3D Utopia

David M. Raber

April 29, 2004

Abstract

My techlab project will journey into the world of 3D graphics programming using the simple yet sophisticated language C++ and the OpenGL (Open Graphics Library). By the end of the year, I plan to have a fully functional 3D interactive world with which users can explore a complex environment. Although the project is still in a stage of infancy, I have high hopes for the effects that I can create and the amount of graphics programming I can learn over the course of this year.

1 Introduction

Computer graphics is a rapidly growing field in which many computer programmers find their passion. The technology provided by computers allows researchers and programmers to develop extensive simulations of the real world, which in turn can be reapplied to everyday situations. With computers, we can model the motion of water in a river and see what effects adding a dam to that river could have. We could use physics to simulate a car crash, and the results could help designers learn what potential problems exist and

solve them more quickly, efficiently, and safely. Computer graphics can save industries around the world billions of dollars by simulating situations without forcing companies to waste countless months or years testing their projects in the real world.

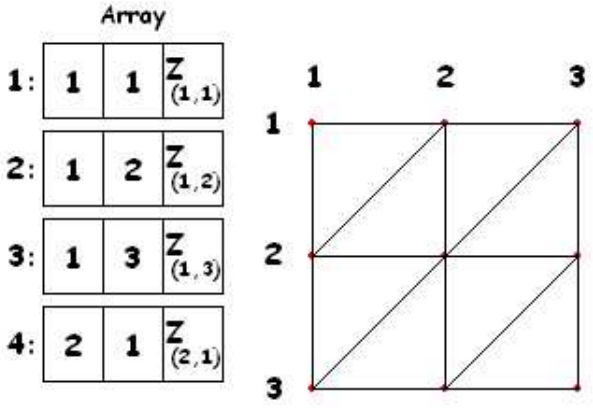
2 Background

Terrain generation has always been an important real world application of computer graphics. By using high-tech software, such as Terragen [2], computer programmers and researchers attempt to create realistic models of surrounding terrain. Terragen uses sophisticated algorithms to create the appearance of a real world environment for use in television, movies, and pure beauty.

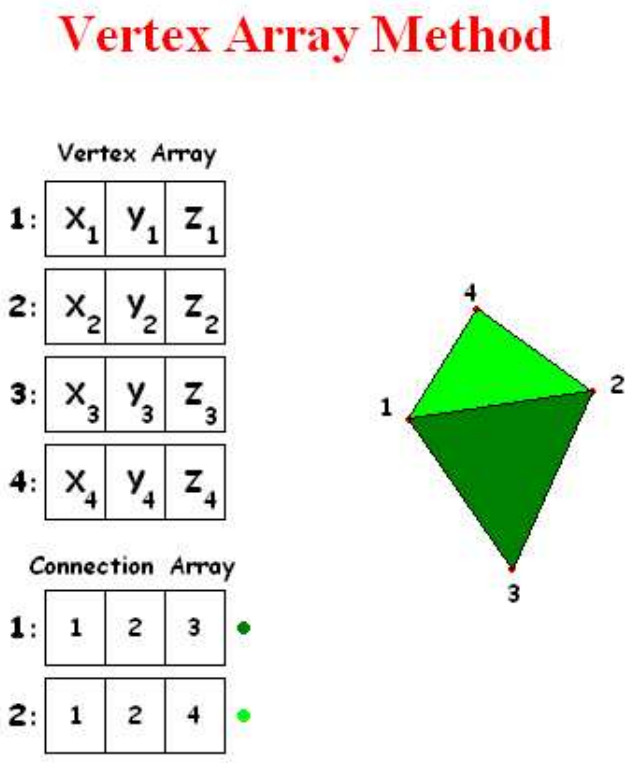
2.1 Simplification

Naturally, the algorithms used by Terragen are extremely complex and difficult to understand, so first let us explore a simpler algorithm for creating terrain. One method used by many beginning programmers is the height-mapping method. This method involves creating an array full of points and assigning each point a value for its height. Then, the program can connect the points to form triangles - creating the appearance of sloped terrain. The downside to this method is that it is not very flexible. Since coordinate points can only have integral (x,y) values, the model will not extremely realistic.

