Chapter 1

Challenges in Procedural Terrain Generation

1.1 Abstract

As demands for size and content in game worlds increases, procedurally created content offers an increase in productivity at the cost of control over the design process. This paper presents the results of a literature research in the field of procedural landscape generation. An introduction to terrain generation is given and current challenges in this field are examined. Where possible the advantages and disadvantages of proposed solutions are given.

1.2 Introduction

The increase in size and scope of games has enticed game developers to look for methods to increase productivity and efficiency at content generation. Generating content per a set of rules or an algorithm is an interesting solution. Landscapes, plants and buildings can all be generated by use of fractals, grammars or noise functions. The use of these techniques is currently not widespread as exercising a sufficient level of control over the generation of content so it can be used in games is often difficult. Additionally, generation of manmade scenery is only recently being explored. This paper will provide a summary of the current state of procedural content generation examining the different techniques for generating terrain, such as the diamond-square algorithm, heightmaps based on perlin noise and examine current challenges in artificial landscape generation.

1.3 Generating Landscapes Procedurally

Generating things procedurally means using a set of rules and algorithms to generate a lot of data with limited need for programmer input. This often includes smart usage of random number generation to reduce time spent by programmers working on the landscapes. Fractal landscapes and noise-based landscaped are often generated from very small amount of data - as small as a single integer seed for a random number generator. Random numbers are not always used, as can be seen in the .kkrieger game[.theprodukt GmbH, 2005]. Textures and levels are generated by building them at runtime using the artist’s directions. This does reduce the space needed to store a landscape but fails to improve productivity. In this section we will look at algorithms that attempt to generate terrain with constrained random number generation.

1.3.1 Generating landscapes: the diamond square algorithm

The diamond square algorithm is one of the simplest algorithms for generating heightmaps. It generates fractal terrain by means of a slightly modified midpoint displacement algorithm. The working of a midpoint displacement algorithm is shown in Figure 1.1. This algorithm is shown iteratively the figure. The algorithm starts with a line which is divided in two segments by taking the midpoint and displacing it randomly by an amount a. We then iteratively do the same for the two segments produced but displace the midpoints by a smaller factor, a/c, until a sufficient level of detail is reached. The constants a and c respectively denote the amplitude and the roughness. By varying these numbers different functions can be generated as desired.
In three dimensions the algorithm works by starting with four known points forming a square. This square is then subdivided into four smaller squares. The vertices that have to be added to form the new squares are displaced along the vertical axis similar to the midpoint displacement in 2d. First the middle vertex is displaced as part of the diamond step. Then, the vertices directly between existing points are displaced as part of the square step. The algorithm can, just like in 2d, be applied iteratively until a certain level of detail has been reached. Figure 1.2 shows the diamond-square algorithm in action in three dimensions.

1.3.2 Noise-based landscape generation

The division-based algorithm explained in the previous section is a good example of procedural content generation. The algorithm itself is very small but can be used to create worlds of vast sizes. It has several flaws however, most noticeably artifacts along major gridlines [Stanger, 2007]. In this subsection we take a look at several ways to heightmaps based on noise. Heightmaps are a representation of a three-dimensional world as could be generated by the diamond-square algorithm.
We take a look at fractal noise by taking two-dimensional perlin noise with turbulence as a basis. Perlin noise is generated in a number of steps. First, random numbers are generated along within a specific bandwidth by means of a seeded random number generator. Secondly, a function is generated by interpolating between these numbers using for example linear, sinusoidal or cubic interpolation. This is repeated with decreased amplitude of the function, limiting then random number generating function whilst increasing the frequency. This is repeated as much as needed to create enough detail. As the final step, all generated functions are summed, as can be seen in Figure 1.3. The summation of multiple noise functions is know as turbulence and is essential for realistic heightmaps [Bourke, 2000].

Perlin noise has a distinct advantages over midpoint displacement algorithms. First and foremost, perlin noise looks more natural since it lacks the square and diamond artifacts along major gridlines. Both noise-based heightmaps and fractal heightmaps have some degree of control over terrain features by adjusting the input for the first step. After generation of a heightmap often more realism can be added by simulating erosion or smoothing the noise. These extension are beyond the scope of this article as this paper only aims to provide a broad survey of procedural landscape generation.

