Software Design Documentation

# Implementation

In order for you to use the game engine you must include all the source files in your project but will only need to add this line of code to your program.

**#include GameEngineCore.h**

From this you will be able to use all the components of the engine, from the graphics, to the terrain creator, to the physics engine, all can be accessed through the Game Engine Core.

# Graphics Engine

The engine was built to use OpenGL graphics API; however it was designed so that another graphics API could be implemented instead, like DirectX, as the graphics engine runs through an interface. This will require you to create your own class to call DirectX functions.

# Physics Engine

The engine was built to use Bullet Physics API; however it was designed so that another physics API could be implemented instead, as the physics engine runs through an interface. This will require you to create your own class to call another API’s functions.

# Terrain Engine

The terrain engine can build terrain from a height map file that loads into the project. It does this through a lua script so that no changes need to be made to the terrain engine to load in your own height map. Simple create a lua script that calls the generateTerrain() method, passing in the path to the file and the size of the image.

# Model Database

Adding your own models requires a little more work but can still be done quite easily. Firstly go the ModelManager.h source file and add the name of your models to eModels, like so

**enum eModels { mDog = 0, mGuard, mDoctor, mBuilding, mHouse, MODELS\_SIZE};**

Then in the ModelManager.cpp source file, in the init function write the path to your models like so

**mModels[mDog]->load(“models/german\_shepard/dog.obj”);**

# Design