David Ogle *AC41001 Graphics*

Assignment 1, Moving Parts

For this project I modeled a tank focusing especially on the caterpillar tracks, their movement and lighting.
The main class for the project is the ‘run’ class which is the entry point for the program and makes use of the wrapper_glfw class to initialize glfw and build shaders. The project has four shape classes; cuboid, cylinder, sphere and tank_track.
Cuboid is a modification of the example cube class and has a functions to define vertices, set normals and add a triangle to the vertices array. The cube class was originally used to create a single ‘track’ in a set of caterpillar tracks. The sphere class again is a modification on the example sphere object in the examples and is used only to create the ‘lamp’ objects for when the light sources are within the scene. The cylinder shape was the hardest to achieve and creates a cylinder object using triangle fans, triangle strips and an indexing of vertices.
The class first defines the vertices for the top circle of the cylinder, then the bottom, it then redefines the vertices minus the central points for use with the triangle strip around the side that will have different normals. The code sets normals and colours for each vertice as it is defined. Lastly the class defines an elementBuffer with the relevant index numbers for defining the two triangle fans and the triangle strip. The cylinder class is used as a base for the wheels for the tank and also for the gun parts.
The last shape ‘tank_track’ is an elaboration upon the cuboid class. The original design was to add an extra five sides to the cube shape to create an 3D ‘T’ shape to create a more realistic ‘track’ object. The protruding cuboid was originally meant to be centered at the top of the object however due to a miscalculation the cube became offset to one side. The resulting effect of drawing multiples of this object was that the tracks look more realistically joined and so I opted not to redefine the vertice coordinates.
The tank_track object was used to replace the cuboid object when drawing the tracks.
Each object has a lighting class which is a entity class holding values for lighting including if the object has an emissive lighting value.
The track object is used to draw a single caterpillar track. It creates one tank_track object and one cylinder. When first created the track class sets the starting positions for all objects. It loops through all 36 tracks, splitting the tracks into 4 sections; the anti-clockwise curve, the left to right straight, the clockwise curve and the right to left straight. For each of these tracks a ‘transformation’ object is stored that holds the values by which the original ‘tank_track’ must be transformed to draw the track in the correct position. In the draw loop, for each track drawn the code requests the appropriate transformation-matrix from the transform object, then draws the base ‘tank_track’ object transformed by the appropriate matrix.
To move the track the function ‘move’ checks each current transformation and increments its value by a fixed small amount. This was quite challenging as it involves tracking when a transformation switches between two of the four above-mentioned zones. Also for the curve sections of the track it was important to adjust the transformation by the correct value of theta and also to rotate the tracks in a consistent manner so they always face the center of the caterpillar tracks. To increase, decrease or reverse the speed of the tracks the code ‘speed’ variable represents how many times the transformation by the fixed amount will be

performed before the object is drawn, thus for a faster speed the tracks are drawn fewer times to complete a rotation.
Once the tracks had been implemented, the body class was created. This draws a more complex object using triangles to create a rudimentary tank body. This was particularly challenging in getting the vertices and normals correct and involved a lot of sketching out diagrams on scrap paper. The body class also has a function that stores transformations for the cylinder object used to create the gun. The gun is split into three cylinder transformations; the turret, the barrel and the muzzel.
When drawn the complete scene creates a transformation matrix for global transformations (transformations applicable to all objects), then that value is timed by the tank body transformation matrix. The tank body is then drawn and the turret spins are then retrieved as a matrix and timed with the general turret transformations. Lastly this is timed with the tank body transformations and then drawn to render the gun parts. The two tracks with four wheels are then drawn in much the same way using three ‘for loops’.
The lighting for the scene uses Phong shading as I felt by lighting each pixel rather than each vertex the scene has a more realistic lighting and the performance hit seemed to be negligible on such a simple scene. The efficiency is slightly increased by the use of a Blinn-Phong lighting model which substitutes the value ‘normalize(L + V)’ rather than the more costly ‘reflect(-L, N)’. The program makes use of positional and non-positional lighting. The default lighting mode is for the light to be infinitely far away and not make use the attenuation variable in the lighting calculation. This however can be changed by the ‘M’ key to set the light sources to two ‘lamp’ objects in the scene allowing a user to control the position of the light sources and including attenuation in the lighting calculations.

For this assignment I had originally planned to create a scene using cogs however I had trouble creating the cylinder object I had planned to use as a basis for my cog class. In the end I decided to create a scene using the cube class I already had and elaborate upon that. I am very pleased with the result and am especially pleased with the cylinder class use in the gun parts and the caterpillar tracks movement which was particularly hard to get moving. I had some problems in rendering the moving turret as it was rotating in strange ways until I remembered I had to define my transformations in reverse order. I also had time to add a second light source to my scene which is written so that it can be expanded to any number of light sources. This was a challenge in finding a good way to pass an array of vec3 objects to the shaders. With extra time I would have liked to get the textures working properly on the tank, however properly defining the points for the texture coords is too time consuming for this assignment.