Edinburgh Napier UNIVERSITY

Computer Graphics
Course work Part 1 - Report

Dimitar Hristov
40201757@live.napier.ac.uk
Edinburgh Napier University - Computer Graphics (SET08116)

Abstract

The aim of this project is to create a realistic 3D scene, rendered in real-time. The project is inspired by the series *Games of Thrones* and previous years projects found on the games website of Napier University. A wide variety of graphics techniques were used to create the 3D scene, from multiple lights and light types to shadowing, material shading and transform hierarchy.

Keywords – 3D scene, OpenGL, C++, GLSL, lighting, shadows, normal mapping, real time, phong

1 Introduction

Scene parts The project is meant to be visually intriguing and more importantly it is meant to demonstrate core understandings of Computer Graphics principles. The 3D scene involves:

- a miniature model of the Earth and the Moon, rotating around it;
- a wall and a spot light demonstrating shadows;
- a realistic dragon egg made with normal mapping;
- a model of a dragon next to the Earth, protecting its egg;
- geometry objects moved with hierarchical transformations (the dragon egg protectors);
- skybox that brings to the scene background and completeness;

Graphics effects The graphics effects implemented in this project include:

- multiple light types (directional, spot and point light);
- texturing and normal mapping that give high level of details;
- shadows that make the scene more realistic;

There are two types of cameras implemented within the project: *free and target camera*. The free camera allows the user to go around and explore the 3D scene and the four target cameras show the scene from four static points of view.

The exact implementations of these graphics techniques are discussed later in the report.

You should cite References like this: [1]. The references are saved in an external .bib file, and will automatically be added ot the bibliography at the end once cited.

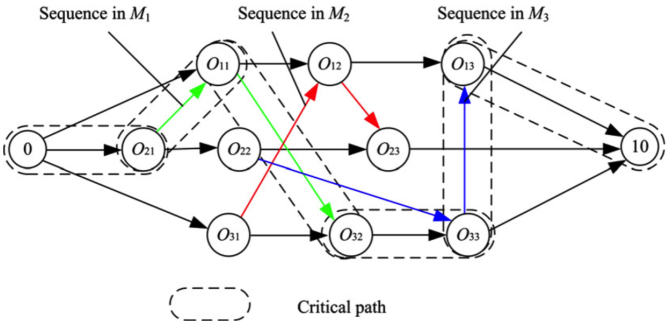

Figure 1: **ImageTitle** - Some Descriptive Text

2 Related Work

All of the techniques used in this project can be found in the workbook for the Computer Graphics module - SET08116 at Edinburgh Napier University. The required skills were developed during the practical sessions of the module. Some of graphics techniques had to be taken further in order to develop the final 3D scene for this project.

Some common formatting you may need uses these commands for **Bold Text**, *Italics*, and underlined.

2.1 LineBreaks

Here is a line

Here is a line followed by a double line break. This line is only one line break down from the above, Notice that latex can ignore this

We can force a break
with the break operator.

2.2 Maths

Embedding Maths is Latex's bread and butter

$$J = \left[\frac{\delta e}{\delta \theta_0}\frac{\delta e}{\delta \theta_1}\frac{\delta e}{\delta \theta_2}\right] = e_{current} - e_{target}$$

2.3 Code Listing

You can load segments of code from a file, or embed them directly.

Listing 1: Hello World! in c++

```
1 #include <iostream>
2
3 int main() {
4     std::cout << "Hello World!" << std::endl;
5     std::cin.get();
6     return 0;
7 }
```

Listing 2: Hello World! in python script

```
1 print "Hello World!"
```

2.4 PseudoCode

for $i = 0$ **to** 100 **do**
    print_number = true;
    if *i is divisible by 3* **then**
        print "Fizz";
        print_number = false;
    end
    if *i is divisible by 5* **then**
        print "Buzz";
        print_number = false;
    end
    if *print_number* **then**
        print i;
    end
    print a newline;
end

Algorithm 1: FizzBuzz

3 Implementation

There are a number of elements that are used together in order to make the scene alluring. These elements are:

3.1 Multiple lights

There are three types of lighting sources implemented in the project: directional, spot and point lights. The directional light is used to light the whole scene.

3.2 Texturing and Normal Mapping

Texturing is the process by which image data can be applied to geometry objects and models to provide mode details. Texturing a piece of geometry requires texture coordinates, which is a 2D vector. The texture function, that is used in the fragment shader, takes the texture image and the texture coordinates as arguments and returns a 4D vector that represent the colour for the current pixel.

$$vec4texColour = texture(tex, texCoord)(1)$$

Normal mapping is a technique that allows us to calculate the normals on a per-pixel level and gives a high level of detail to the objects. It gives the illusion that a flat mesh has depth on its surface by reacting with the light in the scene.

3.3 Shadows

Shadow mapping uses the depth buffer that captures depth information to allow us determine if an object is in shadow based on the light hitting the mesh. In this project the shadows are created with one spot light which is casting light on meshes which are rendered separately from the others. In order to create more realistic shadows, the projection matrix that is used with a wider field of view. The normal angle for the FOV is 45 and for the shadows it is changed to 90. Figure demonstrates the shadow casting in the project.

3.4 Moving objects

For the implementation of the moving objects the sin fuction was used. By using its main property (Figure) a the position of a mesh can be changed in a particular range. The sin function is used for the movement of the moon around the Earth, the torch in front of the wall and the levitating egg and protectors.

3.5 Hierarchical Transformations

Hierarchical Transformations

3.6 Skybox

Skybox

4 Conclusion

References

[1] S. Keshav, “How to read a paper,” *SIGCOMM Comput. Commun. Rev.*, vol. 37, pp. 83–84, July 2007.