CITS3003 – 2016 Semester 1
Graphics and Animation

Project Report – Part 2

Caleb Fetzer – 213849876
Group # 18

The project was tested and run on Linux, in Lab 2.01
Please compile and test in the same environment.

Part 2 Overview

For the second part of the project, my partner, Elijah Fetzer (21516694), and I worked on all of the required functionality together. This part was a little more tedious, especially with Blender, with which we were totally unfamiliar. We took two attempts to get the gingerbread man working correctly, with appropriate animations. The first attempt was a large learning curve and took a while to do, and the second attempt we were able to successfully remake the gingerbread model with ease. Adding in the required code from the supplied addingAnimation.txt file was straight forward and no difficulties were faced. For part 6a, we had to use the glVertexAttribIPointer instead of glVertexAttribPointer, as we faced an issue when adding the gingerbread man model to the scene. To describe, the added object looked as if all of the vertices were being stretched from the origin out to the width of the model. This was resolved when the vertex attribute pointer function call was changed.
Another issue we faced was with our lighting from part F, of Part 1. After adding in this line,

aiMesh mesh = scene->mMeshes[0];*

our object for the ground "appeared" flipped by 90 degrees on the x-axis. To resolve the issue, we simply rotated the ground by -90 instead of the default 90 in the *init* function in *scene-start.cpp*. Overall, all of the functionality required to be implemented and additional is working.

Specific Tasks and Implementation

Part A – Texture Scale

We implemented the scene-editor texture scale menu item by modifying the call to texture2D in our fragment shader.

*fColor = color * texture2D(texture, texCoord * texScale)...*

This was done by changing the default value of 2.0 to the value, *texScale* from *scene-start.cpp*.

Part B – Modelling & Animation

This part was mostly straight forward, as all of the information could be found from the provided link and video. Blender version 2.77a was used, some trivial problems we found included that not all of the tools appeared in the same location as the instructor's in the guide/videos. We used the Linux version of Blender in order to model and animate the gingerbread man.

Part C – Exporting

Again with part B, all of the instructions were found from the project specification. At first we followed the guide provided as a txt file: *how-to-export-mesh-to-x.txt* but we noticed at step 14, that one of the export options (Export Rest Position) was left out, so we exported following the description on project specifcation page. We were able to load the monkey head model in our scene successfully, and also our gingerbread man model.

Part D – Coding to Show Animations

Following the instructions from the *addingAnimation.txt* guide, we were able to successfully add in the

required code to get the animations for the models working, with only one change. Instead of using *glVertexAttribPointer* for part 6a&b, we used glVertexAttribIPointer as we experienced issues when the model was being imported.
The code we mainly needed to consider was the expression for *POSE_TIME*, which is passed to the *calculateAnimPose(..)* function.
Since we were able to know the amount of frames of animation for the models, we hard-coded our *numFrames* variable accordingly. This way, we were able to generate the right function for POSE_TIME.

POSE_TIME = fmod(elapsedTime, numFrames);

This enabled the models to animate all of their frames.
In order to get the models moving forward, and then backtracking whilst animating, we had to consider the *sceneObj.loc* variable and the appropriate displacement function for the model. A sin wave is a periodic wave form that will go back to its original displacement after 1 full period. The value for period was calculated via:

timePeriod = sceneObj.moveDist / sceneObj.moveSpeed;

And our sin function ended up being:

*sceneObj.moveDist * sin(float(elapsedTime) / timePeriod * 0.5 * M_PI);*

This was assigned to our displacement function, which was added to the *sceneObj.loc* in the translation of the model view matrix.
Adjusting the movement speed and distance of the animated models was also successfully implemented, and the process behind that implementation was heavily based on the already implemented tools, such as *adjustSpecularShine*.

Additional Functionality and Implementation

My individual functionality for this part was creating a bouncing animation for the animated models. This was done by slightly modifying the displacement variable we created, in order to move the objects forwards and backwards.

By calling the absolute value of the sin function, it simulates the effect of a parabolic bounce.
This was then added to the displacement vector for the *y* variable, and toggled via the appropriate *toolCallback* that was coded by my partner.

*yFunc = abs(sceneObj.moveDist * sin(float(elapsedTime) / timePeriod * 0.5 * M_PI) * 0.5;*

For our group effort, we created a new model from scratch and created a different animation: crawling to standing up. The animation is 80 frames, which simulated a similar figure to the gingerbread man, crawling on all floors before standing up. The model also included additional bones to simulate better animation.

Reflection

I found the second part of the project a little more straightforward in some areas, particularly with Blender and the animation. I also found it quite enjoyable, and learned a lot. We first followed the documented guide on creating the gingerbread man model, and after that initially failed we did it the second time round but instead followed the tutorial via the videos. This was much quicker, especially with adding the rig for the gingerbread model. The tasks required were something I had never done before, and it certainly opened my eyes into the world of OpenGl and graphics animation – it made me realise just how difficult certain aspects can be, and how complicated graphics are behind the scene. As a result, I thoroughly enjoyed the tasks of part 2.