Computer Graphics Coursework 2

Build:

(tested in feng-linux)

module add qt
qmake
make
./cwk2

(all ran correctly, no external dependencies and uses fixed function pipeline, however was a bit slow in feng, but on laptop fine)

Scene Description:

My Visual Scene attempts to display an old mining cavern, featuring a dirt floor, stone walls as with stone as textures. Then objects that are built inside the room are crates and rocks through instancing plus a hierarchical mine cart that follows a track around the room. Plus various other animated objects.

This mine cart is to go around a pyramid of stacked crates that is supposed to present mining material that has been crated, with rocks strewn around the rest of the room that still needs to be mined. All this adds to the theme of the room as an old underground mine cavern.

To add to the theme of the old school mining cavern the mine cart is built as a manual push bar cart. Also known as a handcar.

Finally a door is added to this cavern and the top crate of the pile is made a bit different with some recognisable people as textures on it.

At the end of this report I have included a link to You Tube showing a video demonstration of my visual scene.

Features:

The scene is constructed around a main function called paintScene() which is called in each repaint() call via paintGL().

The first part is building up the pyramid, this is done with individual crate objects being moved around with glTranslate algorithmically and is dependent on the height of the pyramid set by the user.

The crate object itself is gotten from a .obj file via a .obj parser that I built myself as this allowed for more customisability in the end and removed any additional dependencies.

```
void MineWidget::paintScene(){
    //Draw crate Pyramid
    glPushMatrix();
    glTranslatef(0, 0, 120);
    drawCrates(wooden, 60);
    glPopMatrix();

    //Room is 1500 width, 1500 length and 400 height?
    //Draw the four walls and floor and ceiling
    glPushMatrix();
    glTranslatef(-750, 750, 0);
    this->floor(1500);
    this->wall(1500, 600);
    glPopMatrix();

    //Now the mine cart and track
    mineCart();

    //Instancing the platonic solid hexagon with normals (correct shade
    DrawRocks(wooden, 100);

    //Something rotating spinning box at top of pyramid with faces???
    glPushMatrix();
    glRotatef(idolAngle, 0, 0, 1);
    drawIdolCube(60);
    glPopMatrix();

    //Hexagonal door with mercator projection
    door();

}
```

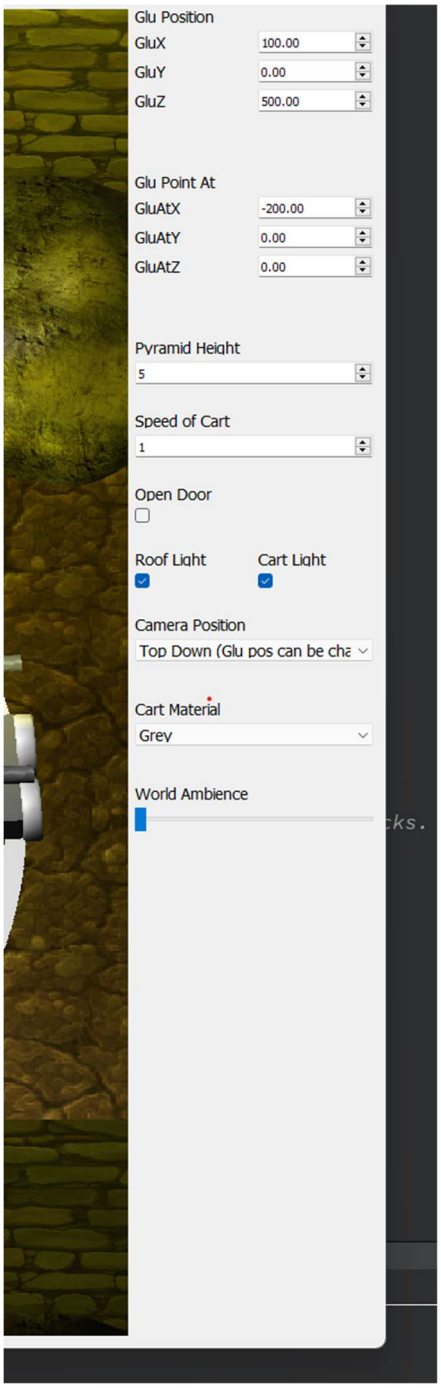
Here we can see an example of the crates being constructed together with a pyramid of height 5. This image is with a light directly above, showing the specular contribution of a more white reflection on some of the crates with the ambient and diffusive reflection showing a more wooden crate colour. The Normals are also allow for the sides to be much darker, as in this picture the light is perpendicular to the sides of the crate Normals, hence why they are darker.

Also around the room I have used a rock and duplicated it around the room to further add to the theme of the scene. As seen below we can see two of the rocks.

