Procedural Generation Using Noise

Michael Li

DRAFT 1.0.1

Dr.Dianne Hansford — Director

Dr.Yoshihiro Kobayashi — Second Committee Member

Ira A. Fulton Schools of Engineering
School of Computing, Informatics, and Decision Systems Engineering
Spring 2021

Abstract

Procedural Content Generation is a method of creating data algorithmically, often using stochastic models. These methods can be used to generate complex environments as opposed to manually creating environments by hand or by using photogrammetric techniques. Procedural generation can use a variety of techniques to achieve a stochastic or partially stochastic goal, including methods such as fractals, noise, deep learning as examples.

Contents

1. Introduction

This paper surveys various methods of procedural generation and their applications in generating geological formations. While procedural generated content can vary from terrain to creatures and stories, the focus of this paper is primarily on terrain and more specifically geological formations. One of the most famous cases of this procedural generation for geological formations is in Minecraft, which implements a modified version of Perlin noise in order to generate all of its worlds. Other examples can include games such as the Elder Scrolls II: Daggerfall, which employed various forms of procedural generation to determine the location of non-player characters, the layout of dungeons, as well as the terrain itself. In more complex cases, procedural generation can be used to create fake histories, with the more well known example of Dwarf Fortress. However, procedural generation's applications are not only limited to games. In the creation of the Lord of the Rings movies, procedural generation was applied to help create the many different forms of orcs in the movie, as well as ensuring that each orc was able to be animated. While procedural generation is often described as being stochastic, in reality it is not entirely so. The majority of the techniques used for procedural generation include some level of user control, as well as a specified input to use as a starting point for the equation. This input, known as a seed is transformed through code to create the output, in this case geological formations.

The main topic that this paper will cover is the mapping of geological formations. While this can be done through height maps, which are black and white, or colored maps where the colors or intensity of the black/white indicate the height of each point. This acts as a two dimensional representation of the terrain, although it has difficulty in representing structures that have undersides.

1.1 History

Procedural generation has a history rooted in math, particularly the work involving Brownian motion as well as fractals. Before the development of

other procedural generation techniques, some basic procedural generation techniques were used in the 1980s. One of the earliest usages of procedural generation was in Rogue. This initial attempt at generating a dungeon in a random manner addressed some of the differences between procedural generation and purely random generation. Rogue addressed this issue through procedural generation, using a three by three grid in order to generate the layout of the level, with hallways randomly connecting the rooms. These rooms would have a variable size in order to increase the variety of levels producible by the algorithm. This technique in particular, was created to address the memory constraints of computers at the time, as even with this more mathematical and less memory intensive approach, levels would need to be cleared from memory when moving on to the next one.

1.2 Overview and Definitions

Procedural generation is often dependent on the use of random numbers, which has varying methods of generation in a computer. There are two strategies for the generation of these random numbers, producing the numbers non-deterministically through the use of unpredictable physical processes, and computing numbers deterministically using an algorithm. While the determination of this random number is not important for procedural generation, as the necessity of security is less, procedural generation algorithms follow the same methodology as the deterministic random bit generators. In procedural generation, a seed is a random number where the rest of the generation will begin from. This ensures regularity to the procedural generation, by allowing for the algorithm used to have repeatable results. By taking an input seed, it will be modified using an equation to obtain the output, possibly using additional random numbers created after the original seed.

One of the applications of these random numbers is through the creation of noise. Noise in this application refers to the spreading of values across a coordinate plane. Lattice gradient noise is a commonly used and widespread form of noise. One of the most well known applications of this is Perlin noise, notably developed for Tron, but also applied in games like Minecraft. This

form of procedural generation works off of the premise of an integer lattice - a regularly spaced array of points in a square array. Pseudo-random numbers are generated and evenly distributed across this lattice, then a low-pass filter is applied in order to smooth the edges between each of these points. The low pass filter's effect is just creating the gradation between each of the lattice points. The result is an image such as this one.

