**A Comparison of Two Procedural Terrain Generation Methods: Perlin and Fractal**

Tim Hake

**Abstract**

There are many methods of producing procedurally generated terrain and such terrain has many applications and uses. Different applications bring different requirements and limitations to challenge these methods. In this project I seek to analyze and define the differences, strengths, and weaknesses between two specific methods of procedural terrain generation in order to determine where they are best used. The Dungeon Template Library API was used to run its “Perlin Island” and “Fractal Island” methods and render the results in the included “image viewer” in two dimensions. Perlin Noise proves to create realistic, appealing terrain while Fractal Noise proves to be chaotic with high variance in elevation distribution.

1. **Introduction**

Procedurally generated terrain has a wide range of important uses, from entertainment to scientific research to military training. Procedurally generated terrain can be used for Civil engineering or flight and vehicle simulations. It is frequently used in video games and animated movies. It can even be used for military training [9]. The objective of this project is to compare the Fractal Island method to the Perlin Island method in the Dungeon Template Library API. Different important aspects of the methods will be analyzed, such as realism, variability, and customizability.

Procedural terrain generation is available in a variety of methods, and it isn’t clear which methods are better than others at specific tasks. Defining the differences between two relatively basic methods will create a basis for making comparisons between other methods of procedural terrain generation. This will allow further analysis to be done in the future and even potentially allow new, more specialized procedural terrain generation methods to be created. Perlin noise is widely used, and is a basis for other methods of terrain generation [7] [11].

1. **Materials & Methods**

The comparison between the Perlin Island and Fractal Island methods was completed using the Dungeon Template Library API. The Dungeon Template Library API contains a wide variety of methods for procedurally generating different structures, terrain, and dungeons. A subsection of these methods focuses on procedurally generating terrain. These methods are programmed in C++ which were accessed and worked with using Visual Studio 2019. The API was used to run the Perlin Island and Fractal Island methods ten times each. Each time the method was run, it was rendered in the “Dungeon Template Library Viewer”, which creates a two dimensional rendering of the terrain generated. Numerical trends such as the distribution of different elevations were analyzed. This analysis focuses on the percentage of landmass generated that is in each elevation range designated Low, Medium, High, and Very High. Specific colors were associated with each elevation designation when recording the results.

The Fractal Island method was run by using the FractalIsland.cpp program included in the Dungeon Template Library API. A two dimensional render of this output can be viewed in the program included in the library called “Dungeon Template Library Viewer”.

The Perlin Island method was run in a very similar method to the Fractal Island method. The PerlinSolitaryIsland.cpp file was used to generate terrain, and the output was rendered in two dimensions by “Dungeon Template Library Viewer”.

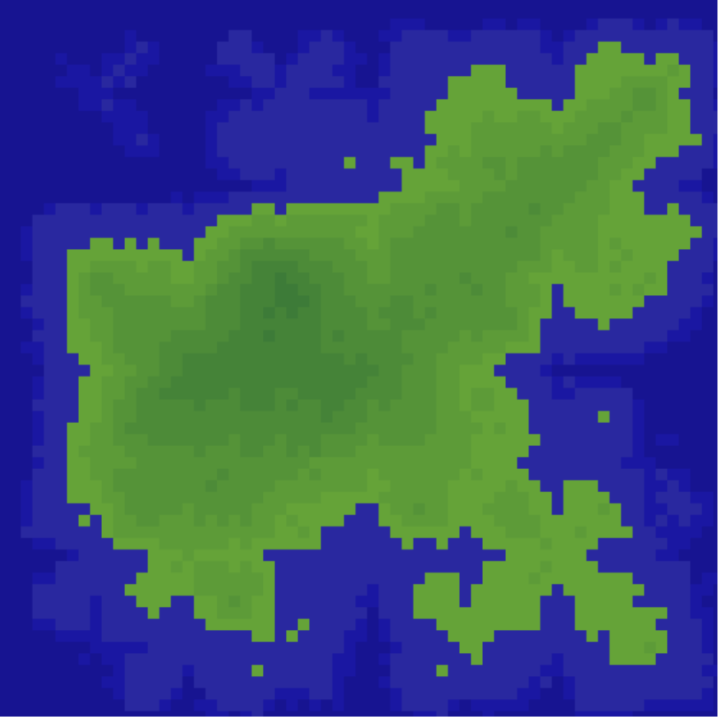
Once the renders of the methods were saved as .png files, they were put into the open source program paint.net. Using this program the quantity of pixels that fell into each category was recorded in order to determine the percentage of the landmass that they made up.

