

TJHSST Computer Systems Lab Senior
Research Project
Use of Various Techniques Implementing
Procedural Generation in Video Game Design
2006-2007

Justin Warfield

October 31, 2007

Abstract

The goal of this project is to create a basic 3-dimensional video game utilizing several techniques (especially fractal geometry, multi-variable algebra, and statistical analysis) to procedurally generate terrain, obstacles, enemies, and other game objects, suited to best complement a users playing style.

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

1 Introduction

1.1 Scope of Study

Procedural generation techniques will be applied to a 3D camel-jousting 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 will 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.

1.2 Expected results

Exploration into the topic will hopefully expand techniques of procedural generation into new areas, such as the random generation of 3D characters and obstacles. Common applications of random generation techniques will also be implemented, such as generating random 3D terrain and textures using fractal geometry. Results will hopefully allow procedural generation to be used in a larger variety of circumstances, allowing for larger and more player-friendly video games.

1.3 Type of research

My research falls under the category of use-inspired research. My goal is to apply my procedural techniques in a working game, but I still need to understand it well to apply similar techniques in a variety of settings.

2 Background and review of current literature and research

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. 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 Procedures and Methodology

Using OpenGL and Python (and the OpenGL binding for Python, Py-OpenGL), the first step will be to create a generic 3D game (a camel jousting game). Hopefully the game will be functional by the end of first quarter, at

which time I will begin implementing algorithms in order to procedurally generate terrain, sky, and ground. This shouldn't take too much time, as much experimentation has already been done in these areas. My next goal would be to procedurally generate a variety of enemies. The algorithms must determine both a 3D model, as well as animations and game stats for each creature. So in the best situation, I should have plenty of time to try out various techniques for procedurally generating different things (like obstacles and items) and develop my own techniques. The ultimate goal is to create a game world generated entirely procedurally, as the game is run. The intermediate and more manageable goal is to utilize procedural techniques to create terrain and texture, along with one or two less conventional applications. Testing wouldn't be clear cut for such a program. I need to make sure what's generated is both random and realistic, which are not easily quantifiable measurements. Human testing would be most effective.

4 Expected Results

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). Even if my contributions are minor and don't meet my expectations, hopefully they will build upon current techniques and allow later programmers to further build upon my findings.