Game Engine Design Document

Flying Panther Games

*Justin Pettit*

*Daniel Manganaro*

*Samual Lindsay*

## Installation

To install the game engine copy the following files into your game project directory:

* AI
* AssetManagement
* Collision
* Extras
* GameObjects
* Graphics
* Input
* LuaScripting
* AudioEngine.h
* AudioEngine.cpp
* Singletons.h

After copying, add these files to your project solution so that they may be called in your game.

To be able to access the game engine within your game project you need only add the following include.

include “Singletons.h”

From this you can access all the different components of the engine and add the elements of your game.

## Loading Assets

With the use of the assetManager you can load your models and textures, for example,

assetManager->(string FilePath);

## Playing Sounds

For our audio engine we used a third party sound library called IrrKlang which provided us the ability to play 2D and 3D sounds. This sound library was façade to only provide the basic functionality that was required.

With the use of audio you can play and stop 2D or 3D sounds and set the volume as well, for example,

audio->Play2D(const char\* filePath, bool loop);

audio->SetVolume(float volume);

audio->Stop(const char\* filePath);

## Creating Terrains

Create a terrain object first, for example,

TerrainModel\* m\_terrain;

After you have loaded a heightmap texture with the assetManager you can use this to load it into the terrain object.

m\_terrain->LoadHeightField(Texture\* file);

From this terrain object you can also set the scale of the terrain, add a detail map to it and add textures to the terrain and even create lighting for the terrain, for example,

m\_terrain->SetDetailMap(unsigned int tex);

m\_terrain->CreateProceduralTexture();

m\_terrain->CreateSlopeLighting();

To render the terrain you just need to calls its render function after it’s been set up.

m\_terrain->Render();

# Creating Skybox

Create a skybox object first, for example

Skybox m\_skybox;

You can load the textures of your skybox by using the following call.

m\_skybox.LoadTexture(int side, char\* filePath);

The size of the skybox can be set with the following function.

m\_skybox.Set(float x, float y, float z, float size);

To render the skybox you just need to call its render function after it has been set up.

m\_skybox.Render();

## Creating Game Objects

You can create game objects by first creating a GameObjectFactory instance, for example,

GameObjectFactory m\_objFactory;

From this you can create game objects, for example

objects m\_objects;

m\_objects = m\_objFactory.Create(string objType);

The object type can either be “player”, “npc”, “model” or “terrain”.

Once the object is create you can set its position, mesh, and bounding box with the following functions

m\_objects->SetPos(float x, float y, float z);

m\_objects->SetMesh(GameAsset\* mesh);

m\_objects->SetAABB();