CG Assignment 3 - Part 2 Report

Bradley Quinn - a1668522

Bradley.Quinn@student.adelaide.edu.au

June 10, 2017

My graphics application is the beginning of a game engine. For the most part I have implemented everything I said I was going to in the previous report.

It creates the terrain for a level from a grayscale heightmap. It can load and display models using textures, and draw copies of these models anywhere in the scene. It implements the three types of lighting (Directional, Point, Spot). It includes a day/night mode to showcase these lights. One interesting feature is text rendering, which is useful for debugging purposes. Shown Figure 1 is an example of the engine being run.

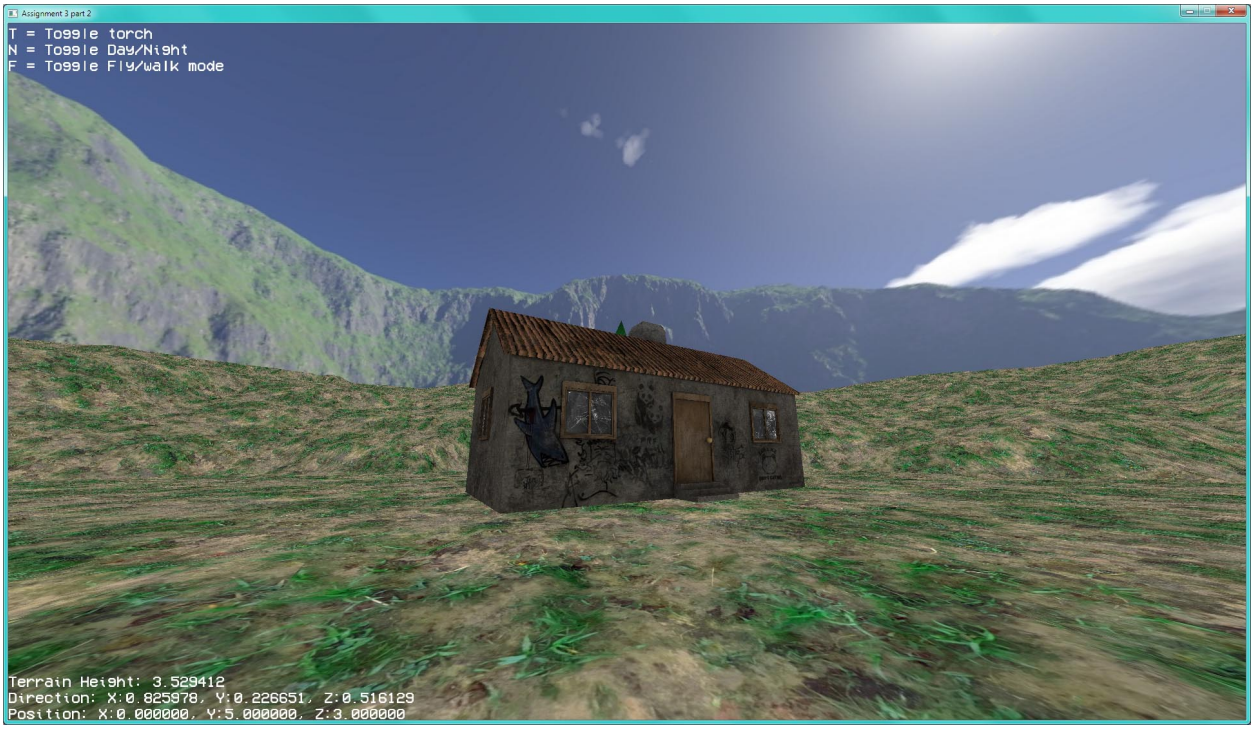

Figure 1: The game engine running in day mode

1 Readme

This is just extra information about compiling, the makefile handles all this.

For font rendering the application uses freetype2. The makefile determines if freetype libraries exist, if they don't it will still compile (just without text rendering). This has been tested on a linux machine with and without freetype and the program should still compile either way

If for some reason the makefile doesn't work you can still compile the application like you would any other OpenGL application (the code looks for the preprocessor definition -DUSEFREETYPE to determine which

section of code to compile with). If the system doesn't have freetype and you would like to compile it with freetype, install the library libfreetype6-dev.

