Edinburgh Napier UNIVERSITY

Computer Graphics
Coursework 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*[1] (see **Figure 1**) and previous years projects found on the games website[2] of Napier University. A wide variety of graphics techniques were used to create the 3D scene, from post-processing effects to shadowing, material shading, reflective and transparent water. This report covers how the scene was implemented and what future work is considered.

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

Figure 1: **Scenes used as inspiration**

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 and its' protectors);
- a skybox that brings completeness to the scene and also background;
- island terrain;
- instancing of 800 dragon eggs around the island;

Graphics effects The graphics effects implemented in this project include:

- multiple light types (directional, spot and point lights);
- texturing and normal mapping that give high level of details;
- shadows that make the scene more realistic;
- reflective and refractive water with implemented Fresnel effect, depth and distortion;
- post-processing effects, including mask, edge detection, sepia, motion blur and wire framing;

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.

Further information about these graphics techniques is given later in the report. **Figure 2** shows part of these elements and graphics effects.

2 Related Work

Part of the techniques used in this project can be found in the workbook for the Computer Graphics module - SET08116 at Edinburgh Napier University[3]. The required skills were developed during the practical

Figure 2: **Scene from the project**

sessions of the module. In addition to this, for the water effect an online tutorial[4] was followed and adapted for the purposes of this project. Some of the graphics techniques had to be taken further in order to develop the final 3D scene for this project.

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. They are essential to make the normal maps and shadows working. The directional light is used to illuminate the whole scene. There is one point light between the earth and the egg protectors that enlighten the objects around it when the directional light is turned off. Two of the spot lights are located in front of the earth and the boxes. Together with the normal maps they give the meshes a good realistic view. In addition to this, there is a spot light inside the torch that casts shadows of the objects on the wall and on the ground.
The Phong shading was used throughout the project. The Phong shading is a graphics technique that calculates light on a per-pixel rather than per-vertex level. It improves upon Gouraud shading and provides a better approximation of the shading of a smooth surface. On **Figure 3** is shown the difference between the Gouraud - per-vertex (left) and the Phong - per-pixel (right) shading.

Figure 4 represents the Phong equation where the light is white, the ambient and diffuse colours are both blue and the specular colour is white. The same properties are applied to the meshes in the project and used in the fragment shader when calculating the correct colours to be displayed.

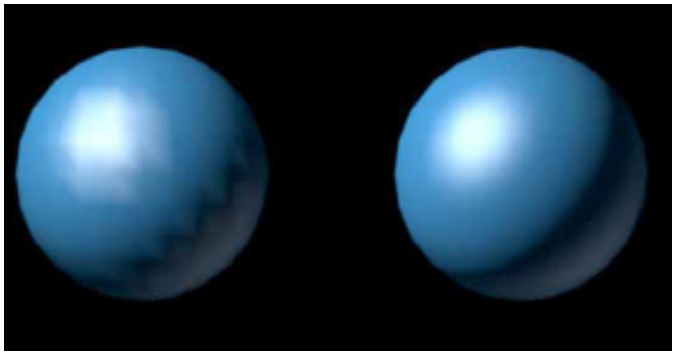

Figure 3: **Gouraud and Phong shading**

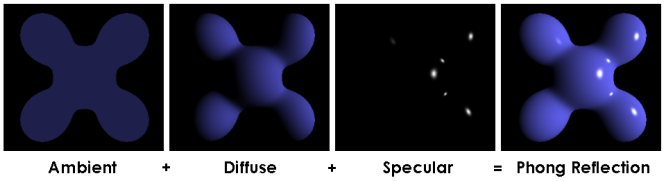

Figure 4: **Phong shading equation**

3.2 Texturing and Normal Mapping

Texturing is the process by which image data can be applied to geometry objects and models to provide more details.
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 creates the illusion that a flat mesh has depth on its surface by reacting with the light in the scene. In the project there are normal maps applied to the Earth and the dragon eggs. See **Figure 5** for references.

