Introduction

This report provides a brief overview for the 2nd CG assignment submitted by 19127639. The report mentions implemented features, usage and justification of the algorithms which allowed the implementation, discusses external tools used and the animations provided in the assignment.

Features Implemented

All features specified in the assignment specification were implemented in the assignment, with an issue with one of them which is talked about later in this report. These features implemented in the assignment which fall under 'modelling the environment' are:

- Including 6 composite objects.
- Including 6 different image textures for those objects/the scene.

Furthermore in regards with light, the following features were also implemented:

- Use of a single light source, a lantern.
- Ability to pick up said lantern with <F>.
- Implementation of light source attenuation in the scene.
- Ability to increase/decrease brightness radius with <K> and <L>.

In terms of 'Camera Control' features, the following was implemented in the assignment:

- Ability to rotate left/right with <A> and <D>.
- Ability to pan up and down with <Q> and <E>.
- Ability to zoom in and out with <1> and <3>.
- Ability to jump with <SPACE>.

Lastly features relating to Projection and Animation which were implemented in the assignment include:

- Ability to toggle projection with <P>.
- Ability to toggle between "Darkish" and "Bright" using <O>.
- 2 Initial complex animations that run until the program is ended. One animation was a skeleton walking up and down the scene and the other animation was multiple ghosts 'rising' from the graves and rotating.
- A complex animation that is activated via user interaction, specifically an animation where the doors open automatically when standing in front of them.

Usage and Justification of Main Algorithms

Modelling in this assignment was done using an external open-source 3D editing software, Blender, which is talked about further on in this report. Since an external tool was used to model the objects I will instead describe the process of making the models and also the main algorithm when it came to importing said models. Modelling each object was relatively easy, they were constructed using Blender, a UV Map then mapped an image/texture to each object and then the object was finally exported as an .obj file, ready to be imported into my assignment.

The main algorithm for modelling the objects in the assignment was done using the ASSIMP library and the Mesh class and Model class provided by Joey de Vries* and his online tutorial. Using those classes the models were placed in an object folder in the resources directory. Finally a reference/variable declaration was made towards the top of my assignment.cpp file, in order to store a reference to each of those models and loading the models.

The reason I used this algorithm/way of doing it was because it was the way the tutorials had done it and simply because the only other way it could of been done was me creating my own parser to parse the .obj file but that would have not be the most efficient use of my time as I could just use the ASSIMP library.

The next step was rendering the imported models. Although the methods in regards with illumination effects/surface finishes was different for every object, the generic main algorithm used in my assignment was (Modelled using PSEUDO):

```
REFERENCE TO OBJECT X/LOAD MODEL X
RENDER LOOP

    model=:mat4
    model := APPLY transformations
    Shader.setMat4<-model
    APPLY illuminations effects
    ourModel.Draw<-Shader
    RESET illumination effects

END RENDER LOOP
```

To demonstrate my understanding I will outline the algorithm specified above. A Matrix called model was initialised, transformations (for example a translation and a rotation) was applied/added to the matrix, which was then set to the shader. The model was then drawn using the shader.

This was the main algorithm used to render a generic model in my assignment, I say generic as it does vary depending on the model, the settings of the model and the number of similar models, for example for the fences I just had to declare the illumination effects for one of them and it applied to the rest whereas with the house I had to apply them and then reset them after so that it would not affect the other objects.

The reason I used this algorithm for rendering is because it was the one specified in the learnopengl tutorials but most importantly because I understood what was going on, it all made sense to me so that's why I used this particular algorithm.

Lastly, for the animation the algorithms varied depending on what I wanted to do, but the generic algorithm for all 3 of the animations in the assignment described by Pseudocode are:

GLOBAL animationVariable:= 0;

RENDER LOOP
animationVariable INCREMENT BY 1;
someTransformation<-animationVariable;
END OF RENDER LOOP

This algorithm is self explanatory. You have a global variable/a variable that is stored between renders. It gets assigned an initial value, usually 0, and then as the render loop executes each time the variable is updated and then used in as a parameter of a function or conditional statement.

The justification for using such an algorithm is simply because it allows relatively complex animations to be constructed without too much complexity in the actual code. Not only that but another justification for using such method is because I was already familiar with such a method due to the POV ray assignment earlier on in the semester.

