Realistic Terrain Generation

Jaiden Baker - 300376937
jaidencolebaker@gmail.com

Abstract

This document details and discusses the Realistic Terrain Generation project. It starts by giving a general introduction on scope and challenges of the project regarding plate tectonics and biome classification, along with a background of related works used as references. It then details how the challenges were resolved, along with the implementation of the processes and algorithms used. The results of the implementation are presented and its effectiveness is reflected on. Finally, the conclusion summarises the results and states the successes, limitations, and potential improvements for the implementation.

Table of Contents

1. Introduction

1.1. Scope

This project's goal was to procedurally generate realistic terrain at a terrestrial scale in real-time situations. 'Realistic' is defined as terrain that demonstrates real-world features and environments. An example real-time situation would be generating a world map for a video game, which should at most take a minute.

Terrestrial features define the overall shape of large parts of the terrain e.g. continents, oceans, islands, mountains, and valleys. These and are the result of terrain's height generation algorithms. Physical environment of the terrain can be defined by its biome e.g. desert, grassland, tundra, and forests.

Common terrain generation procedures based on noise or fractals do not produce realistic looking terrestrial features, hence are not viable. Instead, this project uses plate tectonics. By simulating the physics which formed Earth's terrain, in theory realistic terrain with similar terrestrial features can be generated. Fluvial erosion was also considered, but due to the terrestrial scale its effects would be minimal.

This project uses Whittaker's Biome model to determine physical environments. The model itself is derived from average annual precipitation and temperature of Earth's biomes. As long as annual precipitation and temperature can be modelled realistically, the biomes that define the physical environment can also be realistically modelled using this model.

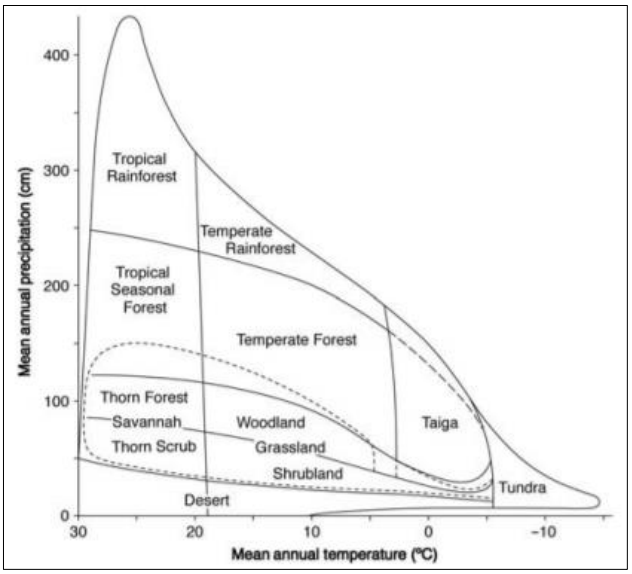
1.2. Technical Challenges

Plate tectonics happens at an enormous terrestrial scale, in three dimensions, over billions of years. Generating it in a reasonable amount of time requires multiple shortcuts and simplifications, some of which may compromise the realistic nature of the terrain.

Tectonic plate interactions deforming physical terrain is a complicated process. There are many ways in which it can deform terrain, such as convergence, divergence, slip faults and volcanic activity.

Plate movement is also complicated topic undergoing lots of debate. The leading theory proposes that currents within the upper mantle underneath a plate's crust force it to move through Basal drag; friction between plate and the mantle causing the plate to move with the mantle. Plates can also move due to Slab suction; gravity pulling down on plates. An accurate plate movement model would therefore have to accurately model mantle currents underneath the surface of tectonic plates, despite the fact that there are lots of unknown factors on what affects this.

Whittaker's biome model (right) has many non-linear boundaries between biome classifications. Many graphs online depict this model, however there is no mention of any actual mathematical equations. These equations might be included in Whittaker's book which includes the model, however it was published nearly 50 years ago and is not available in New Zealand.



Precipitation is the result of many other physical processes. These include wind currents (a result of barometric pressure), interference of mountain ranges, distance the wind must travel from multiple bodies of water to a specific spot, and sunlight intensity on the bodies of water. The simplest part is sunlight intensity, as that can be derived from latitude. However, the rest require multiple other complex models and algorithms which can take years to learn an understand.

Fortunately, temperature is simpler. Its main factors are sunlight intensity (latitude), whether or not the area is ocean or land, and altitude. All of these factors are easily derived from the physical model's geographic height and location, which are necessary anyway for plate tectonics.

