Random Terrain Generation and Visualization of Real World Locations
By Yale Zhang

1 Abstract

Computer terrain generation refers to building models of the earth's physical features such as mountains, plains, and seas, etc. Randomly generated terrain uses algorithms based on fractal methods to create models resembling natural terrain. The modeling of real world locations aspect of terrain generation involves combining various geospatial data such as elevation data and polygonal feature data to create three dimensional virtual representations of such locations.

2 Background

Virtual environment generation has been around for a while. The U.S. military has been using it for battlefield simulations since the 1980s and is currently building models of cities around the world as part of its combat preparedness program. This project will focus on developing more practical use for virtual environments such as for navigation. Random terrain can be very detailed and complex, but it becomes more and more repetitive as the area becomes larger. On the other hand, terrain based on real world data is limited by the resolution of the data source. One of the aspects of this project is to create realistic representations of real world environments by combining randomly generated terrain with terrain based on real world data; The randomly generated terrain is used to interpolate the terrain between areas where no recorded geospatial data exists.

The scope of the project will encompass research in the following areas:

1. Terrain generation methods
2. Computer Graphics
3. 3D Engine concepts
4. Geospatial data sources

3 Random Terrain Generation

Terrain has the property of being self similar. If you pick up a rock and look at at its texture, you will notice that it has roughly the same overall pattern as its parent rock formation when viewed closely. This is where the use of fractals comes in. Fractals are part of a relatively new branch of math known as chaos which seeks to describe complicated systems found in nature such as weather patterns that are not easily explained by traditional theories. Although the topic of fractals and chaos theory are beyond the scope of this project, many of the methods used to generate terrain employ simple fractal methods to produce such natural patterns.

So far, two algorithms for random terrain generation have been implemented in the project: the midpoint displacement method and the Perlin noise function.

Midpoint displacment

The midpoint displacement algorithm uses an iterative approach to produce random terrain by displacing the midpoint between two boundaries by a random amount and repeating the process for the left and right subdivisions recursively. The mathematics behind it is called Brownian motion. This concept is easily visualized in two dimensions as shown in the following pictures and can be easily extended into three dimensions.

In midpoint displacement algorithm in three dimensions is called the Diamond Square algorithm.

Perlin noise

Perlin noise is a continus random variable function developed by Ken Perlin in the 1980s. The basic idea behind it is to implement a function that returns a value that varies continuously as a function of its parameter. This is done by taking a discrete random number function such as C's rand(), who's parameter is a seed, and interpolating the values of the function evalulated at two points to get a continuous function. Depending on the method of interpolation, very complicated natural patterns can be produced.

Perlin noise can be implemented in N dimensions. A one dimensional function can be used to produce a random waveform which can be used as "noise" in sound effects. A 2D Perlin noise function can be used to generate heightmaps, which can look like clouds, by producing a height as a function of xy pairs.

4 Computer Graphics

(Until I make it less subjective, this section definitely doesn't belong in a research paper)

The Evolution of CG

Computer graphics has grown in complexity over the decades. Once only capable of drawing only lines and boxes, computers are now able to draw complex scenes of near photorealistic quality, thanks mostly to Moore's law.

The theories for CG have been devised and known since the 1960s. The first 3d graphics libraries were mostly proprietary tools developed by universities and companies such as Silicon Graphics. Then in the late 1980s, several standardized graphics library came into being.

RenderMan

The movie industry was one of the first major users of computer graphics. RenderMan is a high level graphics language that was originally designed to produce extremely complicated and photorealistic effects in movies. Practically every movie special effect has used RenderMan in some way (here's a list https://renderman.pixar.com/products/whatsrenderman/movies.html).

The liquid metal T1000 in Terminator 2 was a landmark in CG.

Sulley in Monsters Inc. is modeled with roughly two million computer generated hairs.

