CS7GV5-A-SEM202-201920 REAL-TIME ANIMATION
Assignment 1 - Plane rotations

- **Simple representation of a plane with pitch, roll, and yaw rotations, using Euler Angles (~40%) (Observe gimbal-lock occurring when 2 axes are aligned):**

```
glm::mat4 local;
local = glm::eulerAngleYXZ(rotate_y_angle, rotate_x_angle, rotate_z_angle);
glUniformMatrix4fv(model_location, 1, GL_FALSE, glm::value_ptr(local));
```

Used glm library's eulerAngleYXZ function to implement Euler angles. With every keypress the angle was increasing by 0.05 radians, which resulted in Gimbal-lock when two axes aligned together. Keys were used to increase and decrease angles over axes.

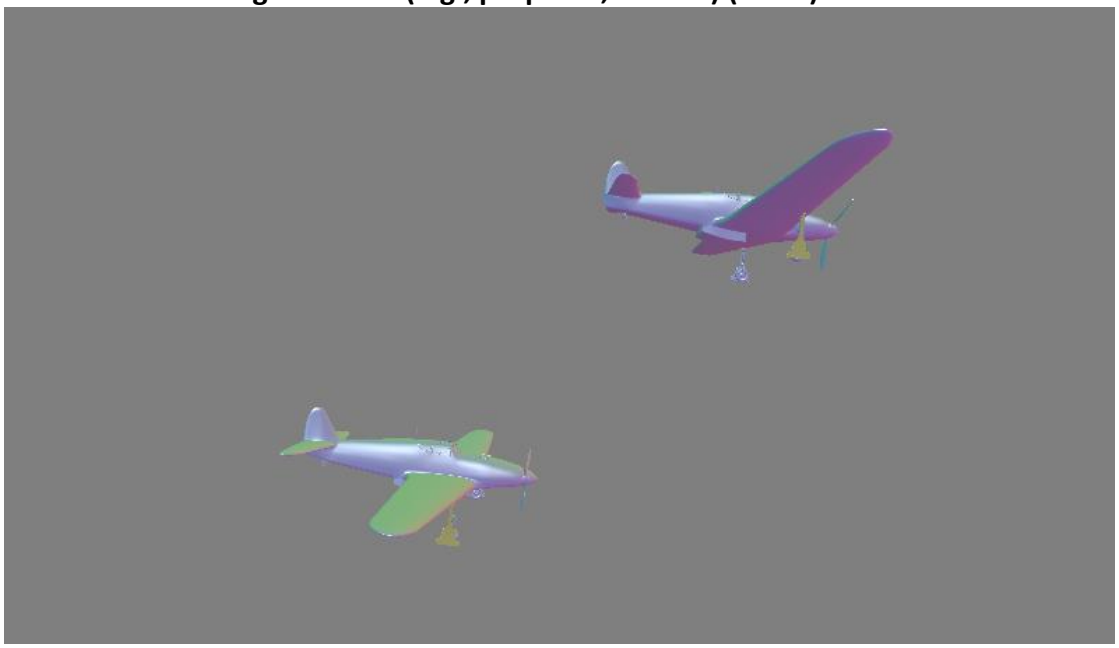
- **Extra Features (~60%):**

- **Overcome gimbal-lock using quaternions to represent the rotations (~20%)**

```
//quaternions
versor local_quat = versor();
local_quat = quat_from_axis_deg(qy, axis.x, axis.y, axis.z);
mat4 quat = quat_to_mat4(local_quat);
glUniformMatrix4fv(model_location, 1, GL_FALSE, quat.m);
```

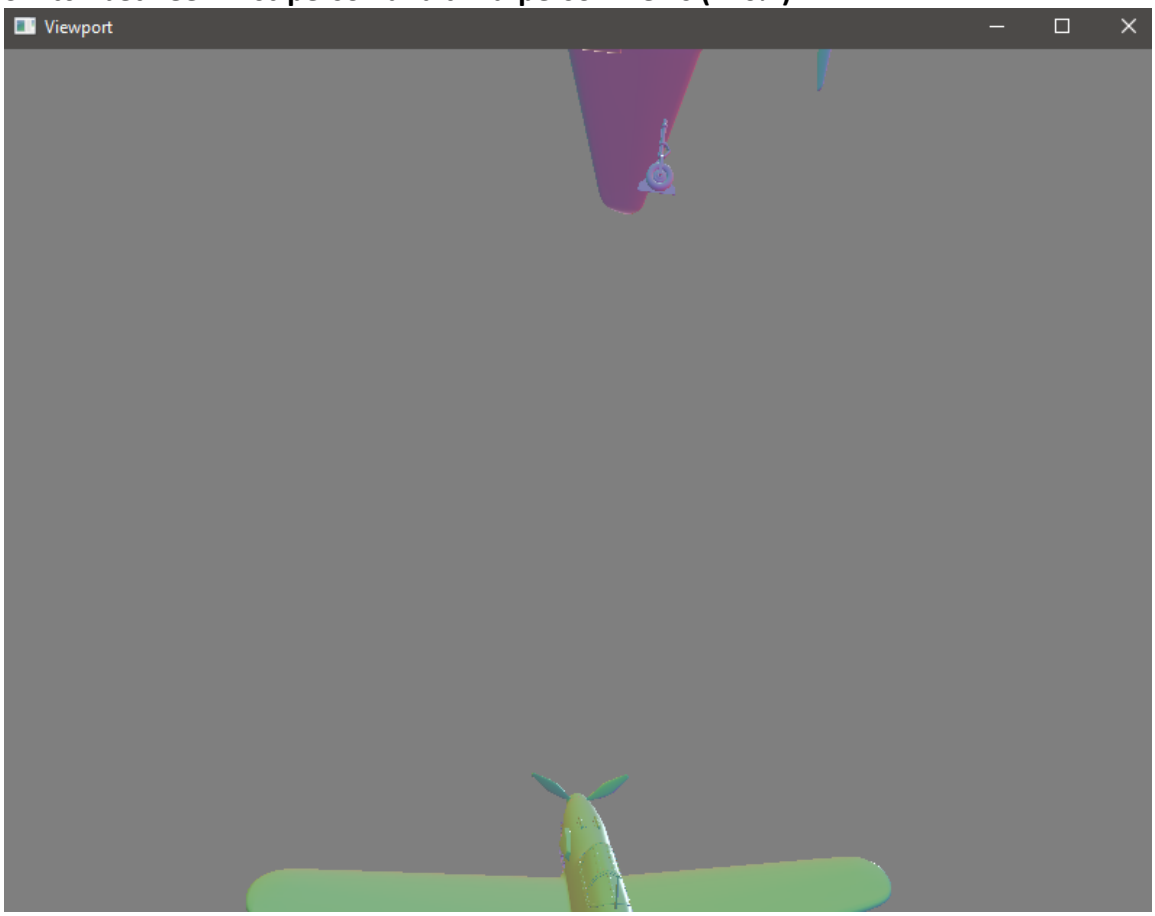
This function was defined in Anton's maths class. The first creates a quaternion using degree and axis given and the second one converts the quaternion to a 4x4 matrix which can be used with the model matrix.