A link to a copy of the Dungeon Template Library API is given here: <https://github.com/Kasugaccho/DungeonTemplateLibrary>

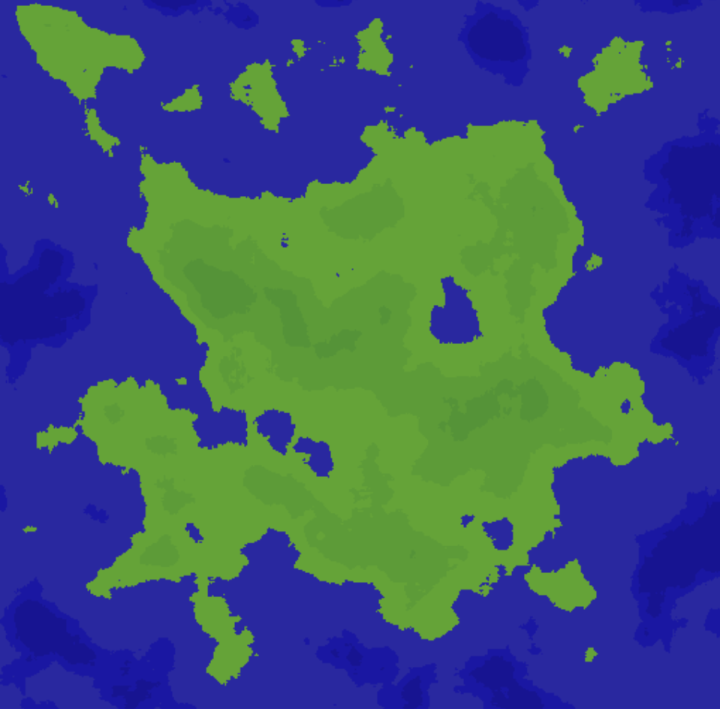
1. **Results**

As expected, the Fractal Island method was far less consistent in the terrain output than the Perlin Island method.

In the 2D rendering, the Fractal Island method’s output (**Fig. A**) has a significantly lower pixel count than the Perlin Island method (**Fig. B**). The shades of green on the landmasses represent elevation, with darker shades of green being higher elevation. I defined these elevations into four categories: Low, Medium, High, and Very High Elevation. Low, medium and high elevations are considered the main elevation designations, as terrain that falls into the category of Very High rarely appeared in the renderings of the islands in both methods. There were three Fractal islands that displayed Very High elevation and three Perlin islands that displayed Very High elevation.



(A) Fractal Island Method 2D Rendering #4



(B) Perlin Island Method 2D Rendering #5

The Fractal Island method often produced coastlines with straight, unnatural shapes. Additionally, the distribution of the three main terrain elevations is remarkably more even among Fractal islands, as can be seen in the graphs below. The percentage of low terrain in a Fractal island went anywhere from 28 percent to 63 percent, compared to the low terrain in Perlin islands ranging from only 57 to 66 percent. **(Fig. C, Fig. D)**

(C) Distribution of Elevation in Fractal Islands

(D) Distribution of Elevation in Perlin Islands

The Perlin Island method produces very natural and smooth looking islands with irregular coasts that mimic real geography. Perlin islands have a far more consistent distribution of elevations, consistently having at or above 60 percent of the landmass being Low elevation and 30 percent of the landmass being Medium elevation. High elevation terrain never reached above 10 percent or below 3 percent. **(Fig D)** Additionally the Perlin Island method tends to create smaller islands surrounding the main landmass that also have a natural shape to them. **(Fig B)**

Despite the differences in variance, the averages for both Perlin and Fractal islands are closer than expected. Perlin Islands had significantly more Low terrain on average at over 10 percent more, and Fractal islands had 8 percent more High elevation terrain on average. However, both Fractal and Perlin methods had Medium terrain percentages within 3 percent of each other. **(Fig F)**

(F) Average Elevation Distribution of Perlin vs Fractal Islands

A link to the spreadsheet containing all of the elevation distributions on each trial can be found here: https://drive.google.com/open?id=1O\_Hg7cwtyPWjXlAQwT8c9HIyqhpaYKWt

1. **Discussion**

The results from comparing the Perlin Island method with the Fractal Island method mostly met expectations. The Fractal Island method produced unrealistic, highly irregular terrain with dramatic peaks and valleys that do not reflect natural terrain. This is reflected in the more even distribution of terrain elevations. Meanwhile, the Perlin Island method produced terrain that had smoother peaks and valleys, represented in a consistent elevation distribution [7]. The Perlin Island method also successfully emulated natural irregular coastlines well. The realism of the mountains in the Perlin Island method actually exceeded expectations, appearing more varied than expected although far less chaotic than the Fractal Island method. This was unexpected because the results far exceeded the level of realism achieved by the regular Perlin terrain generation seen in “Designer Worlds: Procedural Generation of Inﬁnite Terrain from Real-World Elevation Data” [4].

