Build Systems

Options for build systems:

1. Visual Studio (VS). Probably the best option for ease of use as we are all willing and capable of using it and it has convenient features for this purpose such as a build configuration manager.

2. CMake. This may be the best option for cross-platform compatibility since the same builds can be used on Windows and Linux whereas VS projects differ slightly across platforms. VS has native support for CMake so we could write and build our code for both Windows and Linux inside VS.

Build configurations:

It's necessary to have multiple build configurations for different purposes such as debugging and final production. These configurations involve compiling different parts of the project for different purposes.

Debug: This build has all optimizations turned off, and all debugging features enabled. The result of this build will run more slowly than production, but will enable the developers to test newly-written code.

Release: This build retains some debugging components, but will be closer to the final experience.

Production: The final build. No debugging, all optimizations.

Tools: Tool builds are for the tools that go alongside the engine, such as the editor. For example, if we were to build a file converter for assets, compilation of this file converter will have different requirements than compilation of the editor.

Hybrid: Hybrid builds involve having debug builds for some parts of the project, and release builds for others. These are for testing individual components of the project.