Solar System Model

Dimitrios Tsolis
40204497@live.napier.ac.uk
Edinburgh Napier University - Module Title (SET08116)

Abstract

The aim of this project was to render a realistic scene using the skills developed in the Computer Graphics Module at Edinburgh Napier University using OpenGL and C++. The scene produced in this project is a solar system, inspired by my adoration for the stars and the Star Wars movies add reference later. Advanced 3D effects and techniques are used to accomplish the generated result, such as lighting,shadowing and texturing. These techniques are widely used in a variety of games.

Keywords – Shadowing, Lighting, Multi-Texturing, Skybox, Multiple Cameras, Material Shading

1 Introduction

Referencing 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.

The key effects being used are lighting, texturing, shadowing, skybox, material shading and transformation hierarchy. Various rendering techniques are required to produce a realistic depiction of the scene. The different lights being used in the scene are a spot light and a point light in the middle of the sun. Transformation techniques are also being applied to the different meshes of the scene, and each of the objects render have their own textures.Furthermore, in order to create the stars image around the solar system a skybox effect is being used. In order to make the earth more realistic, normal mapping is being implemented on the project.

Diffuse light is being applied to the scene, adding some depth perception to the objects, as well as specular light which adds shininess of materials to objects and changes depending on the camera position. The transformation and translating of the meshes in the scene is the scaling and the positioning of the objects. Texturing combined with the lighting is applied to give the objects a more realistic aesthetic. A more advanced technique being used is normal mapping, which uses the texture and a normal mapped version of the texture, which helps determines the direction the pixels are facing, giving an ectra feeling of depth to the object.

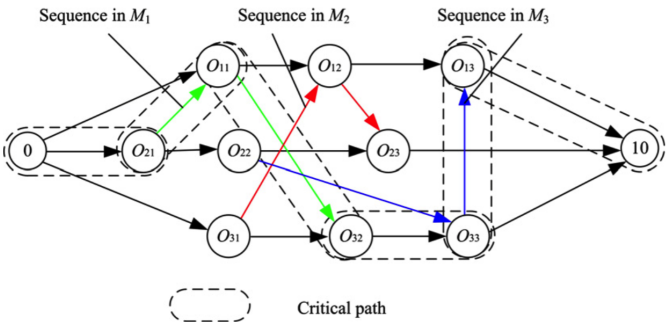

Figure 1: **ImageTitle** - Some Descriptive Text

2 Formatting

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 Conclusion

References

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