Square Array Method



One slight alteration of this method would be the vertex array method. This method entails making two arrays, one to hold all of the vertices used to map the terrain and the other which holds all of the various ways these vertices are connected. The second array could, for example, connect vertices 1, 2, and 3 to make a triangle in three-dimensional space. The disadvantage of this method is that usually more space will be required to store the data required to make the scene.



Each method has its own advantages and disadvantages, and both are used today to create realistic models of terrain.

2.2 OpenGL v. DirectX

OpenGL is commonly used in C or C++ programs to produce graphical models of objects of various problems in the real world. The graphics library OpenGL is used by many commercial businesses, especially those in the computer game industry. The biggest advantage of OpenGL is its ability to port easily from one type of system to another. OpenGL's cross-platform capabilities make it an ideal selection for gaming industries that want their programs to be available to all computer users, rather than just Windows users or Linux users. The downside, however, is that OpenGL lacks some advanced capabilities that other libraries offer.

DirectX is a Windows based library written by Microsoft for the purpose of graphical applications. DirectX is implemented into many games for its high-end three-dimensional simulation capabilities. One advantage DirectX has over OpenGL is that most applications written in DirectX code run faster than those written in OpenGL code for Windows. Since DirectX cannot be used on other platforms, however, the library is not optimal for those who want to work on two different operating systems.

3 The Mathematics of Terrain Generation

3.1 Lighting

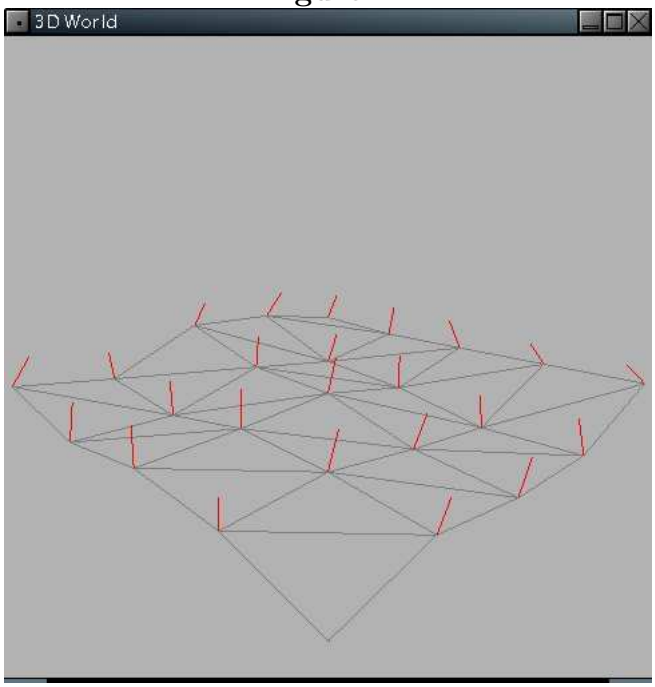
Lighting in terrain generation applications is important if a realstic simulation is desired. Terrain in the real world looks different during the day than it looks at night, and terrain in shadow appears darker than terrain in direct sunlight. In order to understand how programmers tackle the problem of realistic lighting in their simulations, one must first understand the following terms.

1. **Plane** - The set of all points perpendicular to a single vector. It can be expressed as the equation $Ax + By + Cz = D$.

2. **Normal** - The vector perpendicular to a plane. It can be expressed as $< A, B, C >$ from the above equation.

3. **Vertex Normal** - The vector that is the normal for a vertex rather than a plane. To calculate this vector, we add the normal vectors of the surrounding planes. See **Figure 1**.

Figure 1



This image is a picture of the terrain drawn in wireframe. The red lines are the normal vectors for each vertex.

References

[1] I would like to thank Mr. Hyatt for his great tutorial on LaTex which I will use to create this paper.

[2] Terragen is a terrain modelling program that looks quite interesting in many aspects and could prove useful in the future. The address is http://www.planetside.co.uk/terragen/

[3] NeHe's tutorial site is what introduced me to advanced OpenGL programming. His tutorials are extremely helpful and comprehensive. The address is http://nehe.gamedev.net

[4] Another important programmer whose work influeced this paper is Dragonslayer at http://www.dragonslayer.dk/gamedev/gamedev.html. His website taught me several things about octrees and frustum culling that were important to understanding the intricacies of terrain generation.