Procedural Techniques for City Generation - Reading Report

Riry Pheng

McGill University, Montreal, Quebec, Canada H3A 2T5

Abstract. The increasing complexity in modern computer games forces the computer grame industry to hire a skilled workforce. By doing so, their production costs are extremely high. One of the most time consuming aspects is the creation of the virtual world inhabitting by the player [1]. Procedural techniques have been used in computer graphics to create natural textures, simulate special effects and generate complex natural models like trees and waterfalls. An overview of several procedural techniques including fractals, L-systems, Perlin noise, tiling systems and Voronoi cellular basis is provided. The function of each technique and the resulting output they create are discussed to better understand their characteristics. Finally, we will present how these techniques could be applied in the context of city maps generation.

1 Introduction

With the advancement of technologies, computer game consumers are more and more demanding in terms of graphic detail, realism and scale. The traditional approach to meet consumer demand was to increase the number of artists working on a project to produce larger, more detailed and realistic content. However, the amount of content generated is not proportional to the number of additional artists. The additional cost incurred is then added to the already high development costs, which is then paid by the consumers. Had they had better ways of generating content, all this time and money could have been allocated to improving game play or adding new features. Therefore, a potential solution for the content creation problem is the application of procedural techniques.

Procedural techniques have been used for over 20 years in the field of computer graphics for a wide range of applications: adding noise to existing textures, creating 3D textures of natural materials like wood and marble, visualizing life-like models of various tree and plant species, and generating detailed cellular textures such as skin and tree bark. Recently, procedural applications have been expanded further to simulate special effects including particle systems like fire, smoke, magic spell and glowing tails.

The key property of procedural generation is that it describes the entity, be it geometry, texture or effect, in terms of a sequence of generation instructions rather than as a static block of data. The instructions can then be called on when required to create instances of the asset and the description can be

parameterised to allow the generation of instances with varying characteristics. A typical example of this approach would be the population of a forest with procedurally generated unique trees.

In the domain of computer graphics, procedural techniques have been applied successfully in the generation of numerous complex entities and have proven beneficial for a number of reasons. Using procedural algorithms, textures, geometry or effects are not created at a fixed resolution or number of polygons. Procedural techniques are multi-resolution in nature, which allows the possibility of automatically generating models at multiple levels of detail. Concise descriptions for generated objects are possible and can often be expressed in the terms of a few simple parameters. These small descriptions can be used to create large amounts of detailed textures and geometry. This allows developers to create an entire game world that is easily distributable over low-bandwidth network connections. The flexibility and control provided by procedural techniques provide a mean to create new visual effects and original objects by experimenting with parameter values that exceed normal boundaries.

This reading report will describe several fundamental procedural techniques and algorithms that have been successfully used within the domain of computer graphics. Some of the discussed techniques include fractals, L-systems, Perlin noise, tiling systems and Voronoi cellular basis. We conclude by examining how procedural techniques are used in the context of city maps generation.

2 Fractals

Fractal is a word derived from the Latin term fractus, meaning broken. The basic concept of fractals is that they contain a large degree of self-similarity. This means that they usually contain little copies of themselves buried deep within the original, like the stars embedded in the Koch Snowflake shown in fig. 1. Moreover, fractal shapes possess infinite details such that for any given fractal, the closer we look at it, the more detail it can reveal. Fractals are created using recursive algorithms where each successive recursion yields more detailed versions of the basic shape.

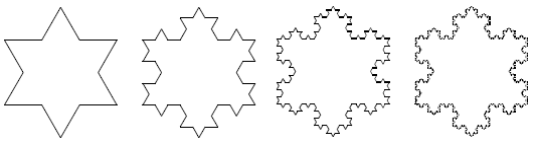

Fig. 1. First four iterations of the Koch Snowflake [1]

The Koch Snowflake in fig. 1 shows four such recursions. Self-similarity is achieved by generating the same shapes or patterns at smaller and smaller scales as the recursion progresses. Fractal algorithms are often used to generate a wide

range of natural structures from simple plants, like trees and ferns, to a whole terrain. However, fractals are limited to self-similar structures. Often, objects to be modeled may not necessarily contain this self-similarity property.

3 L-Systems

Formally known as Lindenmayer systems, L-systems are formal grammar invented by biologist A. Lindenmayer. They were originally developed to study bacteria replication and the growth patterns of simple organisms like algae. Over the years, L-systems have evolved and are now also applied in the field of computer graphics. They are used to generate fractals and model plants more realistically. The central concept of L-system is rewriting. Rewriting is a technique where given a set of rewriting rules; parts of a simple initial object are being successively replaced to create a complex object. L-systems are made of the following components:

- V : variables a set of symbols containing elements that can be replaced.
- S : constants a set of symbols containing elements that remain fixed.
- ω : start a string of symbols that define the initial state of the system.
- P : rules a set of rules defining the way variables can be replaced.