Boxes were the only objects allowed in this assignment, as specified in the assignment specification. Because of this, only boxes were used when constructing composite objects. That being said I will go into detail for each of the more challenging objects in this assignment:

House Object - The base was created using 4 boxes, followed by the roof with two more boxes. This was relatively easy but what was difficult, and you could see from looking at the textures on the house, was trying to close the gaps created between the base shapes and the roof. I'm referring to the triangle gap on both sides of the house that is formed, I had to put multiple boxes of varying lengths to cover up all of the gaps.

Fence/Gate Object - This was also quite complex because of the number of boxes and the alignment of them. Looking back on the whole process I should of just created one

long fence and not multiple fences as it would have saved time as you can see from looking at my code I had to declare about 50 fence models.

Oil Lamp Object - This object was created using a main box in the middle and then much smaller boxes around it to make it look like an oil lamp. This was complex because I wasn't quite sure how to actually model it using just boxes.

All other objects were modelled similarly to the ones listed above but weren't as complex as these three. Purely to save space I'm not going to mention all of them but hopefully this provides some insight into the composite/simple objects I used to model my objects.

The following surface finishes/illumination effects were applied to the models:

- The global illumination settings/ones applied to all objects were specular of 1 and ambient of 0.1 and diffuse of 0.3.
- Specular illumination affect was only applied to the fences as they were the only metal/reflective type of material, the metal on the fences. All other materials had 0.1-0.2 specular, which was done to create a glossy effect/make it look better but not a reflective finish. The specular applied to the fence was around 0.7-0.8 because I wanted to simulate a reflective kind of finish since in the real world the metal would reflect the light and create a specular effect.
- I decided to let the attenuation and the other values play around with the defuse because I didn't want to hard code it as it would appear less realistic/complex. There was a default diffuse, as mentioned above, but beyond that the light attenuation formula would determine the final diffuse value.
- In terms of ambient surface finishes, the only one was the lamp as it's a light source and is suppose to have some kind of a glow/light around it, hence I added that in. I didn't add ambient to anything else because the theme was halloween and the setting was a forest, so no other lights/glowing material would be around.

Use of external Tools

As mentioned above the only external tool used was Blender and it was used for modelling the objects and UV Mapping textures to the objects. Blender is known for being a high level modelling tool and although there was limitations with it, as we could only create boxes in it, it was still better than hard coding the values into the C++ file. The reason I used Blender was because it allowed me to create complex shapes using boxes but also allowed me to create them relatively easy. On average I would say it took 30 minutes to create a model in blender where as if I had tried to hard code the values into the OpenGL C++ file, it would have been a lot less effectively and would have limited the potential of my final product.

The tool was used simply to create models and map textures to the models so that ASSIMP could import them. A description of using the tool was simply using the create tab to create a cube, using the dimensions tab to manipulate the size of the box, using the transformation tab to manipulate the position of the object and then finally using UV Mapping function to map an imported texture to the created object. My reflections on using Blender were that it was beneficial, as I mentioned above, it overall allowed me to create complex objects without the actual process to create those objects being too complex.

Animation

I briefly discussed in the main algorithm section about the animation and the generic algorithm involved. Now I will go into detail on all 3 of my animations, more specifically their implementations.

The ghost animation was simply using a variable to edit the height of the ghost and make him go through and back down the graves. This variable was incremented and applied to the translation but it had a bounds on it so it was the same length of movement each time. At the same time I was also using a value and a rotate function to rotate the ghost along the y axis, which gave the effect of him spinning in a 360 degree turn. I did this to make the animation more complex as it would have been pretty simple without the addition of the rotation.

The door animation was a bit tricky as although on the surface it would seem like the easiest one, the doors wouldn't line up and would always clash with the side of the house. This was why I was translating the doors right after they were being rotated, to make them align as close as possible to the door frame. The actual animation was, again, using a variable but this time instead of just rotating by -90/90 I increment the variable so it looked like the doors were opening slowly and hence create a better animation. The interaction was simply done by comparing the coordinates of the object with the coordinates of the camera and executing if they are equal.

The final animation was a skeleton. It was the simplest animation and I added it in towards to end just to make things more complex. Implementation was just checking x coordinate bounds and sending him in the negative x direction if they were greater than the bounds or sending him in the positive x direction if he was in the positive bounds. I then combined that with a few rotates to make it look more realistic, like he was turning.

References

Apart from using Antoni's tutorial/prac setup for my assignment space, the only reference I really used was the learnopengl. His model/mesh classes, a few methods and overall just most of my ideas for implementation were from him.

de Vries, J. (2014). *Learn OpenGL, extensive tutorial resource for learning Modern OpenGL*. [online] Learnopengl.com. Available at: https://learnopengl.com/ [Accessed 10th-29th Oct. 2018].