1.3. Related Works

Below is a list of references to related works. They all focus on terrain generation and biome classification, and are ordered by their relevance. The first three references contain algorithms and processes similar to the ones in this project. The rest are alternative models that were considered, but not used.

[1] A. Gainey, *"Procedural planet generation," Experilous.com*, Sep. 30, 2014. [Online]. Available: http://experilous.com/1/blog/post/procedural-planet-generation [Accessed 14 Sept. 2018].

[2] L. Viitanen, "Physically Based Terrain Generation," B. E. thesis, Helsinki Metropolia University of Applied Sciences, Helsinki, 2012. [Online]. Available: http://www.theseus.fi/bitstream/handle/10024/40422/Viitanen_Lauri_2012_03_30.pdf [Accessed 12 Step. 2018].

[3] R. H. Whittaker, *Communities and Ecosystems.* New York: MacMillan Publishing Company, 1975.

[4] J. Rankin, "The Uplift Model Terrain Generator," *International Journal of Computer Graphics & Animation*, vol 5. no. 2, April 2015. [Online]. Available: http://airccse.org/journal/ijcga/papers/5215ijcga01.pdf [Accessed 12 Sept. 2018].

[5] G. Cordonnier, J. Braun, M. Cani, B. Benes, E. Galin. *Large Scale Terrain Generation from Tectonic Uplift and Fluvial Erosion*. Computer Graphics Forum, Wiley, 2016, Proc. EUROGRAPHICS 2016, 35 (2), pp.165-175. Available: https://hal.inria.fr/hal-01262376/document [Accessed 31 Aug. 2018].

[6] M. Dangschat, Y. Weweler. *Procedural Generation of 3D Maps A Study of Polygon Graph based Map Generation*. p. 9. Available: https://yweweler.de/downloads/posts/2017-03-15-polygon-graph-procedural-map-generation/report-patel10-stanford.pdf [Accessed 31 Aug. 2018].

[7] Patel, A. (2018). *Making maps with noise functions*. [online] Redblobgames.com. Available: https://www.redblobgames.com/maps/terrain-from-noise/ [Accessed 31 Aug. 2018].

2. Implementation

The project was implemented using Java, chosen as the developer is most familiar with it. It is also designed to make programming faster, which is suitable for experimental approaches, such as the ones used in this project.

The source code for this project can be found on this GitHub repository.

2.1. Efficient Data Model

In order to make tectonic plate calculations as efficient as possible, plate objects are represented by a 1D array of floating-point values. Each cell in the array represents the height of a 2D co-ordinate point on the map. I used 1D instead of 2D as iteration is faster (1 array lookup operation instead of 2). I also added a 1D bitmask array for faster collision and overlap calculations between plates. Finally, some methods such as 'forEachCell' are included to take advantage of Java's JIT compilation. Since multiple algorithms use them, the methods are used more often than if repeated in multiple other spots, therefore the JIT compiler is more likely to optimise those methods.

The Map object contains all data that a terrain map has for this project, such as heights and biomes. It also contains a fixed random seed which is used by all generation algorithms to re-generate a map with the same seed and parameters. Most importantly, it contains the 'cellIndex' method, which gets the 1D index for a map's cell from 2D co-ordinates. It implements screen wrapping both horizontally and vertically. It uses wrapping, otherwise artefacts form when plates move away from the edge of the map. It uses simple wrapping, and not globe-based wrapping as that would require cells to have different virtual sizes on the final map, which would be computationally expensive.

The plate list object can be used instead of an array list, providing faster plate lookups from a 2D co-ordinate by using a pre-processed plate index mask.

2.2. Tectonic Plate Generation

Tectonic Plate Generation uses the same process as Andy Gainey's planet generator [1]. The algorithm is implemented in this class, and is as follows:

- For n plates, pick n random points on the map and add them to a queue of points
- While the queue contains points
  - Remove a random point
  - Add the point to a combined mask, with its plate index as the mask's value
  - For each adjacent point, if it's not in the mask, give it the same plate index and add it to the queue.
- Create n new plate objects
- For each point in the mask, get the plate using the index value at that co-ordinate and add it to the plate.

This algorithm was used because it creates natural, un-even plate boundaries. Generated plates also have a variety of shapes and sizes, further enhancing the realistic nature.

A Voronoi cell-based generator was trailed, which assigns random points for the centre of each plate and then assigns all cells a plate based on the nearest plate centre. However, it produced unrealistic linear boundaries between plates.