The rocks and crates are parsed by the application on start-up by loading the files into one QVector variable that allows for much faster access, as a face can be either a triangle or a quad the QVector is split up into either three or four vertices. This shows good results as for example, the rocks load well even with one rock containing 6000 polygons.

With various different material properties set on these objects we can see we get a more yellow hue with a slightly lighter coloured specular reflection for the rocks while the dirt floor and the crates have a more wooden diffusiveness and ambience but unlike the rocks have a lower specular reflection. The same goes for the tracks around the crates, but these tracks not only have a high specularity but a high diffusiveness and high shininess to try simulate real world train tracks.

User Interaction:



Moving onto the user interaction requirement, I have split up the application into the OpenGL widget on 90 percent of the window and the right 10 percent of the window is taken up with options for the user to take advantage of. This is done with a Qt vertical layout box and nested layouts inside this.

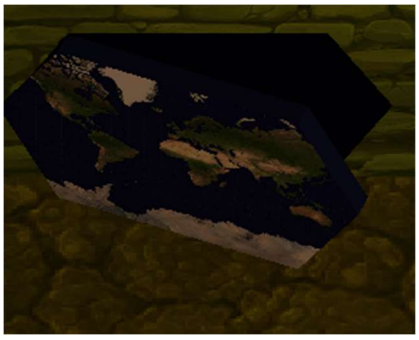
This can be seen on the right with option to change the gluLookAt() function at the top, both position and what it is looking at.

I have also included other options such as changing the height of the pyramid, (as mentioned above), changing the position of the camera to automatically follow the cart around (this changes the size of the ortho box as well to allow for a perceived follow along position from the user's point of view with objects coming into and out of view). Things like the cart material and lights can be changed too.

Using glEnable() and glDisable() for either ligth0 or light1 we can turn them on/off. The two lights are also given different properties such as their own colour and cut off. Also the world ambience with glLighModelfv() can also be changed by the user, this is there to allow the ambience setting on materials to be seen a bit clearer.

Animation:

This scene contains three animated objects, firstly we have the platonic solid cube at the top of the pyramid rotating with the two portrait textures provided, secondly we have the convex polygon object (hexagon) opening door that has the Mercator projection on the front that can be opened by the user and finally we have the cart. QTimer allows for the animations to occur much like an event loop. Door image left and Idol Cube right.



Hierarchical Object:

My scene features a mine cart following a track around the room in a circular movement. This cart is built upon various layers, these are all constructed together in the function minecart(); Here no objects are actually constructed but rather various smaller objects are all brought together whether that be the wheels or the push bar to construct a minecart(handcar), inside the function minecart()

nested matrix pushes/pops allow this to happen without the objects being explicitly sent a world position and instead using a model matrix and then being put into the world matrix. So for example for the wheel, first a push matrix operation then a translate and rotate to get to right part of the handcar then a call to the function drawWheel() to actually load the vertices, finally a popMatrix() to return to relative previous coordinate. This minecart when moving along the track has an animated push bar that goes up and down to imitate a source of the movement. This also stops when the cart is brought to a holt in the provided speed setting in the user interaction options. This whole minecart() function is wrapped by a rotate which moves it around the track.

Various material changes can also be set to this cart providing the user with a high degree of options to act upon this hierarchical model.

Finally the cart also has its own separate light that acts as headlamp that can be turned both on and off by the user, this headlamp is actually done as a directional light in OpenGL set to LIGHT1 with a more yellow hue in the ambient and diffuse setting and a lower spot cut off so it only lights ahead. The direction of the cart has also been calculated to allow for the perceived appearance of a forward facing light.

Going clockwise from top left: We see the cart set to its standard colour grey following the track. Next we have changed just the push bar colour to a green and finally we have the cart with the rooflight disabled via the provided button and the whole cart material set to shiny yellow. We can also see the specular reflection on the stone wall in this image. The images show the push par moves up and down.

Textures:

As seen above we can see how the three provided textures have been used, in both the spinning idol cube at the top of the pyramid and the opening door, but other textures that have also been used as can be seen in the stone wall, the dirt floor, the crates and the rocks. For the walls and floor we can see the specular light displays quite well with the phong reflection model, this is due to the wall and floor being broken up into much smaller tiles (quads) to allow for more vertices which takes better advantage of the model. Then by using GL_REPEAT we can assign a certain number of tiles to each texture before it repeats, so for example the wall is actually the image repeated 9 times. Also the Mercator projection door, to allow the image to be stretched onto a hexagon without actually stretching it I have provided uv coordinates that also fit into a hexagon shape instead of a box shape that would happen if it were from 0,0 to 1,1 and the image would appear more stretched.

See You Tube Video Attached for Demonstration of Application:

https://youtu.be/yWS8320qrXU

Acknowledgements

Texture and obj file (containing normalized vertices positions) for the crate, rock and wheel were gotten with free licences from www.turbosquid.com.