McGill University

COMP 521

Modern Computer Games

TerraGen: Procedural Terrain Generation and Resource Analysis

Benjamin San Souci
Maude Lemaire

April 14, 2015

1 Introduction

TerraGen aims to generate realistic island terrains using a relaxed polygonal map and assigning biomes based on simulated moisture and elevation. Following the map generation process, the city generation algorithm analyzes the terrain and produces a set of cities with unique statistics directly related to their location and access to resources.

Island generation was modeled after Amit Patel's "Polygonal Map Generation for Games"[6], where Chrisophe Le Besnerais's JavaScript version, "Island.js"[5] was extended to accommodate for additional features. First a graph structure of polygons is generated, then annotated with elevation, moisture, and labeled as either land or water. In order to generate a more natural-looking island, Voronoi polygons were generated and relaxed using an implementation by Raymond Hill [4]. Extensions to the implementation included adding fishing regions, minerals, volcanoes, tectonic plates, forest density, and fauna abundance.

The TerraGen city statistic generation addresses a problem that has been little discussed in the scope of games beyond the intersection of geography information systems (GIS) and 3D game engines. Using the generated plot, the city generation algorithm finds a suitable location at which to place a city. From this location, resources in close proximity are identified and used in determining the city's population size, army size, happiness level and wealth, among other features.

The result is a randomly generated island with over two dozen biomes, oceanic regions, dispered minerals and a handful of cities. A sample map at different stages of generation is included in the methodology portion of the report.

The project was separated into two major components: the terrain generation and extensions, and the city statistics generation. Maude was mainly responsible for the terrain generation portion of the project and the majority of the final report. Benjamin produced the city generation algorithm.

2 Background

Map generation has been a important focus within the gaming industry in recent years as a means of promoting re-playability without having to manually generate a large number of maps. By randomly generating a map with new instances of a game, the players must face new challenges and adapt their game-play to suit the new terrain. A wide range of approaches already exist, with varying levels of complexity and randomness.

2.1 Grids

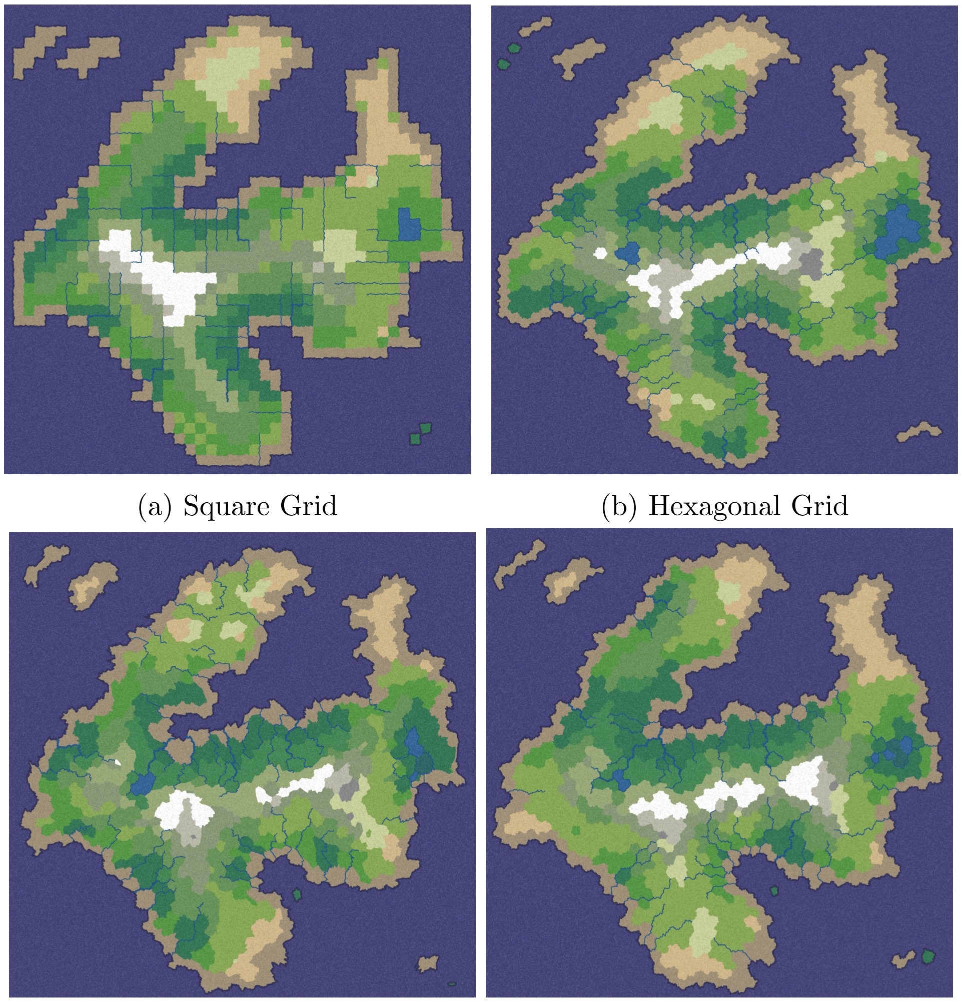
Terrains are typically built off of a grid system; this can be a simple tile-based square grid, a hexagonal grid, or a polygonal grid either composed of randomly sized convex shapes or relaxed polygons. Figure 1 displays the difference between each of these grid types and how “natural” each of these appear in island generation.