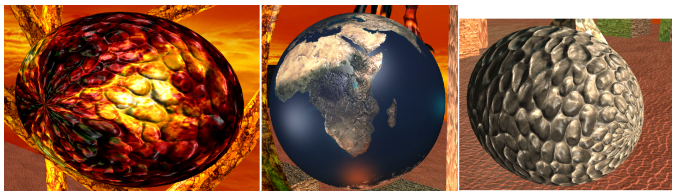

Figure 5: **Normal maps in the project**

This effect can be achieved with normal map images similar to the one on **Figure 6**. The RGB values of the texture represents the <x,y,z> components of the normal at this point. In addition to the normal, the bi-normal and the tangent are also required for the transformation matrix that is used to calculate the sampled normal. This normal map is used for the effect achieved on the two dragon eggs on **Figure 5**.

3.3 Shadows

Shadow mapping uses the depth buffer that captures depth information. This allows us to determine if an object is in shadow based on the light hitting the mesh. **Figure 7** shows an example of how the depth buffer is working to

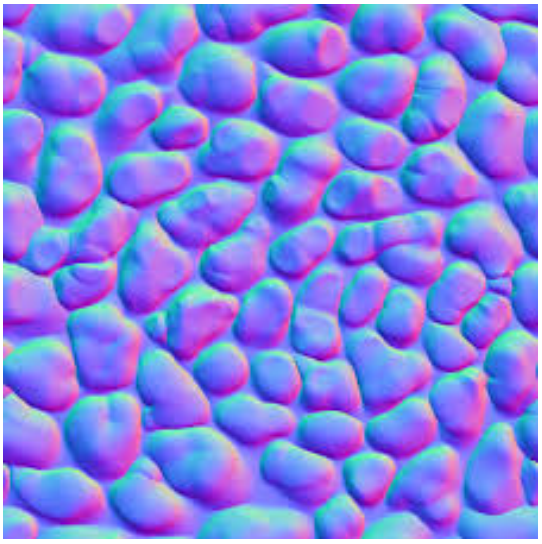

Figure 6: **Normal map**

calculate which mesh is lit and which one is in shadow.

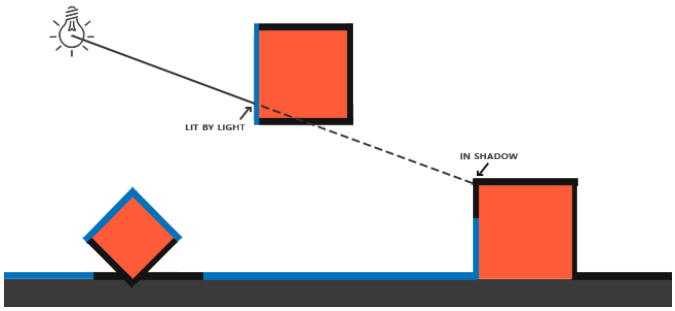

Figure 7: **Depth buffer**

In this project the shadows are created with one spot light which is casting light on meshes in front of the wall. In order to create more realistic shadows, the projection matrix that is used is with a wider field of view. The normal angle for the FoV is $\pi/4$ and for the shadows it is changed to $\pi/2$. The reason for this is because the camera has a narrower FoV than the cone of the spot light and this may result in clipping. **Figure 8** demonstrates the shadow casting in the project.

3.4 Moving objects

For the implementation of the moving objects the $\sin$ and $\cos$ functions were used. By using their main property (shown on **Figure 9**) the position of the meshes are changed in a particular range. The $\sin$ and $\cos$ functions are 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 is a cheap way of inheriting all the scale, rotation and translation of one mesh by another, all with just one multiplication. In this project the egg protectors are an example of a hierarchical transformations. In order to achieve the final effect the

