There is a need for an easy to use, multiple platform, open source game engine.

New developers and smaller studios are often very limited due to the difficulty of learning the existing tools.

Many of the frameworks available today do not expose their source causing rigid development.

There is a need for an easy to use, multiple platform, open source game engine.

New developers and smaller studios are often very limited due to the difficulty of learning the existing tools.

Many of the frameworks available today do not expose their source causing rigid development.

The object model of many systems can often become highly complex, leading to unstable products.

The constant recompilation and high learning curve of some languages increases the overall difficulty.

As the market for games continues to grow more expansive, new developers and smaller studios are often very limited due to the difficulty of learning the existing tools.

Many of the frameworks available today do not expose their source causing rigid development.

A component based design was used in designing the game world in order to avoid excessive inheritance chains.

Both the game world and scripting also use the singleton pattern.

The game world maintains the record of all of the objects contained within the game world.

The component design of the object model reduces crowded object designs based on classical inheritance.

The scripting system has bindings that can directly invoke functionality spread across multiple subsystems.

Cmake was used to generate cross platform builds so that the binaries could be compiled for specific environments.

The engine was developed with C++ and the scripting engine allows the use of Lua.

A bottom-up approach was used to test the entire project.

Google Test was used for test automation.

Test demo projects were made to test all the requirements correctly.

I am thankful to the help that I received from my group members, and Peter Clarke.