Procedural Generation and Terrain Rendering in a 3D Game

Justin Warfield- Period 5
TJHSST Computer Systems Lab 2007-2008

Abstract:

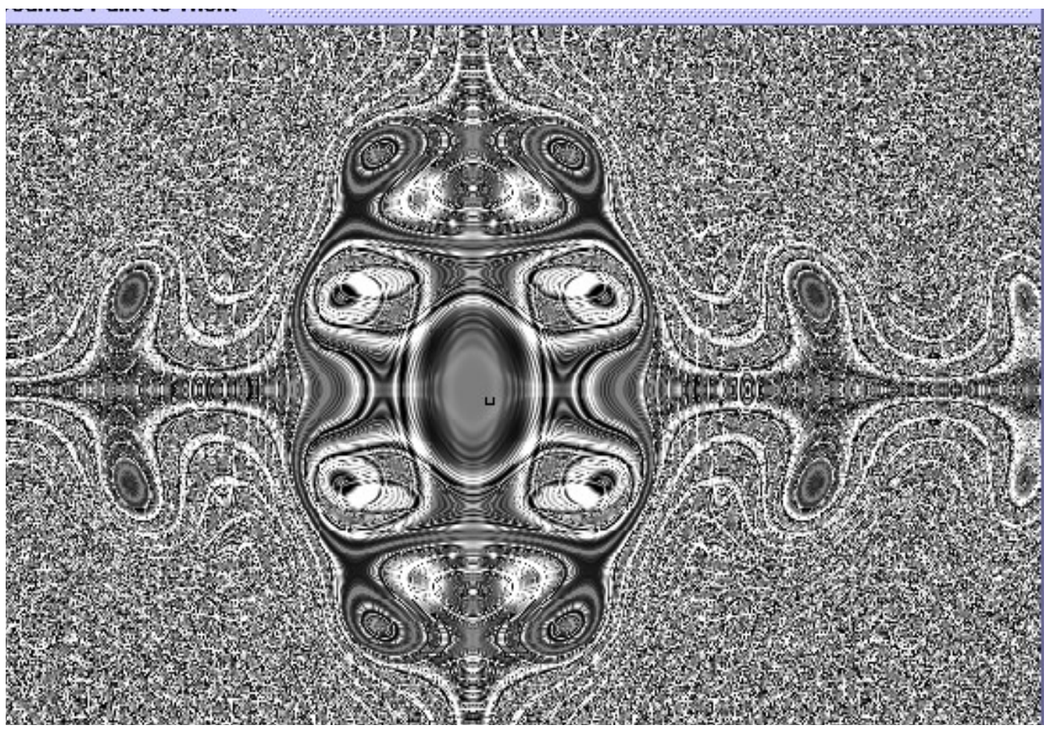
The goal of this project is to create a basic 3-dimensional video game utilizing several techniques (especially fractal geometry, multi-variable algebra, and statistical analysis) to procedurally generate terrain and game environment and render them in an efficient and effective manner.

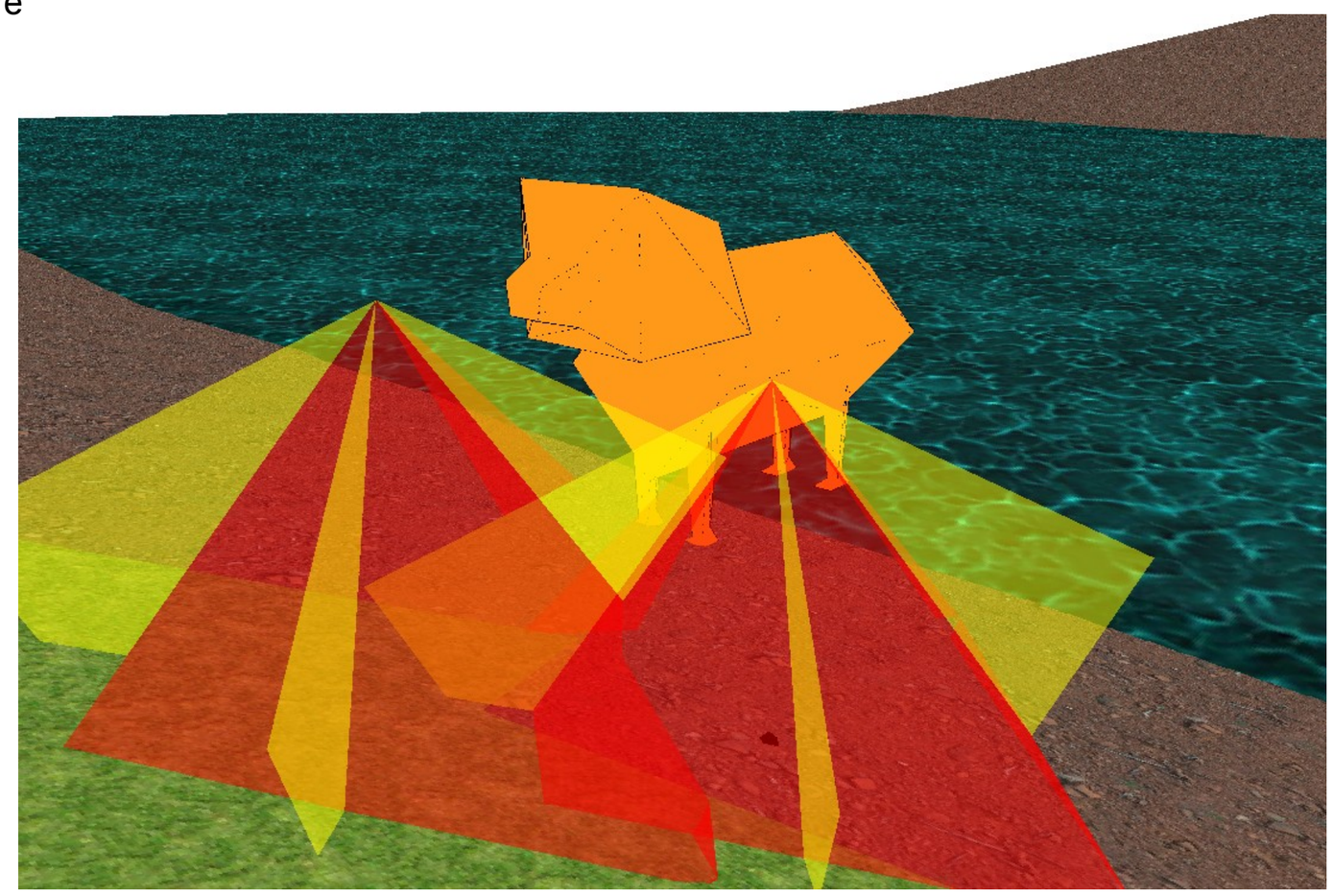
Procedures/Methods:

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 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. 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.

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 I've found a few articles on gameprogrammer.com and the Intel website.





Expected Results:

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.

Conclusion:

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. 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.