CI224-Semester Two - Project Brief

Overview

Team members

- George Vicarey
- Lord of pointers
- Warden of the Procedural Realms
- Richard Hankins
- Commander in chief of Null Terminators
- Master of Multiple Inheritance

Features we'd like to implement

BOLD items are definite, ITALIC items are markable but not necessary, un-formatted items are stretch goals

- **A cube world**
- *Procedurally generated terrain.*
- *Basic inventory system*
- **Entities**
- *Entities with pathfinding*
- *GUI/HUD*
- *Terrain has some form of chunk system*
- *Add and Remove blocks*
- Some form of networking
- **First person camera**
- Night and day cycle
- Pretty shaders
- Audio
- Load/Save game
- Game menu
- **Multi-platform**

Project Pitch

For this project we will be endeavouring to make a basic minecraft clone. It will be procedurally generated and contain at least 3 different types of textured block. The above list is a list of all the features we'd like to implement, although this doesn't necessarily mean we actually will. The mechanics we're definitely aiming to have finished are a procedurally generated block world saved using a

chunk type data structure, in game entities which will have a form of artificial intelligence which will allow them to use path finding to attack the player, and a playable player this means we want interaction with the environment including a basic inventory system and heads-up-display.

Our aim is to work on these three features until they are completely polished. We'd much rather have three working features that leave us with a playable game than implement every feature badly and have a game that's so bug ridden it needs an exterminator.

Project Plan

Our first goal will be to use both of our existing code from semester one to make a working basic cube world, in one codebase that we can then start working on our other features. At this point we will list the features we want to implement (which we already know) and break them down into tasks that need doing, if the scope of a single tasks looks to be too big we will break it down into sub tasks. Once we've broken down all the features into manageable tasks we can assign ourselves tasks to work on and implement individually.

Proposed technology

- C++
  Libraries we're likely to use:
- SDL2 - Opening a window and creatign OpenGL context
- SDL2_image - Loading an image to be bound as a texture
- libNoise - Generating noise to be used for procedural generation
- GLM - for ease of use when calculating vectors/matrices
- BulletPhysics - possible substitution to
- openGL
- Unit Testing
- Automated build tools
- Multi-User Git integration

Marking Scheme

The main aim:

Implement fewer features fully, rather than all features badly.

This marking scheme is divided into categories some of which have multiple criteria. some categories allow for multiple selections, i.e. whichever ones apply can count to final total, other categories are on a *one* only basis i.e. pick the highest marked item which applies.

Use of industry standard tools (any apply)

- Git - 5%
- Continuous Integration - 5%
- Build System - 5%
- Unit testing - 5%

Quality of code (any apply)

- Clean structure - 5%
- Modularity of engine/codebase - 10%
- Cross platform build - 10%

Working block world (only one applies)

- A basic block world (hard coded) - 5%
- A procedural block world - 10%
- A procedural block world with proper data structure - 20%

Working player (only one applies)

- A player that moves - 5%
- A player with HUD - 10%
- A player with interaction - 15%
- A player with interaction and HUD and inventory - 20%

Working entities (only one applies)

- Entities with basic AI (Just wander aimlessly) - 10%
- Entities with path finding and goals (can actively attack/find the player) - 20%