Technical Report

3D Graphics Module - GAV2004-N-BF1-2022

DEVINE, PATRICK (Student)

Teesside University  A0101165

Contents

[Features Of 3D Simulation 2](#_Toc124425449)

[User Guide 3](#_Toc124425450)

# Features Of 3D Simulation

A cube can be generated and rendered relatively easily thanks to the GenerateCube() function. This cube rotates and can be translated via input from the keyboard. In addition, the cube is created via techniques that utilize cube mapping, therefore each side can be rendered in different colours with the right textures being loaded.

Terrain is generated using a GenerateTerrain() function. The terrain has a diamond triangle pattern and can be easily altered by changing a few parameters around. A texture is being properly tiled across the terrain, also, the terrain makes use of a height map to decide the heights of the individual vertices. Noise may be applied by changing a few variables around in the GenerateTerrain() function and normal are calculated correctly.

A 3D model can be loaded, however, there is no current hierarchy class system and the transformations are applied somewhat manually. The textures are applied normally across the 3D model, the model however does not handle lighting properly and needs some more debugging so as to remove this error (lighting does work it just comes from the opposite direction than where it should be coming from). The rotation of the propellor is done not by using hierarchical transformations but applying a rotation after transforming to it’s required position in the model.

Models and terrain are Phong shaded, everything is included (specular light, diffuse light, attenuation etc.). Directional lighting (Lambert) is also functional, two lighting modes cannot be used at once at the moment (they can be changed in the creation of the program in the Renderer.cpp file CreateProgram()). Spotlights and point lights are fully functional and can be paired together well. There is a maximum of ten lights on the screen at once, there is currently no functionality to only use the ten closest lights.

A sky map is used and altered during the render of the scene, its near cut distance is set to 0.0f so the sky doesn’t get cut out of the scene, it is always drawn behind any other object as the depth checking is disabled for the skybox. A cube map is used to create the skybox and it is not affected by lighting.

All these features all work at once inside the application. In addition, the code does support multiple models and mesh however it requires some work to get another model working (adding another VAO, loading a new texture, adding a new object to the map etc.). Finally, all pointers and dynamic memory are deleted upon the program’s completion and or exit.

# User Guide

WASD moves the camera.

Holding left click (and moving mouse) changes the camera’s orientation.

The arrow keys translate the cube that represents a light around the scene.

Z and X increase/decrease the height of the cube that represents a light.

Space resets camera to original position and orientation.