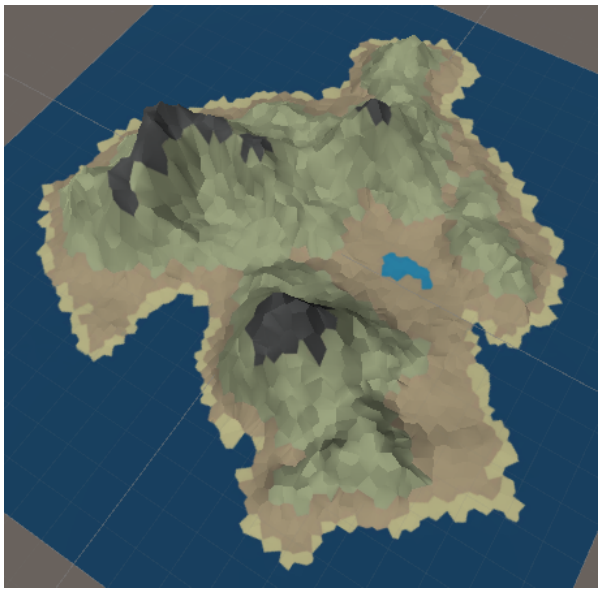

Figure 4: An example of a generated island with $moisture = 0.0$ and $seed = 2$.

may be instantiated with yet another procedural generation algorithm. Additionally, enthusiasts may try to add different other biome types that may lead to some extraordinary island look.

References

[1] Dangschat, M., Weweler, Y. N (2017). Procedural Generation of 3D Maps A Study of Polygon Graph based Map Generation.

[2] Patel, A. (2010). Polygonal map generation for games. Red Blob Games, 4.

[3] Togelius, J., Yannakakis, G. N., Stanley, K. O., Browne, C. (2011). Search-based procedural content generation: A taxonomy and survey. IEEE Transactions on Computational Intelligence and AI in Games, 3(3), 172-186.

[4] Wikipedia contributors. (2020, September 16). One Piece. In Wikipedia, The Free Encyclopedia. Retrieved 13:00, September 22, 2020, from https://en.wikipedia.org/w/index.php?title=One_Pieceoldid=978750131.

[5] Wikipedia contributors. (2020, September 19). Sea of Thieves. In Wikipedia, The Free Encyclopedia. Retrieved 14:30, September 22, 2020, from https://en.wikipedia.org/w/index.php?title=Sea_of_Thievesoldid=979175385.