RenderMan is a minimal graphics language compared to many of today's graphics libraries. There is no concept of rendering details such as depth buffers or multi-texturing nor are there any limits such as the maximum number of lights. Instead, Renderman's most powerful features are its shaders. RenderMan shaders are essentially C functions that give CG artists endless user programmability in drawing things. Instead of implementing fixed function lighting functions or texture combining modes, all rendering effects are implemented as shaders. As a result, RenderMan is very general purpose and has changed little in twenty years and continues to meet the

requirements of today's CG industry.

The following is an example of a shader which is used to draw an apple. Do not be mistaken to think that texture mapping was used. The entire surface of the apple except the geometry data is defined by this surface shader:

Despite popular belief, RenderMan is not a ray tracer (it is implementation dependent). Pixar's PRMan implementation uses the REYES (Re-

alistically Everything You Ever Saw) algorithm developed at Lucas Film's Industrial Light and Magic division (the CG branch of ILM eventually split and became Pixar in 1986) which draws the scene in a fraction of the time it takes to do ray tracing. Even with this algorithm, drawing RenderMan scenes takes mammoth amounts of computer power. *Toy Story*, the first completely computer generated film took 800,000 hours of computer time (50 hours real time) just to produce less than two hours of actual film. Industrial Light and Magic had pioneered the use of renderfarms, networks of hundreds or thousands of computers working on the drawing process, and currently has the second most amount of computing power in the world after the Defense Department.

OpenGL

First introduced in 1992 by Silicon Graphics, OpenGL has been widely used for high performance applications such as CAD and for scientific visualization as well as in interactive games. Unlike RenderMan, OpenGL is a more low level graphics language.

OpenGL lacks the user programmability of RenderMan but makes up for it in speed. OpenGL is able to draw in real time unlike RenderMan because OpenGL operations such as drawing a triangle or texture mapping can be done blazingly fast using specialized hardware.

The following are images plaques and descriptions taken from the OpenGL Red Book (the image quality might less than optimal because the images were scanned from the book).

A scene with objects rendered as wireframe models

The same scene with textures maps and shadows added

A dramatically lit and shadowed scene, with most of the surfaces textured. The iris is a polygonal model.

Direct3D vs. OpenGL

No 3D graphics discussion is complete without mentioning Direct3D. In my opinion, Direct3D and DirectX is the epitomy of Microsoft's desire to reinvent the wheel. Basically, Microsoft realized that Windows 95 wasn't suitable as gaming platform, so they decided to add APIs to make Windows a viable gaming platform. Just like how they ripped off Word from Word Perfect and Internet Explorer from NCSA Mosaic, they bought the rights to Rendermorphics, a 3D graphics library made by a British company. As always before adopting anything not theirs, it had to be "improved".

The first releases of Direct3D were barely workable. It was a pain to program with because of the use of low level structs. OpenGL was superior in almost every way. DirectX 6 was the first truly workable version and every succeeding version made marginal improvements. As of right now, DirectX 9 has evolved to a point that it's 3D API is esesentially the same as OpenGL. However, each version of DirectX since 6.0 is incompatible with earlier versions causing the need to have multiple run time support for each version. OpenGL, on the other hand, has always been backward compatible since version 1.0 and is supported on almost every computer platform there is; DirectX only is supported on Windows. However, Microsoft evangelists

claim that OpenGL is only theoretically portable and that DirectX is only theoretically not portable.

Direct3D does have an advantage over OpenGL in that it is more object oriented (you specify which context you want to render to) and has much better support for video hardware extensions. Game makers are even starting to stop supporting OpenGL renderers in favor of Direct3D ones (i.e. Half-Life 2) - Microsoft evengilists at work?

The Future

The current trend in CG today is the increasing emphasis on both user programmability and speed. RenderMan graphics might be very generic, but is not at all suitable for drawing interactive environments. Lower level graphics languages such as OpenGL offer real-time rendering but consist of mostly of a fixed function rendering pipeline. New shading languages such as NVIDIA's Cg, OpenGL shading language, and Direct3D's HLSL are all attempts to give more programmability to low level graphics libraries by introducing shaders that are able to run at interactive rates by using dedicated video hardware.