2.3. Initial Heightmap Generation

Instead of performing thousands of land deformation iterations to create large and small terrestrial features, it's much faster to make an initial heightmap instead. The initial map can demonstrate large features such as continents, while the deformer only needs to run a handful of iterations for smaller features such as mountains and island chains.

This class generates an initial heightmap using the same plate generator algorithm, although it uses a different set of parameters than the actual map's plates. This makes the coastline of continents rough and realistic.

It then assigns plates as either land or ocean. Points on land are elevated with a gentle root function for based on how far inland it is. Ocean points are similar, but are de-elevated instead. This provides elevation for inland areas and deep seas. It also forms continental shelves due to the land and ocean functions having different directions.

A Simplex Noise generator was trailed, but it produced unusual continental shapes and had an unrealistically smooth coastline.

2.4. Tectonic Plate Deformation

The tectonic plate deformation process uses an iterative algorithm which has three stages: plate movement, deformation by convergence, and deformation by divergence. These three stages are executed in sequence to deform the map in a single iteration.

For simplicity sake, this project only uses convergence and divergence to deform the map. Additional methods such as slip faults and volcanic activity are not used, but could be added in later in order to enhance the terrain's realistic nature.

The abstract Tectonic Plate Deformer class handles the number of steps, while extending classes define a single iteration, initialisation processes and early termination conditions. Basic Plate Deformer handles plate movement, convergence and divergence.

2.4.1. Movement

Instead of modelling plate movement using physical interaction with constantly changing mantle currents, I experimented with a simpler model based solely on interactions between plates themselves.

I did this by randomising the initial velocities of the plates. Then with each step collisions between plates were used to calculate an exchange of momentum. Only linear momentum is considered: there is no rotational velocity and momentum is not used as it would be too difficult to implement with the efficiency-focused data model.

The collision algorithm is implemented in Tectonic Velocity Calculator. Initialisation and early termination conditions are defined in the Basic Plate Deformer class.

2.4.2. Convergence

Convergence works by calculating the overlap between all plates. Plates are subducted under others based on their order. If a plate is overlapped by a higher plate, the overlapped area is removed from the lower, subducted plate and added to the plate on top. Then, the height of each cell is increased based on the 'depth' of the overlap at that point i.e. distance to the nearest non-overlapping point.

The convergence algorithm is implemented in Tectonic Convergence.

2.4.3. Divergence

Divergence has a similar algorithm to convergence. First, the regions which aren't occupied by any plates are calculated. Then, all edges on the regions are added to a queue, assigned a plate based on the nearest adjacent plate. Using this queue, the unfilled regions are uniformly flood-filled, assigning a plate and decreasing the height based on the height of the previous cell. The decrease is logarithmic towards a global minimum value set by the user.

The divergence algorithm is implemented in Tectonic Divergence.

2.5. Biome Assignment

As stated in the scope, this project uses Whittaker's biome model in order to classify land biomes. The overall implementation is found in the Biome Classifier class, which has a few algorithms included while handling precipitation and temperature models off to other classes. The overall algorithm first sets all points on the map as either ocean or land. Then, it calculates precipitation and temperature data for the whole map. Finally, for each land point that data is used by a simplified Whittaker's biome model to assign it a specific land biome.

2.5.1. Separating Ocean & Land Biomes

Ocean and land biomes are separated based on their height and a global sea level.

The sea level algorithm takes in a desired percentage of the map to be covered by ocean. It then takes all heights on the map, sorts them in ascending order and picks the height that is the same percentage in the ordered height list. This height defines the global sea level, and ensures that a fixed amount of ocean and land biomes are present in the map.

Then the Biome classified class goes through all cells on the map, assigning them as ocean or land if they are below or above the global sea level respectively.

2.5.2. Precipitation Model

This project simplifies the effect of wind currents due this complex nature. The precipitation model only factors in two elements when assigning precipitation to a point: the inland distance and the current latitude. Inland distance is used in place of wind currents: the further inland a point is, the more distance rain clouds have to cover to reach it. This means more opportunities the clouds to lose all moisture before reaching that point. The ratio between these factors, and the rate of moisture loss from inland distance, can be configured by the user.

Another model was made which combined simplex noise with latitude, replacing distance to the ocean. However, it produces very smooth boundaries between biomes which may not be particularly realistic.