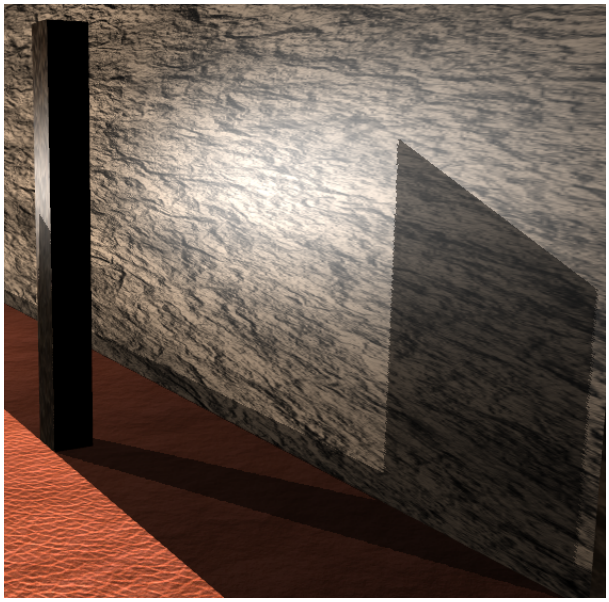

Figure 8: **Shadow**

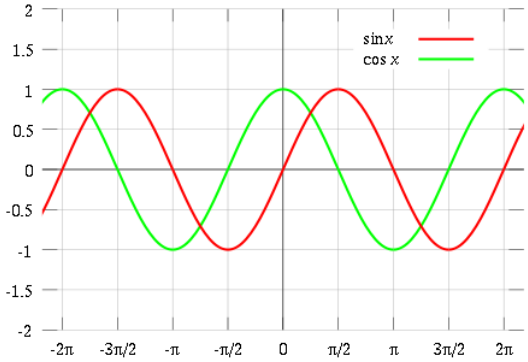

Figure 9: $\sin$ **and** $\cos$ **functions**

model matrix of the current torus is calculated from the model matrices of the bigger toruses. **Figure 10** shows the result of this graphics effect in the project.

$$[ModelMatrix] = [Translation] * [Rotation] * [Scale]$$

3.6 Skybox

The skybox is a great graphics effect that brings completeness and background to the 3D scene. The effect is achieved with a cube that has a texture applied to its inner sides and the internal parts of the cube are rendered rather than the external parts by disabling the cull face. Cube map similar to the one on **Figure 11** can be used for the inner parts of the cube.

3.7 Terrain

The terrain is a geometry object that makes the scene even more realistic. It is generated using a height map. **Figure 12** shows an example of a height map. A height map is a texture that contains height data where the pixel colour represents the y-component of the vertex position. The height map from **Figure12** is used in this project. The dark and white pixels represent the low and high parts of the scene, respectively. A crucial part of the terrain is

Figure 10: **Hierarchical Transformations**

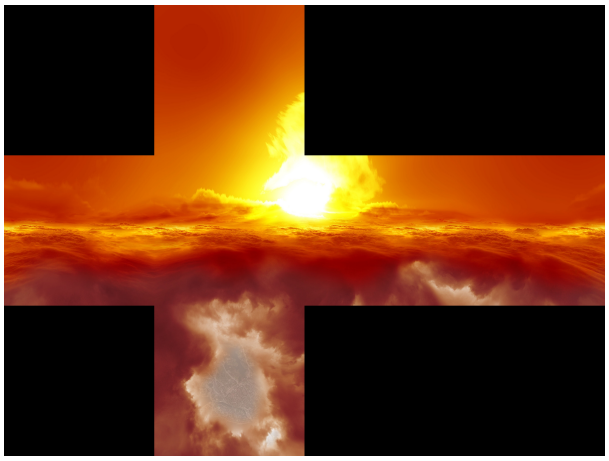

Figure 11: **Cube map used in the project**

multi-texturing and generating texture weights. These two features make the terrain more realistic and natural. The texture weights are generated based on the height of the terrain and later used if the shaders to blend between the textures. **Figure 13** shows an example of multi-texturing terrain from the project.

3.8 Water

3.8.1 Reflection
3.8.2 Refraction
3.8.3 Distortion effect
3.8.4 The Fresnel effect
3.8.5 The depth effect

3.9 Post-processing effects

Post-processing is a technique where a frame is captured in a texture, different effects are applied to the colours of the pixels and it is rendered back to the screen.