An initial state is provided. It is then rewritten using a series of rewriting rules P. These rules are applied iteratively, which allows complex objects to be defined using a simple set of rules. Fig. 2 shows how rules are applied in order to change the outputted string in the Thue-Morse system.

$V = \{a, b\}$	ω :	a
$\omega = a$	$n=1$:	ab
$P_1 : a \rightarrow ab$	$n=2$:	$abba$
$P_2 : b \rightarrow ba$	$n=3$:	$abbabaab$

Fig. 2. Thue-Morse system

L-systems were designed to define and visualize sophisticated plants and other natural structures. Significant advances have been made to the point that L-systems can now be used to generate rich landscape of detailed flora, covering a wide range of different species.

4 Perlin Noise

This technique was developed by Ken Perlin, used in the film Tron in 1982, in order to create more "natural looking" textures. Noise is created by first using a pseudo random function to generate a series of values which are then

interpolated into coherent noise. Several layers of this coherent noise are then combined together using different ratios to create a "natural looking" texture with fractal like detail. (i.e. infinite detail)

Random data are generated using a seeded noise function as shown in fig. 3. A seeded random generator will produce the same result when given the same input number or seed, but still produces numbers in a random pattern. Those data are then interpolated such that for a finite set of data points, a function can be generated allowing us to obtain an infinite range of points. Several algorithms are available to perform this interpolation. The simplest one is linear interpolation as depicted in fig. 4. It is often selected when speed is important and quality is of secondary importance. Results produced from interpolated noise have random properties but appear quite artificial rather than natural. To provide a more useful texture source that resembles nature more, turbulence is applied by combining several noise textures of differing scales (as shown in fig. 5).

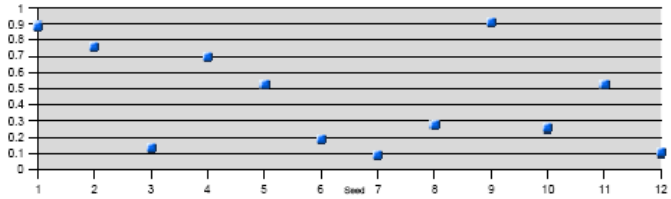

Fig. 3. Noise generated using a seeded random generator

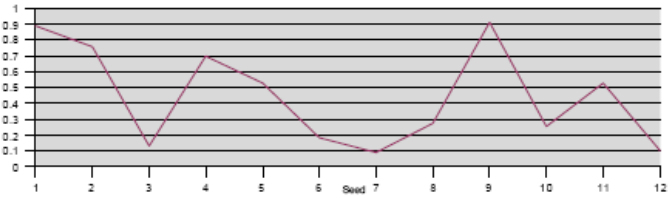

Fig. 4. Coherent noise created by interpolation of random data

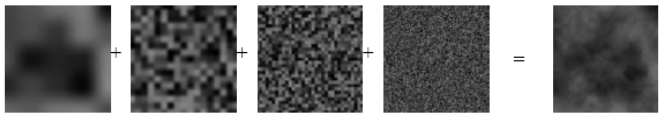

Fig. 5. Turbulence created by combining several layers of noise

Fig. 6 showcases the details and scale of output that can be achieved by using the Terragen procedural generation software, which uses Perlin Noise algorithm to generate photo realistic terrain, clouds, and seas.

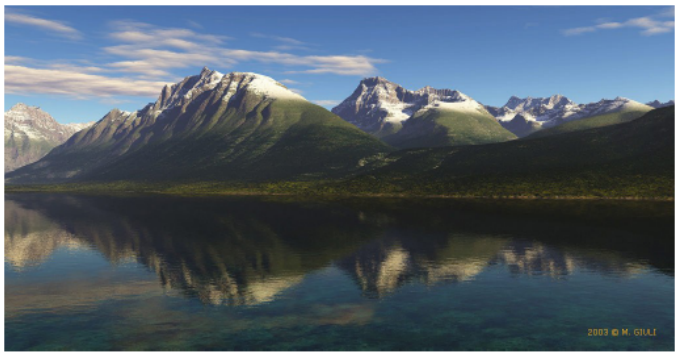

Fig. 6. Photo realistic scenery and rendered using Terragen with procedural geometry generation and procedural texturing. ©2003 M. GIULI Terragen Artist [1]

5 Tiling System

One of the most basic procedural techniques traditionally used in game development is tiling. It has been used in many classic games like Sonic, Mario and R-Type [1]. Originally, tiling consisted in creating small sections of 2D graphics that could be repeated on the screen and assembled together to create the virtual world. More recently, tiling techniques have evolved and are used in the form of multi-texturing. This consists in layering several layers of base textures in order to create highly detailed and varied textures. Using this technique, terrain can be procedurally textured by applying several layers of detailed tile-able textures. Some texture layer examples include rock, grass, sand and snow.

