Project goals

Here is a list of my goals for this project:

• Use at least two, possibly three different fractal generation methods to create multiple landscapes. Some such methods include:
  o brownian motion
  o square diamond mid-point displacement
• Generate several fractal terrains and modify a few to make them look like rough landscapes.
• Simulated both clouds and a mountain range in at least one of the images.
• For one of my images, map a texture image onto the height map generated using fractals. In combination with OpenGL lighting, This should create a somewhat realistic looking landscape.
• Enable user to reproduce all of the above with the UI. So, user needs to be able to:
  o Generate height maps, choosing dimensions, algorithm, seed values, etc.
  o Add "special effects" to height maps. These include:
    • Add coloration based on height values
    • Anything I do to achieve landscape images

Methods for achieving goals

Before implementing any of the fractal terrain algorithms, I wrote code that would process, tesselate, and render height map data. Specifically, I made sure that I could take a two dimensional array of height data and do the following with it:

• render ground plane (floor from prev. projects)
• calculate (x, y, z) point for each position in the 2D height map array
• map the (x, y, z) points to (x', y, z') points that lie above the ground plane
• convert the new points into triangles with could be rendered a points, lines, or filled polygons
• assign colors to triangle vertices based on height (Y) values
• process height maps large enough to create a smooth, highly tesselated surface
I built a piece at a time until I had the basic infracture in place. As I added functionality I also added controls to the GUI so that useful changes could be made by the user without changing code.

My next step was to implement the two chosen fractal terrain map algorithms. I started with the diamond square algorithm since it was conceptually simpler. I first worked through a few iterations of the algorithm on paper to make sure that I understood it well enough to implement it. I then implemented it in two steps: the diamond step and the square step. I used debugging statements and my written example to work out the bugs. I proceeded similarly with the fBm algorithm. I added all the necessary inputs to the algorithms as user inputs on the GUI.

Once the algorithms were implements, I worked on adding extra features such as landscape coloring and texture mapping. As I developed interesting terrain maps, I took screen shots and have since added them to the web page for the project.

**Graphical User Interface**

The graphical user interface includes four groups of commands:

- **Camera** - Allows the user to move around the scene
- **Light** - Allows the user to turn the light on and off, as well as move it
- **Height map generation** - Enables both algorithms for terrain map generation
- **Effects** - Includes landscape coloring, texture mapping, and sample preset terrain maps

**Command Line**:

There are no command line arguments. The user interface handles things such as window-resize and a reasonable viewpoint.

To run the program, do as follows:

> openGL5

-OR- (if path variable is not set)

> ./openGL5

**Know Bugs/Problems**

- When smoothness constant is high for height maps generated using the diamond square algorithm, there are "dimples" in the 3D surface. I have re-examined my code for this algorithm, and it looks correct. I don't know if this is a bug or a normal artifact of the diamond square algorithm.
• Terrain map generation and rendering for highly tesselated surfaces is extremely slow. I've looked at the algorithms and the rendering code, and while there are optimizations that I could make, these are inherently slow methods.

List of outside resources

• I used the web page at http://www.gameprogrammer.com/fractal.html as a guide for implementing the diamond square midpoint algorithm. Note that although this web page supplies source code for a sample program, I did not look at or use any of the code provided, as I wanted to figure out the details of implementation myself.
• I used the book Game Programming Gems 2 as a guide for understanding and implement the fBm algorithm. This was the only clear explanation of the algorithm that I was able to find. Even then it took my a few passes to understand it.
• I used the following two websites as a glance at more advanced projects using fractal terrain maps:
  o http://wissrech.iam.unibonn.de/research/projects/gerstner/fractal.html
  o http://www.fractal-landscapes.co.uk/maths.html
• I used this web page to compare my results with the fBm method.
  o http://www.public.iastate.edu/~bthomp/homepage.html
• In order to find a standard interpolation function for random numbers, I used this website about interpolation functions and noise:
  o http://freespace.virgin.net/hugo.elias/models/m_perlin.htm

What I learned from this project

• Among other things, the meaning of life.
• Genuine mathematical principles can provide elegant solutions for real-world programming problems.
• Creating realistic landscapes is a complex task. Generating basic terrain maps is just the first step in this process.
• Once again, texture mapping appears the be the best option for achieving any kind of interactivity.
• Fractal brownian motion is actually used in the gaming industry.
• I gained a thorough understanding of both algorithms.
• I learned the difference between white and pink noise, as well as how to use bilinear interpolation to create pink noise.

Future improvements

• Test having multiple height maps concurrently (this might use up too much memory).
• Use some form of triangle stipping to reduce rendering time for terrain maps with high resolution.
• Allow texture maps of high-res terrain maps to be created and mapped onto low-res terrain maps that don't take as much rendering time.
• Do a great deal more with landscape simulation. My current landscape approximation is extremely rough and cartoonish.