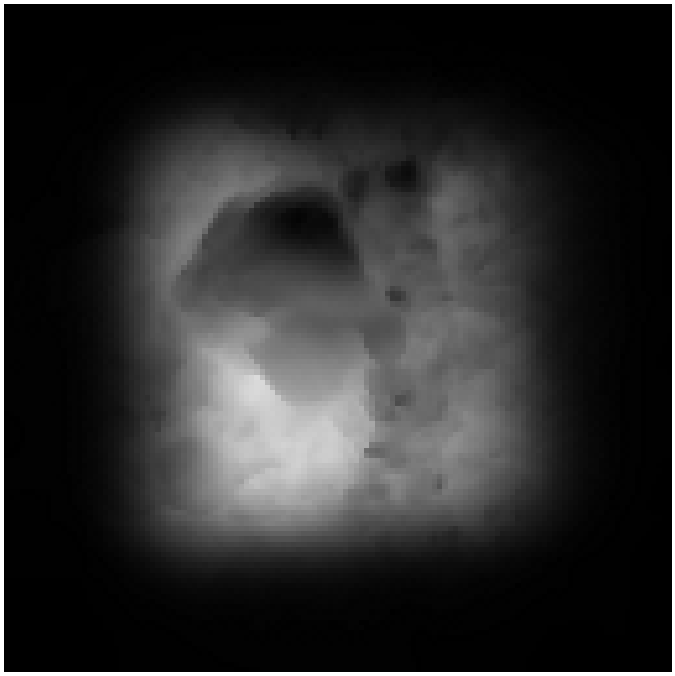

Figure 12: **Height map**

3.9.1 Mask

This is a multi-texturing technique. This effect is achieved with a back and white mask similar to the one on FIGURE. In the fragment shader both the mask and the texture from the frame buffer are sampled and there colours are multiplied together to get the final colour. In this way only the white parts of the mask are visible in the final image. FIGURE shows an example from the project. The effect of zooming in is achieved by changing the FoV angle in the projection of the camera.

3.9.2 Edge detection

Edge detection is a post-processing effect where only the edges of the geometry objects are visible. For every pixel it is determined if it is part of the edge based on the colours of the pixels around it. Suppose there are 9 pixels as shown on **Figure 14**.
After applying the edge detection filter, the intensity of pixel I_x is determined from the formula:

$$I_x = \frac{|I_1 - I_7| + |I_5 - I_3| + |I_0 - I_8| + |I_2 - I_6|}{4}$$

In order to be able to calculate pixel intensity we need to find out intensity differences between neighboring pixels and average them. Then the average value is checked against a threshold. If the intensity is greater than the threshold that mean that there is hight contrast and hence this is an edge. FIGURE shows the edge detection effect in the project.

3.9.3 Sepia

The Sepia post-processing effect is easily achievable by first applying the gray scale effect. In order to do that we need to work out the intensity of an individual pixel. This can be done with the equation: $i = f \cdot c$, where c is the vector representing the colour value and f = (0.299, 0.587, 0.184), which is the gray scale intensity

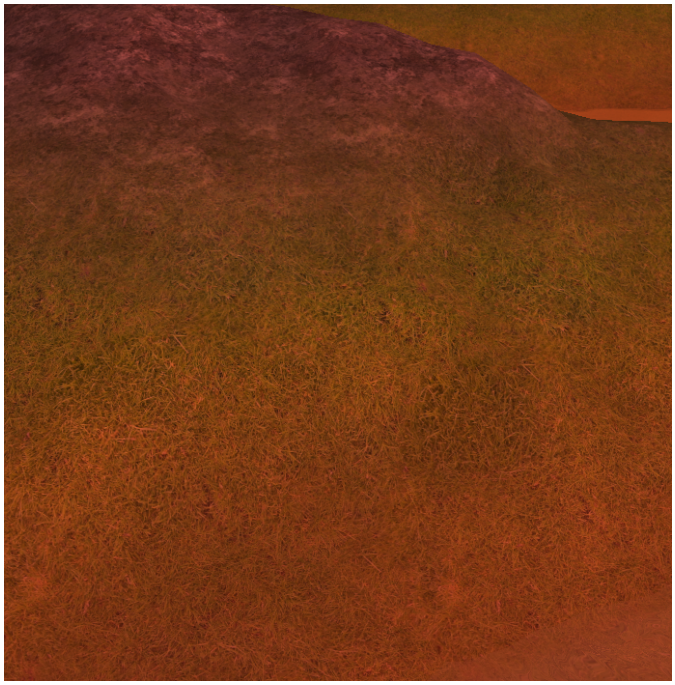

Figure 13: **Multi-texturing terrain. Sand-Grass-Rocks**

value and gives different weight to the red, green and blue colours. After this the Sepia effect is achieved by adding the colour (0.314, 0.169, -0.090) to the outgoing gray scale. FIGURE shows an example of this effect from the scene.