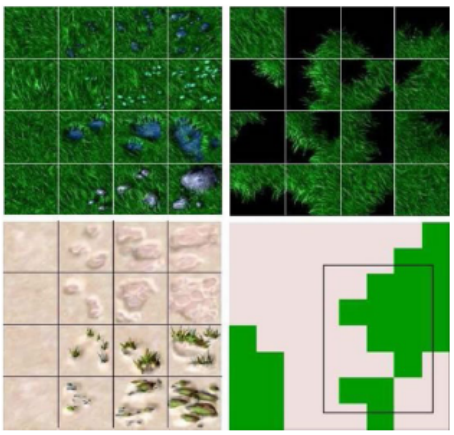

Fig. 7. Warcraft ®III uses stochastic information to procedurally generate Textures. ©2002 Blizzard Entertainment [1]

Extended algorithms exist that use stochastic information such as probability distribution maps to procedurally texture landscape. As depicted in fig. 7 (fourth image), an image map for the terrain area is supplied. This map stores the probability of using various tiles. Constraints can be specified to state which tiles can be joined under what conditions and whether they may be joined directly or require transitional tiles. By using a pseudo random function, thousands of different permutations of worlds are possible from a single probability map. Each possible world can be stored and recalled by simply taking note of the seed used to create the world [1].

6 Voronoi Texture Basis

In the paper titled 'A Cellular Texture Basis Function', S. Worley demonstrated that Voronoi diagrams could be used as way to generate content procedurally. The paper discusses a detailed algorithm that partitions space into a random array of cells creating cellular looking textures. The technique was devised to complement existing procedural techniques such as Perlin noise. It also provides a way to generate cellular surfaces like skin and tree bark procedurally.

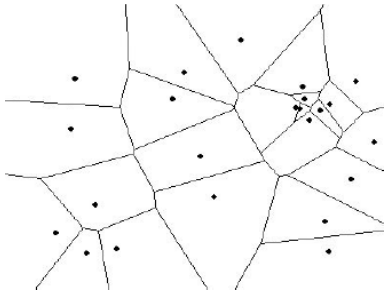

Fig. 8. Voronoi Diagram with coloured cells [1]

A Voronoi diagram is the decomposition of some metric space determined by distances to a specific discrete set of objects in the space [1]. Fig. 8 shows an area partitioned into cell by lines which are plotted using the points on the map. Each boundary line is positioned equidistant between each pair of neighbouring points. The resulting Voronoi diagram is a result of the position of the original points. A wide range of cellular patterns can be created by using different configurations to place the points used to create the diagram. Fig. 9 shows examples of Worley's algorithm applied to procedurally generated natural textues.

7 Conclusion

The procedural methods outlined in this paper have largely been applied to the generation of natural objects and textures. Only recently researchers turned

Fig. 9. Photo-realistic surfaces procedurally created using Worely's cellular basis algorithm. [3][2]

their attention to their application in the context of generation of man-made phenomena such as urban area. It is in our interest to examine how procedural techniques could be applied in the context of city generation. City generation is achieved through a series of stages that each use a number of techniques to create roads, lots, building structures and building faces. Road networks are a key aspect of city character and identity. When viewing road networks from a map, a number of patterns can be observed. It is these patterns that are key for procedural generation as they encode the structure of the road network. There are numerous road network patterns deployed in cities ranging from the tightly structured grid plan with perpendicular roads in a regular chequerboard structure to the hierarchical network with sprawling secondary and tertiary roads feeding into arterial roads in a branch like system. The observed patterns are the results of numerous factors including location, geography, cultural influences, planning trends, etc., such that cities can be categorised by their road patterns. It is in our interest to explore and write rules to generate road systems in the context of city generation using AToM3 and MoTif, which respectively are a tool for multi-formalism meta-modelling and a programmed graph transformation language.

References

1. George Kelly, Hugh McCabe: A Survey of Procedural Techniques for City Generation ITB Journal **Issue: 14** (Dec 2006)
2. David S. Ebert, F Kenton Musgrave, Darwyn Peachy, Ken Perlin, Steven Worley: Texturing & Modelling - A Procedural Approach Morgan Kaufmann (2006)
3. DarkTree Software: Cellular Texture Basis Functions implementation DarkTree procedural software (2006)