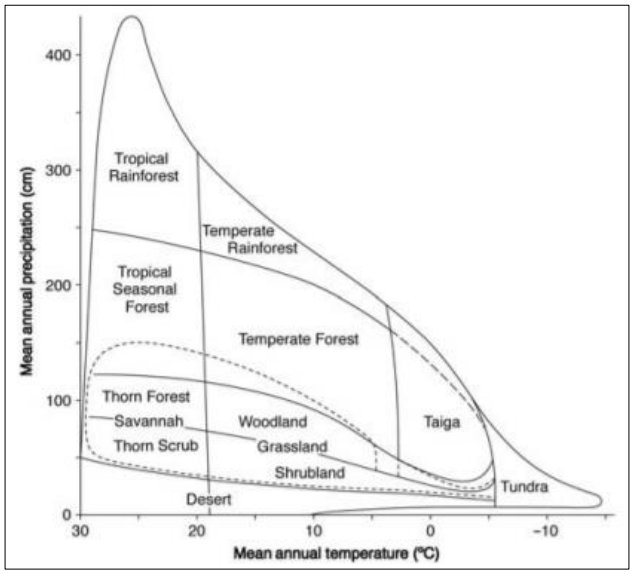
2.5.3. Temperature Model

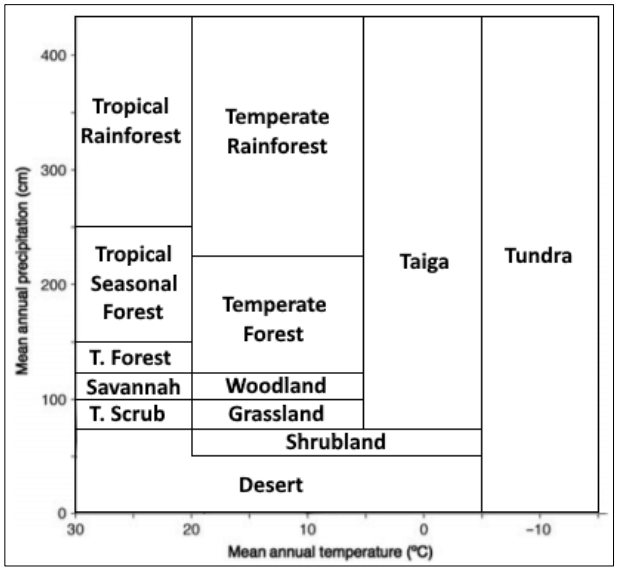
The temperature model for land regions models what happens in the real world.

First, a polar and equator latitude temperature are randomly generated between 30°C — 40°C and -15°C — -10°C. Then, for each row of map points a latitude temperature is interpolated between these two values. Finally, for all points a temperature is assigned by taking the latitude temperature and subtracting temperature linearly based on the cell's elevation above sea level.

2.5.4. Whittaker's Model

Instead of using the difficult-to-obtain non-linear model (left), a simpler linear model was derived from it (right). This also greatly reduces computational costs when assigning biomes to different points on the map. The classification algorithm is implemented in the Biome Classifier class's 'getLandBiome' method.



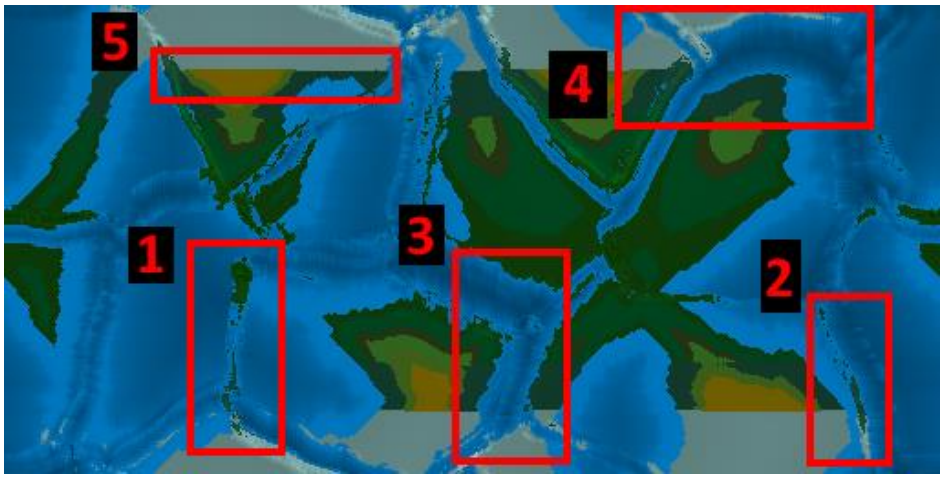


3. Results & Discussion

The final program that packages the algorithms is capable of generating and rendering maps that demonstrate realistic terrestrial features and biomes. It is capable of rendering: Tectonic Plates, Height, Biomes, Precipitation and Temperature maps.

