Student Name: Hanul Rheem
Student ID: 20109218

COMP612 Computer Graphics Programming

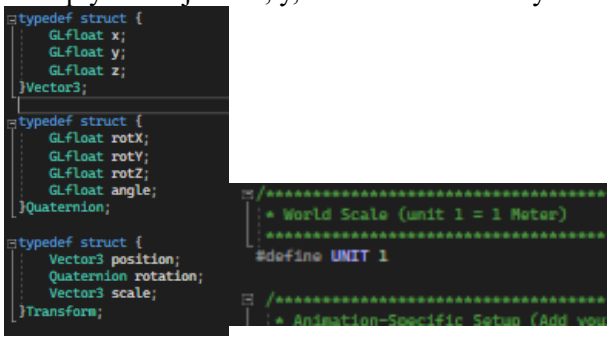
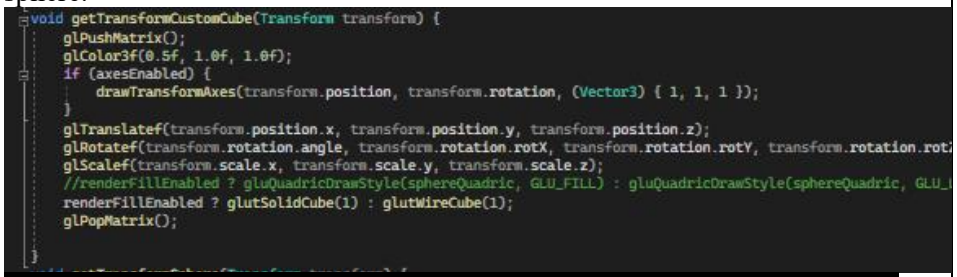
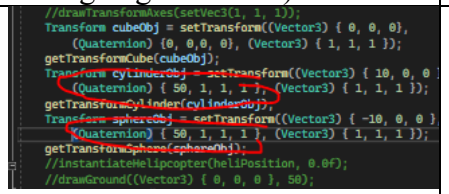
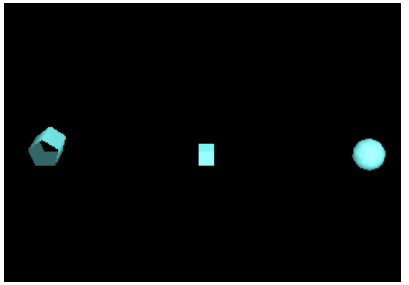
Semester 1, 2023

Assignment 2

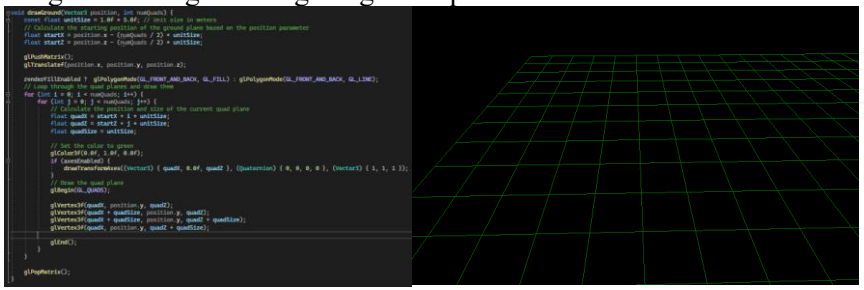
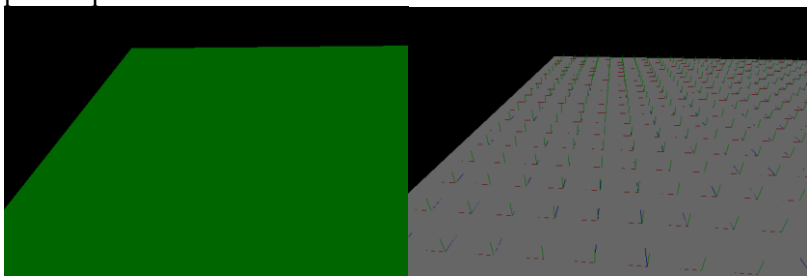
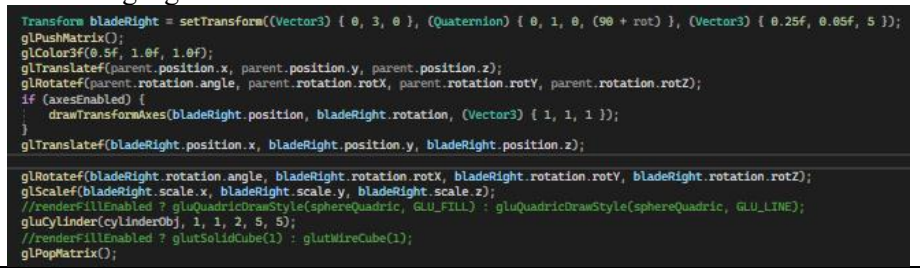
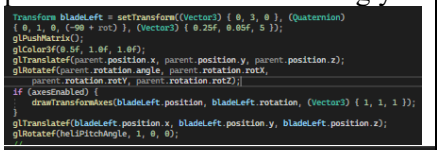
Project Logbook

