Noobwerkz platform : Report 17 May, 2015

Abstract: Engine progress has rapidly increased.

Details:

Since the last two weeks there has been, in sequential order:
- Reorganization of code to better reflect upon the needs of a bare-bones cross-platform graphics engine.
- Graphics engine is up to preliminary performance and flexibility standards (more on that later.)
- Added basic editor world core.
- Added PolyVox, a fast voxel (3D volume unit) library with features to extract 3D meshes from voxel data.
- Added Eigen (one of the fastest and most reliable solvers available, with extensions into many branches of mathematics and spinoff projects. Used a lot by academics, major projects, and software firms.)
- Added Noise++, the fastest noise library available. I intend on extending it in the future.
- Added Lemon Graph: The fastest, most solid, and most flexible graph library found. Graphs are very useful self-searching data structures that will serve as the backing of many features and optimizations.
- Added Voro++, a fast Voronoi-cell library. Good for many geometric needs. I intend on using it for efficient space partitioning.
- Added Rigid Body Dynamics Library - Fast simulation of linked body sims. Great for animation.
- Added Castor (logical programming)
- Added cppformat, a super-fast and safe text handling library

I also deviated a bit from the map and tested GUI rendering and it works well. Now, the goal is to use the aforementioned graph structures imported as a fast GUI communications system. User interface data gets fed into the graph, and searches are made when info is needed for any purpose.

A less successful experiment was trying to add Gecode (fast, solid constraint solver framework.) It will require some massaging to fit into the build system and its not immediately required. Also found a rendering bug for a different compiler than normally used (Clang.)

Another unsuccessful experiment was trying to compile many libraries separately and then trying to link them into the main app, in the mistaken belief that settings would hold properly. They do not, and only good alternative will be automated patching upon update.

Even worse, but still fortuitous, was finding out that the soft-body simulation relied on restricted code that I'll have to replace myself in order to use.

Future direction:

The dates discussed previously still hold, and no changes to the deadline are expected to be needed.

One pressing issue that was previously put on hold to favour development is the current computing setup, to be dealt with this weekend.

I had previously wanted to do things on FreeBSD as a main platform, but had heard reports of difficulties on the desktop. Now I have read that things have changed (they are a dynamic, well-organized community) and I wish to promote my FreeBSD server (with lots of disk space and many network adapters) to a more central role. It can be easily configured to boot copies of any OS across the network, has strong data-integrity protection guarantees, and very inexpensive snapshotting of data. It is well-documented and of robust design. All of these features make it an ideal OS for the role of a central management system, and adopting it should lead to strong increases in the reliability, productivity, and extendibility of my programming environment. The main drawback is that third-party binaries (and some source code, possibly) might not run, but there is good-performing Linux virtualization and "bare-metal" instances of Linux/Windows can serve as backups if needed.

To-do's upon completing OS migration:
Setup more hardware environments (Win, MacOS, iOS, Emscripten, NaCL)
Change the directory structure a little and rename a few files
GUI and Serialization
Skeletal animation
Test framework

Further on:
General constraint solvers - likely still Gecode
Functional programming