Fractal landscapes is another approach at procedural generation. Fractal geometry being applied to landscapes began with studies of map data and research on the similarities between fractals and the data. Fractals prove as a relevant approach to representing geographical data due to the self similarity and the subdivision of space, made easier by fractal surfaces. This provides a method of predicting or as an initial hypothesis for landscapes in study. The technique works primarily from the subdivision of space combined with random numbers for each of the vertices created from the subdivision. Similarly to Perlin noise, this technique often utilizes a seed random number to start from, in order to make the results reproducible. However, while fractals are very computationally efficient, the parameter that makes this seed is very sensitive to change. Very small changes will completely change the features of a map designed this way.
Another method of generating terrain revolves around the use of cellular automata. define cellular automata. Cellular automata works on a grid of cells, similarly to how noise is mapped, but instead of having a continuous value, the cells in this approach only have binary values, or similar. This approach relies of having the information of neighbors and the states of their neighbors to determine surrounding cells to then modify their states. This can be used in conjunction with a tiling system in order to procedurally generate worlds, albeit with pre-determined cells for usage.

these are ways of representing the maps that are created

One of the ways of representing three-dimensional objects is through the use of polygon meshes. These are created through polygons (typically triangles) joined by at least two vertices. These polygons are then represented by the coordinates of the vertices that compose them, while the space between the vertices acts as the viewable portion. * not too sure what else to add? *

In contrast to polygons representing faces of a geometry, voxels represent a value on a in space. Voxels are represented by their position relative to other voxels, allowing for easier representation of layers of geometry.

2. Historical Usage

Perlin noise was developed for use in the movie industry, although it later became a foundation for many other procedural generation algorithms as it provided a lot of control as well as randomness to be used. It was developed in 1983 for use in the sci-fi movie Tron, to map textures onto computer generated surfaces for visual effects. Perlin noise has been used for many different visual elements, ranging from the texture creation it was created for to particle effects such as fire, smoke and clouds, as well as landscapes and geological features. It has a variety of uses due to its ability to create a naturalistic appearance.

3. Image Based Methods

Some of the other uses of height maps include the voxel space rendering system, using voxel raster graphics to display three-dimensional geometry with low memory and processing requirements. This was developed in the early 90's, involving a height and color map to position the pixels on the screen. While this technique was not historically used with noise generating algorithms, the rendering system fits the requirements for the use of techniques such as two-dimensional Perlin noise. At the time, displaying complex height-maps in three-dimensions was difficult computationally, and the voxel space technique allowed this to happen.

Talk about usage of height-maps, weaknesses
some weakness include - only 2 dimensional geological formations, so archways and caves are not possible
might be mitigatable by layering multiple different height maps, but increases computation time
example of practical application in rendering or pre generating voxel space heightmaps

4. Height Maps

As mentioned previously, Perlin noise can be used to generate height maps in order to pseudo-randomly create geological formations. While height-maps have the advantage of being two-dimensional and less complicated to run, they have difficulty in rendering a variety of geological formations. Due to the two-dimensional nature of a height map, they are unable to render more complex features such as alcoves, arches, and any other three-dimensional feature. In terms of randomness, this can create a sort of uniformity in geometry, where the only features are hills and valleys. This can be mitigated to some extent by the use of layering Perlin noise.

Implementation and how perlin noise works

While Perlin noise saw great success, it was succeeded by algorithms such as Simplex noise, designed to alleviate some of the problems with Perlin noise. This included the computational complexity and the artifacting in the noise created.

5. Polygons

5.1 Rendering Differences

Constructing polygons from height maps, similar problem

5.2

asd

6. Voxels

Voxels can be used in a lot of ways to render maps
Talk about minecraft's storage of voxel terrain data

Contrast with voxel rendering techniques

7. Development

test

7.1 Areas of Research

Possible areas of research?
There is deep learning for procedural generation, but it appears to not be very sophisticated - I cant tell any differences between the deep learning application of noise and just normal perlin noise

7.2 Algorithm Advancements

Talk about areas that have been advanced; i.e perlin noise vs simplex noise

8. Summary

asdasd

9. Appendix

asd

10. References

asd