3.9.4 Motion blur

Motion Blur is an effect that uses two frame buffers and the textures stored in them to create the required blurring. The Motion Blur can be achieved with the standard multi-texturing approach by blending the current render with the previous one. FIGURE shows an example of Motion Blur.

3.10 Wire framing

Wire framing is a nice effect that shows the primitive geometry that structures all models and more complex geometry objects. FIGURE shows an example of this effect in the project.

4 Optimization

The scene in this project consists of quite a lot of geometry objects and additional effects. Doing optimization is crucial in this case in order to improve the performance of the GPU.

4.1 Instancing

One of the used optimization technique is Instancing. Instancing allows us to draw many geometry objects at once with a single render call, saving us all the CPU -> GPU communications each time we need to render the object. With instancing this is done only once and it avoids the performance bottleneck on the relatively

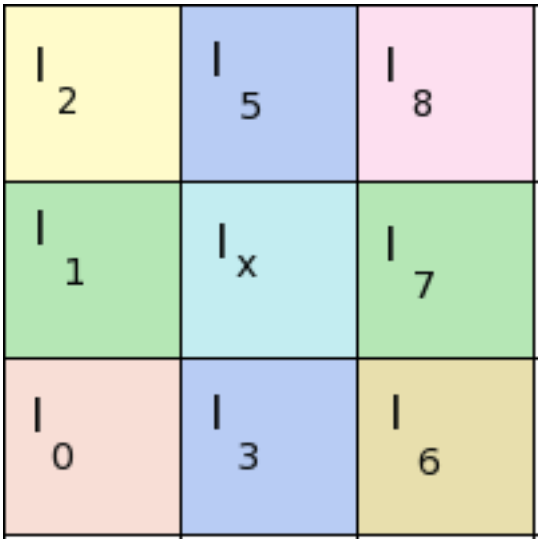

Figure 14: **Multi-texturing terrain. Sand-Grass-Rocks**

slow CPU to GPU bus. All the required data is sent to the GPU only once and then the CPU renders all the instances without having to continually communicate with the CPU. This method of drawing is really efficient for rendering a lot of models where most of these models contain the same set of vertex data but with different world transformations. For example, grass or rocks on the seabed. In this project, instancing is used to render 800 dragon eggs around the island under the water. See FIGURE for references.

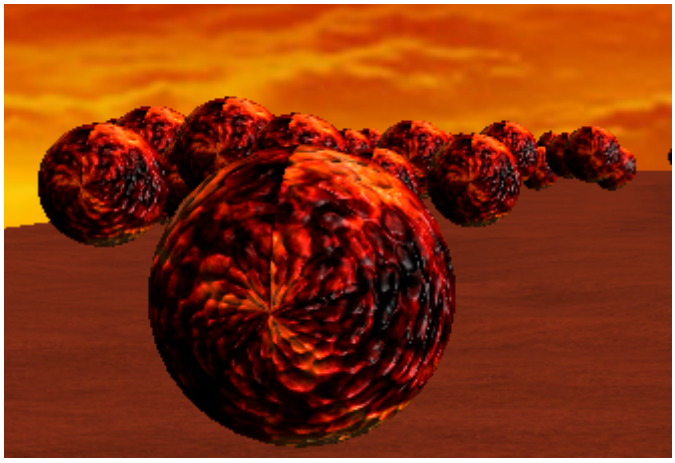

Figure 15: **800 Dragon Eggs rendered with Instancing**

5 Future Work

The initial plan for the 3D scene was to have terrain as well, however, because of the lack of time it was not possible to create it. This is something which is definitely considered for future development. In addition to this, there will be post-processing effects such as: Blurring, Motion Blur, Masking, Greyscale and fully reflective objects.

6 Conclusion

At this stage the scene was successfully finished with all the required techniques and effects. By undertaking the future work, the scene will became even more interesting and visually appealing.

References

[1] HBO, “Games of Thrones,”

[2] Edinburgh Napier University, “Previous years projects:,” http://games.soc.napier.ac.uk/graphics.html.

[3] Dr K. Chalmers, S. Serrels, “Workbook and practicals:,” https://github.com/edinburgh-napier/set08116.

[4] ThinMatrix, “Online tutorial:,” https://www.youtube.com/user/ThinMatrix.