Despite its shortcomings, the Fractal method performed better than expected. The average percentage of each elevation was comparable to the Perlin method at Medium elevation, although it has a significantly lower Low elevation and higher High elevation percentage.

The comparison made here is limited by the scope of testing. If more terrain generation methods could have been used, more knowledge could have been gained. For example even within the Dungeon Template Library API there are methods such as Cellular Automaton Island or even White Noise that could have been used to generate additional terrain for comparison. Additionally, a greater level of control over the terrain generation method could have led to interesting results by testing a wide range of parameters as seen in “Parametrically controlled terrain generation” [9]. I would love to try out other frameworks or APIs to explore more possibilities for terrain generation. It would also be interesting to create my own terrain generation methods from scratch, potentially creating new methods or combining existing methods together like in “Designer Worlds: Procedural Generation of Inﬁnite Terrain from Real-World Elevation Data” [4].

Finally, the elevation distributions used in this analysis were gathered from analyzing the pixels in the output renders from running the methods. If the exact elevation values were harvested from the program itself, the results would be more accurate.

While Fractal Noise proves to fail at creating realistic terrain, Perlin Noise is utilized to accomplish realistic terrain that has believable features such as mountains, valleys, and irregular coastlines. These features are visually appealing and show that Perlin Noise is a powerful option for procedural terrain generation.

**References**

[1]

S. Stachniak and W. Stuerzlinger, “An Algorithm for Automated Fractal Terrain Deformation,” *Computing and Informatics / Computers and Artificial Intelligence - CAI*, vol. 1, Jan. 2009.

[2]

J. Doran and I. Parberry, “Controlled Procedural Terrain Generation Using Software Agents,” *IEEE Transactions on Computational Intelligence and AI in Games*, vol. 2, no. 2, pp. 111–119, Jun. 2010, doi: [10.1109/TCIAIG.2010.2049020](https://doi.org/10.1109/TCIAIG.2010.2049020).

[3]

R. M. Smelik, T. Tutenel, K. J. de Kraker, and R. Bidarra, “Declarative Terrain Modeling for Military Training Games,” *International Journal of Computer Games Technology*, 2010. [Online]. Available:<https://www.hindawi.com/journals/ijcgt/2010/360458/>. [Accessed: 28-Feb-2020].

[4]

I. Parberry, “Designer Worlds: Procedural Generation of Inﬁnite Terrain from Real-World Elevation Data,” vol. 3, no. 1, p. 12, 2014.

[5]

E. W. Weisstein, “Fractal.” [Online]. Available:<http://mathworld.wolfram.com/Fractal.html>. [Accessed: 28-Feb-2020].

[6]

“Snapshot.” .

[7]

K. Perlin, “Improving noise,” in *Proceedings of the 29th annual conference on Computer graphics and interactive techniques*, San Antonio, Texas, 2002, pp. 681–682, doi: [10.1145/566570.566636](https://doi.org/10.1145/566570.566636).

[8]

“Full Text PDF.” .

[9]

K. R. Kamal and Y. S. Uddin, “Parametrically controlled terrain generation,” in *Proceedings of the 5th international conference on Computer graphics and interactive techniques in Australia and Southeast Asia*, Perth, Australia, 2007, pp. 17–23, doi: [10.1145/1321261.1321264](https://doi.org/10.1145/1321261.1321264).

[10]

“Procedural generation,” *Wikipedia*. 20-Feb-2020.

[11]

D. Marshall, D. Delaney, S. McLoone, and T. E. Ward, “Representing Random Terrain on Resource Limited Devices,” presented at the CGAIDE 2004 - International Conference on Computer Games: Artificial Intelligence, Design and Education, Reading, U.K., 2004.

[12]

D. Callele, “Semi-Random Terrain Generation,” p. 2.