Background:
We're developing a program that procedurally generates worlds through the use of noise based fractal algorithms (more about this in a bit) in addition to evolutionary, macro-scale algorithms such as the modeling of tectonic plates, large-scale erosion, wind/precipitation patterns and Holdridge life zones, etc. Using these algorithms we will render a voxel-based world, with the hopes of eventually moving on to a gradient-colored mesh grid or something more advanced/pretty.

Problems:
Sam can't draw squares (update: Sam can draw squares now)
Tectonic plates are weird and hard (but we found a thesis and a library for it)
We have like 7 different algorithmic frameworks to implement: prioritizing implementation or ground-up construction from math
Height map doesn't allow for (damn pretty) caves or arches
Memory and processor constraints (we are representing/evolving an entire world with points)
Process

Harder Problems for the Future:
How do you combine fractal algorithms with evolutionary algorithms
Planes
Trains

Agenda:
1. Explain world generation process (2 min)
2. Talk about algorithms (2 min)
3. Talk about graphics (3 min)
4. Discuss order of applying algorithms, terrain generation design choices (10 min)
5. Discuss program architecture design choices (5 min)
6. Questions (3 min)