3.1. Realism

Putting this altogether and rendering both the Height and Biome map to produce a world map, the application produces maps such as this one:

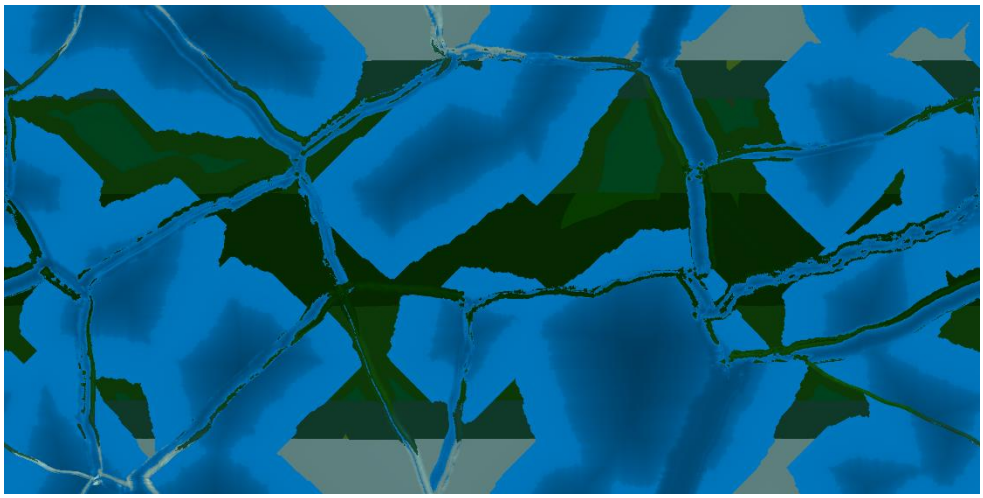


This generated map exhibits some realistic traits and terrestrial features, however certain aspects aren't particularly realistic.

The map generates plenty of islands and island chains (1, 2). It also exhibits continental drift traits (3). The continents themselves are passable as realistic; they have realistically uneven edges yet have distinct coastlines. although not exactly spectacular in their overall shape, they can still be described as continents.

However, the terrain generator only generates mountains on the edges of continents (4), most likely due to plate collision moving plates back and forth, performing convergence and divergence within close proximity. It also produces unnaturally straight-line borders between biomes (5). This mostly due to the linear approximation of Whittaker's biome model.

Furthermore, generating larger maps smooths the uneven nature of coastlines, highlights sharper edges of continents and produces mountain ranges in the middle of the ocean:



A potential solution to this scaling issue without re-writing the algorithms would be to instead generate a smaller map, then scaling it up by interpolating / smoothing between generated pixels.

3.2. Performance

This map is 256x128 pixels large (130,000 pixels). After the Java JIT compiler warmed up it took:

- 140ms to generate the initial continents
- 60ms to generate the plates
- 600ms to perform 25 deformation iterations
- < 50ms to classify biomes

Overall, the map took 850ms to generate: well within real-time situations for media such as video games for world maps of this size.

Plate deformation and biome classifications have a linear computational complexity based on the number of pixels and number of plates. Since the number of plates is usually small in comparison to pixels, it has less of an effect.

However, generating plates and continents has a non-linear complexity of $O(n) = n\sqrt{n}$, where n is the number of pixels. This means plate generation for larger maps takes much longer than the other algorithms involved in the map generation. It is most likely due to randomly removing elements from an array queue in the tectonic plate generator. It could possibly be reduced to $O(n) = n\log(n)$ by replacing it with a binary tree-backed queue supporting random element removal. This is not trivial, as no Java libraries exist which support this exact usage.

3.3. Conclusion

The generator as it stands is capable of producing mostly realistic terrain at a terrestrial scale. As long as small map sizes such as 256x128 pixels are used, the quality and performance of the overall process are suitable for real-time situations.

The caveats and potential fixes are:

- **Unrealistic linear biome boundaries** – Update Whittaker's model to be non-linear and more accurate, or use precipitation and temperature models which have more noise / uneven regions.
- **Poor quality with large maps** – extending the algorithm to up-scale and interpolate smaller maps when generating larger maps
- **Non-linear performance for plate generation** – Replace array queue with random removal binary tree queue. Upscaling algorithm will also help with performance for larger maps.