Sources

¡http://www.geocities.com/CollegePark/5323/1980.htm¿
¡http://www.accad.ohio-state.edu/ waynec/history/ti¿
¡http://www.accad.ohio-state.edu/ waynec/history/timeline.html¿

5 3D Engine Principles

Generating a description of a virtual environment (i.e. position of objects, light sources) doesn't necessarily allow interaction with it. To be able to travel through the environment, render the environment in an efficient way, and to interact with it, the environment data needs to be organized in a way to allow for efficient traversal. This is where a 3D engine comes in.

A 3D engine is a system of methods to handle such tasks such as drawing the world and handling user interactions (i.e. collision detection).

Visible Surface Determination

In a typical 3D environment, there might be thousands of objects. Determining which objects are visible and needs to be drawn has been a classic problem in the development of 3D engines.

The theories for visible surface determination of complex environments have been ound for a while but it was the computer games industry that first made widespread use of them in the 1990s. John Carmack of Id Software, who wrote the engines to the Doom and Quake games, used binary space partition trees partition trees and potential visibility sets in calculating the visible parts of a world. This approach is known as a cell based method, which basically divides the world up into cells and then organizes them hierarchically.

Although the cell based method can run very fast, the problem of this approach is that the environment becomes too static. It becomes nearly impossible to change the world without having to recalculate a lot of data. Another occlusion method is the portal engine, which allows the world to be more dynamic and requires less preprocessing, but which is more CPU intensive to render.

Bsp trees, oct trees, PVSs, and portals can all be generalized as scene graphs. As is true of any graph, a scene graph contains nodes that points to relevant data. To draw the scene, the graph is traversed to find out which nodes should be drawn.

Collision Detection

The

6 Geospatial Data

The following section will cover the use of geospatial data for this project. Most of the information here was acquired from www.vterrain.org.

Creating three dimensional representations of the real world places involves converting feature data (elevation, roads, ground cover, etc.) into 3D models. Geospatial data refers to any such data that describes Earth's features over a certain region.

The main challenge in this project is to combine the different data sets together, such as mapping street information onto terrain contours. Urban environments will probably be the hardest to model because of the density and overlap of so many features.

The chief player in the development of GIS (global...) is the United States Geographical Survey, a government organization that specializes in recording almost every type of geographical data of interest. The majority of the data collecting is usually done by individual states. Simliar organizations exist in other countries as well.

Here's a list of various data sources that can be used in producing virtual environments. The data sets are generally downloadable free of charge (tax dollars at work), except for data that is deemed to be critical to national security. More specifically, this refers to elevation and image data which come in several detail levels, with the most detailed versions not in public domain.

Elevation

DEM (Digital Elevation Model)

DEM is a series of files that contain a regularly spaced grid of elevation data for a region. The coverage for DEM is mostly for the United States. A typical coverage areas are —— for 1:24000 data sets and —— for 1:250000 sets. Recently, the USGS has been phasing out the DEM file format and for the newer, more generic, but much more complex SDTS file format, which is supposed to be suitable for storing all geospatial data, especially DEM data.

DLG (Digital Line Graph)

DLGs also contain elevation data but store it using polygons representing equal height contours. The advantage is that this method takes up much less space than a regularly spaced elevation grid for relatively flat terrain. Just like DEMs, the USGS has been integrating DLG files into SDTS data sets.

GTOPO30 and SRTM (Shuttle Radar Topography Mission)

These data sets have virtually global coverage. GTOPO30 elevation data is on a regularly spaced grid of 1km per point. SRTM is based on GTOPO30 data except that it's much more promising for terrain generation since it has much more accuracy (30m to 90m).