- **Hierarchical moving elements (e.g., propeller, wheels) (~20%)**



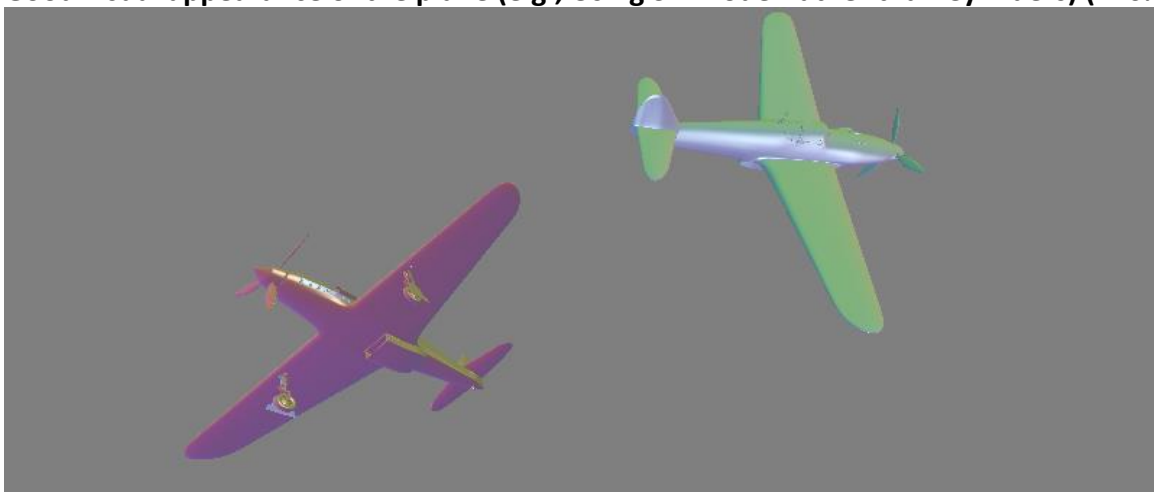
I used two aeroplanes as objects to show hierarchy between them. The idea is to move the second plane when the first one moves.

- **Switch between first-person and third-person views (~20%)**



Used mouse movement for changing the camera direction and up/down/right/left keys for the translation of objects over the scene.

- **Good visual appearance of the plane (e.g., Using 3D model rather than Cylinders) (~20%)**



- Other:

```
//fragment shader phong
#version 330

in vec3 fragPos;
in vec3 nEye;
in vec3 vsNormals;
in vec3 vsLightPos;
in vec3 vsPosition;

out vec4 fragColour;

uniform vec3 lightPos;
uniform vec3 viewPos;

uniform float ambientStr;
uniform float specularStr;

const vec3 ambientColor = vec3(0.2, 0.2, 0.2);
const vec3 diffuseColor = vec3(0.5, 0.5, 0.5);
const vec3 specColor = vec3(1.0, 1.0, 1.0);

void main(){
    // Global Lighting Variables
    vec3 lightDir = normalize(lightPos - fragPos);
    vec3 norm = normalize(vsNormals);
    // Ambient Lighting
    vec3 ambient = ambientStr * ambientColor;
    // Diffuse Lighting
    float diff = max(dot(norm, lightDir), 1.0);
    vec3 diffuse = diff * diffuseColor;
    // Specular Lighting
    vec3 reflectDir = reflect(lightDir, norm);
    vec3 viewDir = normalize(viewPos - fragPos);
    float spec = pow(max(dot(viewDir, reflectDir), 0.0), 8.0);
    vec3 specular = spec * specularStr * specColor;
    fragColour = vec4(nEye*ambient + diffuse + specular, 0.5);
}
```

Used Phong Shading model for lighting and giving the airplane different colours on different angles.

➢ **References:**

- https://glm.g-truc.net/0.9.3/api/a00164.html
- Anton's maths library.
- https://learnopengl.com/Getting-started/Camera
- https://learnopengl.com/Lighting/Basic-Lighting