Student Name: Hanul Rheem
Student ID: 20109218
Student Email: dgw7626@autuni.ac.nz

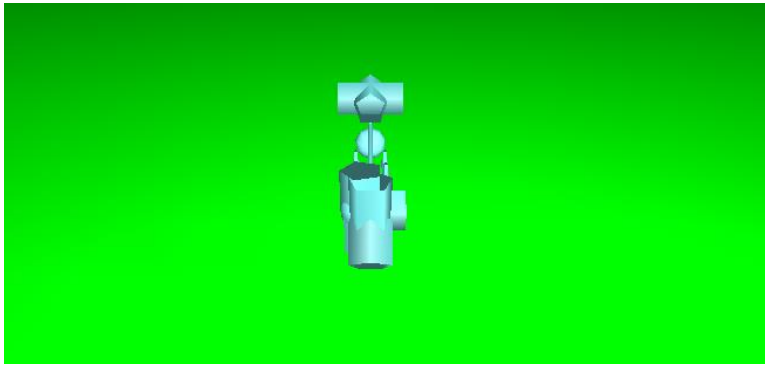
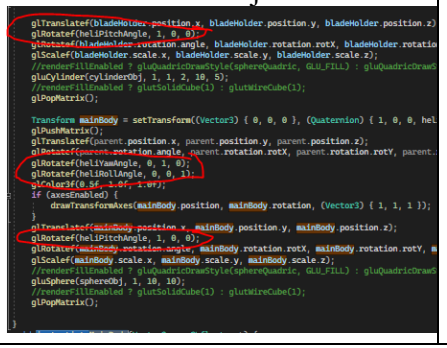
Student Name: Hanul Rheem
Student ID: 20109218

Date	To do	Work Done	Detail/ Description	Bugs Encountered (the bug will be listed if end up having bugs or issues).	Hours
2/5/2023	-Basic Scene Settings -Measuring world scales -Making custom default prefab assets.	-World scale measurement. Custom default prefab assets (cylinder, cube, sphere objects).	In my project, world-scale measurement is represented as Unit 1, with 1 meter being the default value. To incorporate the scale of an object, it is necessary to multiply the object's x, y, and z dimensions by the instance of the struct code.  In the scene, any object instantiated will have the scale values applied to "glScalef" similar to the example provided. I have created additional struct instances for Transform, allowing them to store the current position, scale, and rotation values. The default assets for this project include a cube, cylinder, and sphere.  	 It appears that the rotation along the x, y, and z axes is not being applied correctly. There might be an error in the usage of "glRotatef," or possibly an issue with the code structure itself.  The objects remain stationary, indicating that each rotation axis is unique and cannot be applied simultaneously.	6hours

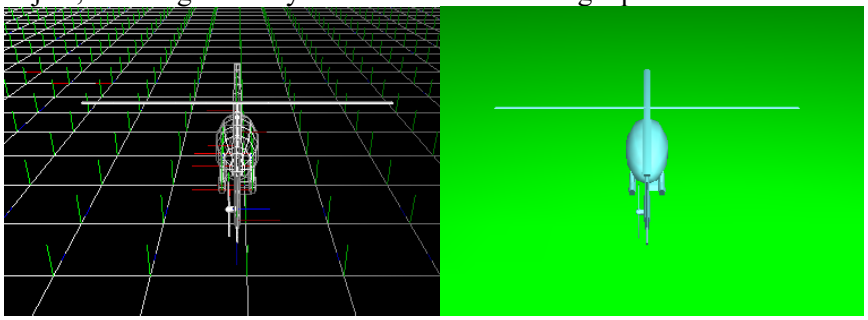
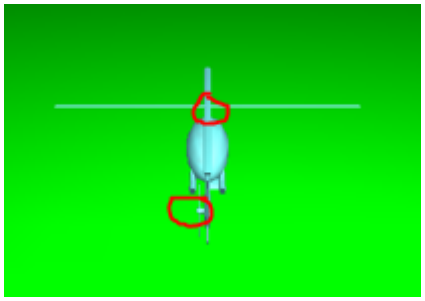
Student Name: Hanul Rheem
Student ID: 20109218