2 Controls

If you compiled with fonts enabled, the controls are displayed in the top left of the screen. Otherwise, they are shown below

Mouse	Look around
W	Move forward
A	Strafe left
S	Move backward
D	Strafe right
T	Toggle torch light on/off
N	Toggle day/night
F	Toggle free flying mode and walking mode
ESC	Exit application

3 Implementation

In this section I will briefly detail everything I have implemented, and then in the following sections go into more detail about each feature.

Directional Light (1)	The sun/moon light
Point Light (1)	The lightpost opposite the house
Spot Light (1)	The torch feature
Loading Obj files (3)	Can load any number of models
Texture Mapping (1)	Loads textures from models
Height Mapping / geometry generation (2)	Using height-maps to generate the geometry for the level
Skybox (1)	Two skyboxes (day and night)
Multiple Shaders (2)	Three shaders in use, lighting, skybox and font
Collision (0.5 - 1)	Collision works with the floor terrain but not models yet

Font (1) Extra feature. Can draw any text to the screen

3.0.1 Directional Light

Not much to explain here, it uses the blinn phong lighting implemented in previous assignments. The sun and moon is displayed as a square model (placeholder). For night time the moon gives off less light, which makes the whole scene darker.

3.0.2 Point Light

There is a lamp post that has a point light above it. It is best seen in night mode. It also uses blinn phong lighting.

3.0.3 Spot light

By pressing T you can turn on a spot light that comes out from the direction of the camera. This simulates the player holding a torch.

3.0.4 Loading OBJ files

The program uses a slightly modified version of the model classes written for part 1 of this assignment. One main feature that I wrote was the ability to render models in different places and different scales, with only loading the model once.

In order to keep the distribution size small, only a few models are used, a house, a lamp post and a two trees. This is not a technical limitation.

3.1 Texture Mapping

This also uses the same code as the first part of the assignment, except I fixed the problem with texture sometimes not displaying properly. It was a very difficult bug to track down, the solution was to set the v of every texture coordinate to -v. The result of this is that textures always load in the correct way.

3.2 Height Mapping / geometry generation

This is sort of a combination of two parts, which is why I think it should be worth 2 marks rather than one. It loads in a grayscale bmp file, and uses the values (0-255) of each pixel to generate vertex data. This is

then scaled to whatever size it needed. Figure 2 shows the heightmap that is used in the currect version of the program. Whiter areas result in lower terrain and darker areas result in higher terrain.

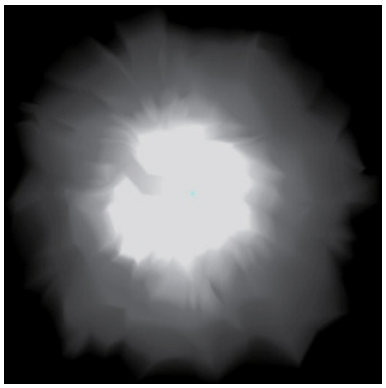

Figure 2: Heightmap image used

The terrain is then mapped with a grass texture.

3.3 Skybox

The Skybox is a fairly simple feature, although it does use its own shader. It helps to give everything a sense of scale. Two seperate skyboxes are used, depending on if the game is set to day or night mode.

3.4 Multiple Shaders

There are three separate shaders used in the program at all times. The first is the light shader, this is used for everything besides the skybox and the font rendering. It implements blinn-phong shading with all three lighting type explained above, as well as texturing.

The second shader is the skybox shader, which is just a texturing shader.

The third is the font shader. It is used because it ignores all other view matrices, to achieve the effect of drawing on top of everything else, and never moving in location.

3.5 Collision

The engine implements the starting point for collision. If you toggle to walking mode (pressing F), you will notice that the camera's height will stay consistent above the terrain. This is done by using the camera's current position to get the height of the terrain at that point (or at least the closest vertex, since the terrain is scaled).

Ideally there would also be collusion for models, but I ran out of time.

3.6 Font

This is a feature that I implemented that isn't on the list. It allowed me to debug values that were constantly changing, like the current location of the camera, this made it easy to place models in the world.

As it uses an external library FreeType, I designed it so the program can still compile if the library isn't installed on the system(it will just not display fonts). This is explained in detail in the Readme section. Figure 1 shows how the font rendering works if the system doesn't have freetype.