Group Name:

Group F

Group Members Name & Student ID:

Molenaar, Gerrit	001238532
Morgan-Owiriwa, Orisakite	001235457
Okeke, Chidalu	001229468

Group Member Contribution:

Molenaar, Gerrit – I combined both codes and gave the dragon the ability to move, move the camera, use orthographic projection, perspective projection and scaling.

Morgan-Owiriwa – I created the dragon object

Okeke Chidalu – I created the table and the room for the project

Working procedure:

Because debugging opengl is hard, we have glCheckError taken from https://learnopengl.com/In-Practice/Debugging.

RenderObject is the most important class, it contains all the data needed to render an object. There is two ways to use the class, either you can use inheritance and override define method or provide a .obj file when generating the object.

With inheritance you would define all the vertices, colours, normals, and faces used to draw the object. _vertices are a list of x, y, z positions. Colours are a list of rgb values. Normals are not being used at the moment. Faces define a square, index colour, and index normal. If index colour and index normal are not defined the will default to 0. If you need a triangle face the last to vertices indices can be the same, though this does result in two overlapping triangles to be drawn. This is an area of improvement.

If instead providing a .obj file, it would first read and load all the data contained within that file. The only thing the .obj file does not provide is colours, as for now this is defined in the load method. For loading the file, we have load, parse_face, parse_normal, parse_vertex, get_vec3, get_face, insert_face methods.

Regardless of method the generate method will call triangulate(), this method will take the vertices, colour, normal, and face list and turn it into two new list that opengl excepts. One of the lists is the _vertices_internal which stores the vertex data. The list is organized as:

Position1	Colour1	Normal1	position2	colour2	normal2	...

The _indices_internal defines the triangles with indices pointing to _vertices_internal.

we create vertex buffer which moves the _vertices_internal data to the gpu. We use vertexArrayObject to tell opengl how the vertex buffer is setup. ElementBuffer moves the _indices_internal to the gpu.

MemoryController class is a collection of renderObject, this is to make it easier to have many objects.

Shader class loads in the shaders compiles them and binds them.

State struct is just collecting all the global variables.

So_called_physics function keeps the dragon on the table.

Display, reshape, keyboard, specialKeys, mouse, mousedrag are all callback function for GULT

Init function creates the shader, and renderobject.

main is main.