Image data

Generally, providing image data coverage for the entire globe is more difficult than doing so for elevation because a. It is dependent on the time of day and atmospheric conditions (clouds). b. The ammount of storage needed for detailed image data is astronomical even by today's standards. A typical USGS Digital orthoquadrangle, which provides 1 meter resolution for a 7 square km area takes up 150 megabytes.

Multi-Feature Data

TigerLine

TigerLine data sets primary contain information on every street in the United States as polygonal data. It also contains information on features such as rivers, political boundaries and landmarks. The file format is ASCII text.

Coordinate Systems/Map Projections

The most straightforward way to describe any arbitrary position on earth is to use spherical coordinates, namely a longitude, latitude pair because the earth is mostly spherical.

Even though longitude, latitude is the most universal way, there is the need to display a small detailed area onto a map, which means converting 3D coordinates into 2D, called a projection. There are several dozen map projections and this is beyond the scope of this paper.

In addition, to make an accurate measurement, the Earth cannot be assumed to be a sphere. Instead, an ellipsoid model of the earth is used. There are many ellipsoid models (called datums) of the Earth, but they are all characterized mathematically by the length of the major and minor axis. Often, two coordinates that are the same in a coordinate system might not correspond to the same physical location on Earth because the coordinates could be based on different datums.

Here's a list of common coordinate systems

Geographical: (longitude, latitude)

This is the simplest and most universal system and has been used for hundreds of years. This is especially favored by sailors because the coverage is global.

Universal Transverse Mercator

This system uses a mercator projection based on projecting a 6 degree verticle slice of the earth onto a cylinder. Conceptually, this coordinate system lets you imagine the surface of the Earth as a flat sheet within certain bounds. The coordinate pairs called easting, and northing therefore correspond to points in a cartesian plane.

7 Building the Environment

Originally, my project was for both random and real world generation, but due to the difficulty of learning L-systems, fractals, etc. needed for working with random world generation, the emphasis is now on building a world from geospatial data.

Terrain

Building the terrain requires knowing the following characteristics:

1. Elevation
2. Texture

There are two types of terrain images: geospecific and geotypical. Geospecific images as it sounds are taken from a specificlocation and captures specific and sometimes unwanted information such as the time of day. Geotypical textures are generic images that are free from specific artifacts that can be put together to make up an environment.

Cultural Features

Roads - From my research, it seems that there are no documented methods for generating three dimensional roads from vector data. Almost all methods are commercial proprietary ones which basically stich the roads into the terrain (vterrain.org). My method is very simple which just puts the two end points onto the terrain.

8 References

Abrash, Michael. Inside Quake: Visible-Surface Determination. Accessed 10/30/2003. Available at http://www.gamedev.net/reference/articles/article981.asp

Avi Bar-Zeev. Scenegraphs: Past, Present and Future. Accessed 10/29/2003. Available at http://www.realityprime.com/scenegraph.php

Bourke, Paul. Fractal Landscapes. Accessed 10/30/2003. Available at http://astronomy.swin.edu.au/ pbourke/terrain/frachill/

Fractal Geometry: The Story of Benoit B. Mandelbrot and the Geometry of Chaos. Available at http://www.fractalwisdom.com/FractalWisdom/fractal.html

Laurila, Pietari. Geometry Culling in 3D Engines. Accessed 10/19/2003. Available at http://www.gamedev.net/reference/articles/article1212.asp

Ocean Waves. Accessed 10/14/2003. Available at http://www.naturewizard.com/tutorial02tem

Martz, Paul. Generating Random Fractal Terrain. Accessed 9/20/2003. Available at http://www.gameprogrammer.com/fractal.html

Midpoint displacement method. Available at http://www.redbrick.dcu.ie/ bolsh/thesis/node19.

Terrain Generation With Height Contours. Accessed 10/7/2003. Available at http://www.geocities.com/powersof2000/Prog_Papers/terrain1.html

Virtual Cities. Accessed 10/29/2003. Available at http://virtualcities.ida.org