Procedural Terrain

Sam Myers, Taylor Sheneman, Will Thorbecke

The Big Idea

We plan to create a program to procedurally generate and simulate 3-dimensional terrain. Our minimum viable product should be able to generate basic terrain made of cubes and allow the user to fly/navigate around it in 3-D. We hope to give the terrain the ability to "evolve" over time by simulating erosion and tectonic activity. Our stretch goals include:

- Rendering polygons instead of cubes for added realism
- Advanced weathering effects
- Cave generation
- Evidence of habitation (procedural city generation/evolution)
- Rendering water/waterflow
- Airplane (???)

Learning Goals

- Evolutionary (not genetic) algorithms
- Terrain generation algorithms (Perlin noise, diamond-square, fractals, etc.)
- 3-D graphics
- More OOP
- Computational optimization (making this run decently in Python)

Implementation Plan

- Panda3D or Pyglet for OpenGL rendering
- Frontend / Backend separation or Model-View-Controller setup
    - Generate terrain and run initial evolutionary algorithms in the backend as an array of 3D coordinates
    - Pass this to the graphical frontend as an array or dictionary to be rendered frame-by-frame
- Compartmentalize classes and functions in appropriately separate files for "plug-and-play" functionality

Project Schedule

- First 1-1.5 weeks: Parallel work on terrain generation and rendering framework
    - Research world generation, create a plan and some early code for a first-pass algorithm (Will and Taylor)
    - Create basic implementation of graphical engine (be able to process any array of coordinates, start with a flat grid or something) (Sam)
- End of second week: Implement a basic algorithm that demonstrably works (generate at least a height-map or something similar)
- Third week: Integrate rendering with terrain generation to create a very basic MVP
- Fourth week: Improve generation algorithms, start working on evolutionary effects, create blocks with different properties
- Fifth week: Finish block properties and evolution (fully functional implementation)
- Sixth week: Polish and optimize, try to implement polygons and/or "airplane" navigation

Collaboration Plan

- Begin with separate tasks (independently develop rendering and algorithms), then couple these parts together (during the first week, each of us will be working independently on research or framework)
- During the second week, Will and Taylor will collaborate to create a first-pass algorithm
- Third week forward: as we progress, we will collaborate on developing our models to a greater extent
- Possible pair programming for first-pass terrain generation algorithm
- Task management with Slack/Trello

Risks

- Evolutionary algorithms are complicated and unmanageable
- Performance is an issue
- Resolution doesn't reconcile well with cubes

Additional Course Content

- Computational efficiency