**Chirag Khatri**

**Kenny Wong**

**Matt Garnes**

**Computer Graphics Final Project Proposal: Perlin Noise**

**Team Name: “Perlin Noise All the Things”**

**Team Roles**

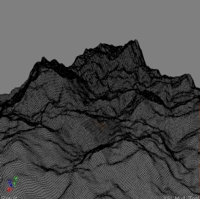
Chirag: Creating the perlin noise algorithm and adapting it for our needs when it comes to applying it.

Matt, Kenny: Research Perlin noise applications and choose a way to implement it.

**Vision and Idea**

Our vision is to implement a version of the Perlin noise algorithm that can be adapted, generalized and applied to several aspects of a demo that we will create. Some possible applications that we may include in our demo are:

* Terrain that appears pseudo randomly generated

 (DevMag Perlin Noise Tutorial)

* Textures for objects that are Perlin Noise generated

(Perlin Noise)

* Perlin noise Clouds (shape determined by Perlin noise)
* Scripts of movements of objects that are generated by Perlin noise (a flock of birds or insects)
* Realistic looking fire or water

We hope to combine some of these applications to create a simple scene or interactive game to show what Perlin noise can do.

**Specification of Requirements (Evaluation Criteria)**

1. Successfully implement Perlin noise and be able to consistently produce results that can be applied in multiple meaningful ways.
2. Create a demo to show what the Perlin noise algorithm can do. This demo should include the following:
   1. At least two of the Perlin noise applications from the Vision and Idea section.
   2. Some level of interaction. At the simplest level this should include a camera that can navigate and explore the scene we will create.

**High Level Architecture of the System**

For generating the noise (1D and 2D) we will use Python. For each noise generation, the script will output raw data which can be consumed for whatever use. For convenience, we will also make the script so that it is easy to change the persistence, frequency, and other variables to get a lot of variation out of our noise generation. In addition we will also be able generate textures with Python with the help of the Python Imaging Library (PIL).

For our demo, we will use XNA. XNA provides several useful features that will help us focus on the Perlin noise algorithm and its applications, allowing us to save some time when it comes to actually implementing the camera and scene rendering.

**Class Diagram (Tentative, does not reflect XNA behavior since we haven’t explored it yet)**

**Camera**

getPosition();

update();

**Interface PerlinObject**

draw();

getPosition();

**PerlinTerrain implements PerlinObject**

draw();

getPosition();

**PerlinSphere implements PerlinObject**

draw();

getPosition();

**Main**

initGraphics();

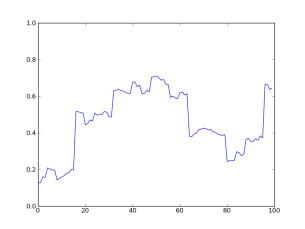
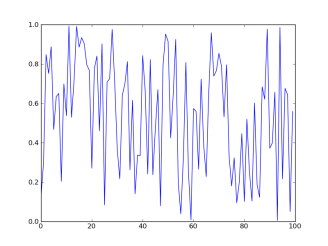
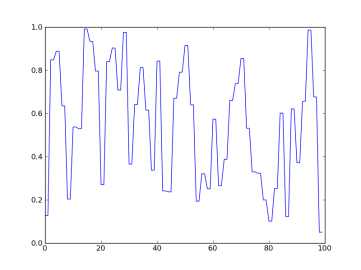
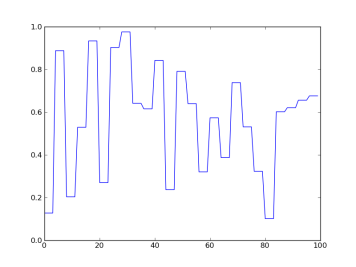
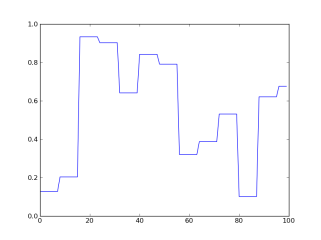
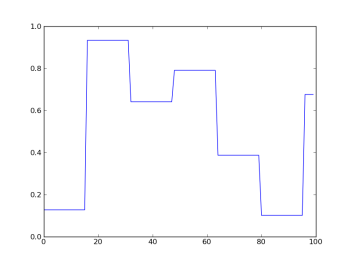
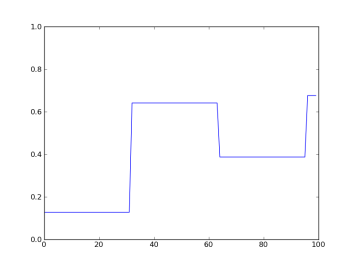
render();

**Specification of Process**

We will split up the work based on the roles given at the beginning of this document, Chirag will implement the Perlin Noise Algorithm and work on preparing methods to use it for different applications. Kenny and Matt will research and decide on applications for the Perlin noise algorithm and look into implementing them for the demo. They will report back to Chirag and work closely to communicate the requirements needed from the Perlin noise class in order to create the demo. Since the Perlin noise class will be fairly separate, integration should be fairly simple as long as the demo pieces that Kenny and Matt produce can fit together easily. We will use Github for version control and issue tracking since it is built in. We can keep track of milestones using Github as well.

**Progress So Far**

We have implemented 1D perlin noise with our python script and were able to plot the data for some visual feedback. Here are some of screen shots of the plots:



The first 6 images is the interpolated noise for the first 6 octaves. The last image is the sum of all of the octaves which gives us our perlin noise.