1.3.3 Examples of games with procedural landscapes

Procedural techniques for generated landscapes are found within quite a few games, such as Spore and Darwinia. Spore features complete procedurally generated worlds, whilst Darwinia combines a premade part with generated content. Another way of using procedural terrain generation is exemplified by The Elder Scrolls IV: Oblivion in the form of procedurally assisted terrain. The artists create the landscape but procedural algorithms that simulate things such as erosion refine it.
1.4 Challenges

1.4.1 Lack of control

A problem with procedurally created content is a lack of control. On a global level, things such as roughness and elevation differences can be controlled. However, things like having a mountain range at a specific location in the landscape provide problems. Several techniques for additional control have been proposed.

Doran and Parberry proposed using agents in a separate pass to alter the results obtained by traditional procedural landscape generating algorithms Doran and Parberry [2010]. Multiple agents were created such as smoothing agents, mountain agents, beach agents and hill agents. These agents redesigned terrain based on parameters given by the programmer, thereby allowing generation of terrain with specific properties.

Another approach is using real height-maps as a template for new terrain synthesis, as presented by [Zhou et al., 2007]. This allows for placement of specific terrain features by fitting real-world data patched to a feature tree. Of course, often some work to adapt the landscape for a game world is still left after using all these tools and this will have to be done by hand.

1.4.2 Caves and overhangs

Caves and overhangs present a problem for most procedural terrain generating algorithms, since they usually generate a two-dimensional array height values as a result of two-dimensional functions, illustrated by Figure 1.4. Generating overhangs either requires an additional dimension or an additional pass after the generation of the heightmap. Some solutions using an additional dimension have been explored, such as generating an isosurface from three-dimensional perlin noise. [Gamito and Musgrave, 2001] present a general solution of generating overhangs by moving points on pre-generated heightmaps along a vector field. This solution has as advantage that it does not need to be adapted to new heightmap generating techniques and generates terrain that is completely connected and cannot intersect with itself.

\[ f(x) \]

\[ x \]

\[ ? \]

\[ x \]

Figure 1.4: Generating an overhang or cave is a problem for most traditional algorithms. Image from [Gamito and Musgrave, 2001]
1.4.3 Urban environments, road networks and other unnatural structures

The algorithms shown in the previous section work fairly well for generation of terrain, but generating man-made structures requires a different toolset. Cities are inherently complex and are often a result of multiple centuries of work. The layout of a city is influenced by things such as geographical factors, population and the level of technological advancement. As of the past decade, procedural generation of urban environments has seen steady advancement and several techniques have been presented.

Lindenmayer systems or L-systems were used by [Miller and Parish, 2001]. L-systems are grammars originally used to describe the growth of plants but were adapted by Parish and Miller to generate roadmaps and buildings. A result of their work is shown in Figure 1.5. L-grammars provide an efficient solution to generating cities and by modifying the grammar control can be exerted over the result [Kelly and McCabe, 2006].

![Figure 1.5: A street network generated in [Miller and Parish, 2001].](image)

[Watson et al., 2003] proposed an agent-based solution to building cities. Agents were responsible for creating and connecting roads and developing residential, commercial and industrial sectors of the city. This method has multiple advantages: The design process can be controlled by modifying the rules of the different types of agents. Additionally, the city is plausible at any point in the generation.

1.5 Conclusion

In this paper we examined several techniques for generating procedural landscapes. The diamond-square algorithm was used as an example and compared to heightmaps based on perlin noise. Perlin noise was shown to be superior due to lack of artefacts and better interpolation between values. In the challenges section we noticed three different problems: Lack of control over the generated landscape, the lack of caves and overhangs is a problem inherent of using two-dimensional functions to generate terrain and we briefly examined the creation of urban terrain using L-systems and agent systems. Certain problems such as the speed of algorithms were not discussed as this paper was meant to provide an introduction to generating procedural landscapes.

1.6 References


URL http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.2.1625&rep=rep1&type=pdf
   URL http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.91.8713&rep=rep1&type=pdf


   URL http://ccl.northwestern.edu/papers/ProceduralCityMod.pdf