Hexagonal grids and polygonal grids typically provide the most natural-looking islands. Hexagonal grids have been extensively developed by academics such as Clark Verbrugge[7], and can be perturbed to add irregularity. Polygonal grid generation has mainly revolved around randomly generating a series of Voronoi polygons. A number of algorithms generate Voronoi diagrams based on a set of random points (Fortune’s algorithm[2]), a set area (Lloyd’s algorithm[1]) or Delaunay triangulations (Bowyer-Watson algorithm[8]). In particular, the set of points upon which the polygons are generated can be mapped to known natural phenomena. For example, researchers at the Université de Lyon,generated Voronoi diagrams based on hydrology [3]. As an extension, Lloyd relaxation can be used to distribute points more evenly for diagram generation.

Following Patel’s polygonal map generation methods, once a Voronoi diagram is generated, the internal representation consists of nodes and edges; the first is used for identifying adjacent polygons. The second represents the Delaunay triangulation which can be used for pathfinding. Conceptually, every corner in the triangulation corresponds to a polygon center and every edge to an edge in the diagram. As polygons in the map are irregular, it is difficult to make assumptions about neighboring cells and paths from one point to another; this representation ensures that the map is traversed in a meaningful way throughout the generation process.

2.2 Islands

Realistic islands must maintain a rather circular shape, with some random variation. A radial function is suggested, using sine waves to produce a round island. The island is generated according to polar coordinates stemming



(a) Square Grid (b) Hexagonal Grid

(c) Random Polygon Grid (d) Relaxed Polygon Grid

Figure 1: Islands generated using Amit Patel's demo. Each is generated using Perlin noise, displays biomes, and demonstrates the use of different grid types.

from the center of the canvas. Depending on the sine waves, the island can resemble an octopus, with thin strips of land extended from the center of the island.

Alternatively, Perlin noise can be generated and used to control the shape of the island given some constraints. This method provides more realistic island shapes due to the fact that Perlin noise consists of visual details of equal size; this gives a more controlled random appearance, closely imitating natural textures.

Once land polygons have been identified, ocean is filled in from the edges and any remaining "water" polygons unreachable from the fringe of the map (landlocked) are marked as lakes.

2.2.1 Elevation

In generating terrains, in order to provide a more realistic map, a natural-looking elevation must be assigned to each grid unit. Some typical approaches include using Perlin noise, real-world data, or fractal landscapes.

3 Methodology

4 Results

5 Conclusions & Future Improvements

References

[1] Qiang Du, Vance Faber, and Max Gunzburger. Centroidal voronoi tessellations: Applications and algorithms. *SIAM Review*, 41(4):637–676, 1999.

[2] S Fortune. A sweepline algorithm for voronoi diagrams. In *Proceedings of the Second Annual Symposium on Computational Geometry*, SCG '86, pages 313–322, New York, NY, USA, 1986. ACM.

[3] Jean-David Génevaux, Éric Galin, Eric Guérin, Adrien Peytavie, and Bedřich Beneš. Terrain generation using procedural models based on hydrology. *ACM Trans. Graph.*, 32(4):143:1–143:13, July 2013.

[4] Raymond Hill. Javascript-voronoi, March 2015.

[5] Christophe Le Besnerais. Island.js, March 2015.

[6] Amit Patel. "polygonal map generation for games", September 2010.

[7] Clark Verbrugge. Hex grids.

[8] D. F. Watson. Computing the n-dimensional delaunay tessellation with application to voronoi polytopes. *The Computer Journal*, 24(2):167–172, 1981.