TJHSST Computer Systems Lab Senior Research Project
Use of Various Techniques Implementing Procedural Generation and Rendering in Video Game Design
2007-2008

Justin Warfield

April 4, 2008

Abstract

The goal of this project is to create a three dimensional game in which most of the games content is generated procedurally (as the game is played). Referential transparency and local randomization are important for a realistic but reliable procedural generation algorithm. Such techniques as fractal geometry, multivariable algebra, and statistical analysis can be utilized to generate terrain, texture, character models and render efficiently and effectively to create a more fulfilling, possibly endless game.

Keywords: procedural generation, midpoint displacement, referential transparency, procedural modeling

1 Introduction

Exploration into the topic will hopefully expand techniques of procedural generation into new areas. Common applications of random generation techniques will also be implemented, such as generating random 3D terrain and textures using fractal geometry for local detail and exploring the possiblity of multivariable functions for referential transparency. Results will hopefully allow procedural generation to be used in a larger variety of circumstances, allowing for larger and more player-friendly video games. Procedural generation techniques will be applied to a 3D lion killing game. After initial development of the game, algorithms will be added to generate terrain and background procedurally. Later, as time allows, a variety of more complex implementations of procedural generation may be pursued, such as creation of enemies, obstacles, items, etc. The exact nature of more sophisticated investigations cannot be determined until more research is done on current generation abilities.

2 Background

Many techniques are out there for creating random terrain, which seems to be the most common use of procedural generation. Fractal geometry is widely used in such algorithms. Similar techniques are commonly used to create random textures, such as cloudy skies and ground. The unreleased game, Spore, is expected to be groundbreaking in the area of procedural

generation, using procedural algorithms to create 3D creature models and animations. The use of 3D equations to model terrain seems to be seldom use and research is lacking, but the speed and potential of terrain functions has drawn me to the use of multi-variable equations. Hopefully my program will be completed implementing procedural generation in a new way, paving the way for further testing and experimentation. As of now, most of my research energies have been spent learning OpenGL and its GLUT library, but Ive found a few articles on gameprogrammer.com and the Intel website.

3 Development

Using OpenGL and Python (and the OpenGL binding for Python, PyOpenGL), the first step was to create a generic 3D game. A basic framework has been created for an interactive game environment with enemies and bombs. After attempting the use of fractal geometry to generate global terrain patterns, this has been determined to be too slow (almost 10 frames per second) to generate a minimum area. Multivariable equations may be better suited to generate terrain, with pseudo-random results. The use of an equation of order 1 to determine terrain would be much faster than current techniques. Testing wouldnt be clear cut for such a program. I need to make sure whats generated is both random and realistic, which are not easily quantifiable measurements. Human testing would be most effective. Examination of different terrain equations is done using a java program that can generate an interactive height map, and the height map included in the game. Realistic environments are being added, but not yet effectively.

4 Expected Results/Conclusions

In the end, even if my ultimate goals are not realized, I will at least have contributed minor tweaks and ideas to the field of procedural generation, and hopefully even applied procedural generation to an entirely new area, paving the way to a wider range of applications. Results can be presented in several ways. Comparison of various techniques would be effective. File size is also a good measurement for how much of a video game is procedurally generated (the smaller the file, the more game content is generated during game play). So far, lines of code and fps has shown that multi-variable equations are far

more efficient than geometric techniques. Use of a circular rendering area with decreasing detail with increasing area has been effective, and textures make the seams undectectable. Even if my contributions are minor and dont meet my expectations, hopefully they will build upon current techniques and allow later programmers to further build upon my findings.

4.1 Literature Cited/Appendices

Coming soon.