3/5/2023	-Implement ground/terrain -Making helicopter model based on default prefab assets.	-Ground plane implementation. -Initial helicopter model.	This is my initial implementation of the ground plane, which is generated as a quad plane. I attempted to use "glTriangle" or "glQuadStrip" methods, but they didn't work well with this particular approach. Implementing it wasn't too challenging since I applied the same logic used in the previous snow assignment for generating the ground plane.  Each ground plane includes the vector3 axes, which will be displayed when the "P" key is pressed. This allows me to view the vector3 coordinates of the plane's position.  I utilized the default asset method to create the helicopter model, which consists of multiple object instances. However, the helicopter model appears distorted because the initial scale was set to 1,1,1. It is necessary to adjust the scale using a glfloat value to correct this issue.  Transform bladeRight = setTransform((Vector3) { 0, 3, 0 }, (Quaternion) { 0, 1, 0, (90 + rot) }, (Vector3) { 0.25f, 0.05f, 5 }); glPushMatrix(); glColor3f(0.5f, 1.0f, 1.0f); glTranslatef(parent.position.x, parent.position.y, parent.position.z); glRotatef(parent.rotation.angle, parent.rotation.rotX, parent.rotation.rotY, parent.rotation.rotZ); if (axesEnabled) { drawTransformAxes(bladeRight.position, bladeRight.rotation, (Vector3) { 1, 1, 1 }); } glTranslatef(bladeRight.position.x, bladeRight.position.y, bladeRight.position.z); glRotatef(bladeRight.rotation.angle, bladeRight.rotation.rotX, bladeRight.rotation.rotY, bladeRight.rotation.rotZ); glScalef(bladeRight.scale.x, bladeRight.scale.y, bladeRight.scale.z); //renderFillEnabled ? gluQuadricDrawStyle(sphereQuadric, GLU_FILL) : gluQuadricDrawStyle(sphereQuadric, GLU_LINE); gluCylinder(cylinderObj, 1, 1, 2, 5, 5); //renderFillEnabled ? glutSolidCube(1) : glutWireCube(1); glPopMatrix();	 The object is not following the camera position because the child object lacks any parent data regarding position or rotation. To address this, I simply applied "glTranslatef" and "glRotateF" functions to adjust the object's position and rotation accordingly.  Transform bladeLeft = setTransform((Vector3) { 0, 3, 0 }, (Quaternion) { 0, 1, 0, (-90 + rot) }, (Vector3) { 0.25f, 0.05f, 5 }); glPushMatrix(); glColor3f(0.5f, 1.0f, 1.0f); glTranslatef(parent.position.x, parent.position.y, parent.position.z); glRotatef(parent.rotation.angle, parent.rotation.rotX, parent.rotation.rotY, parent.rotation.rotZ); if (axesEnabled) { drawTransformAxes(bladeLeft.position, bladeLeft.rotation, (Vector3) { 1, 1, 1 }); } glTranslatef(bladeLeft.position.x, bladeLeft.position.y, bladeLeft.position.z); glRotatef(heliPitchAngle, 1, 0, 0);  I encountered a type error when attempting to insert a double value into a GLfloat variable. It seems there is a mismatch between the data types.	4hours

Student Name: Hanul Rheem
Student ID: 20109218

			`` void instantiateHelipcopter(Vector3 pos) { if (sphereObj != NULL && cylinderObj != NULL) { renderFillEnabled ? gluQuadricDrawStyle(sphereObj, GLU_FILL) : gluQuadricDrawStyle(sphereObj, GLU_LINE); renderFillEnabled ? gluQuadricDrawStyle(cylinderObj, GLU_FILL) : gluQuadricDrawStyle(cylinderObj, GLU_LINE); //Transform cubeObj3 = setTransform((Vector3) { 0, 0, 8 }, (Quaternion) { 0, 1, 0, 180 }, (Vector3) { 1, 1, 1 }); //getTransformSphere(cubeObj3); // //Transform cubeObj4 = setTransform((Vector3) { 5, 0, 10 }, (Quaternion) { 0, 1, 0, -45 }, (Vector3) { 2, 2, 2 }); //getTransformSphere(cubeObj4); glPushMatrix(); instantiateMainBody(pos, 0); glPopMatrix(); } } ``` The object failed to attach to the parent object due to the absence of parent position and rotation values in the original method for the freeglut-based default assets. Consequently, there was no way to store and transfer these values to the transform child object. However, in the provided code, the parent's transform position and rotation values are presented, enabling the application of the parent's position prior to applying the child's position and rotations. 		
9/5/2023	-Animating prefab assets (Rotor animations) -Functional controls/motion controls. -Tracking camera, camera look at object implementation.	-Finalizing the helicopter model -making initial animation with the pitch, yaw, heave angle axes. -camera look at object implementation.	This is my final model for the helicopter. You can observe the model below. The helicopter blades are appropriately scaled and positioned relative to the helicopter object. The axes are correctly assigned to each object. Furthermore, the transform child now contains the position and rotation values of the parent	The same issue I had from the 2/5/2023, the rotation didn’t apply to the instantiated object. 	8hours

Student Name: Hanul Rheem
Student ID: 20109218

			object, ensuring that they are all attached to a single parent.  To enable motion control, it is necessary to calculate the yaw, pitch, and heave angles. These angles will determine how the helicopter moves, including tilting, turning, and other motions. In my local variables, the pitch angle, yaw angle, and roll angle are represented as x, y, and z rotations respectively. Calculating these angles proved to be a challenging task, particularly in determining the appropriate axis for keyboard motion. Motion can be represented in two ways: sway and surge. The angles must consider the motion value to determine whether it corresponds to swaying left or right, or moving forward or backward. `Vector3f camera = { 0,1,10 };` `Vector3f heliPosition = { 0,1,-380 };` `GLdouble heliYawAngleY = 0.0f;` `GLdouble heliPitchAngleX = 0.0f;` `GLdouble heliRollAngleZ = 0.0f;` `GLfloat limitedTurningPoint = 15.0f;` `//Helicopter rotor rotation` `GLfloat rotorRotation = 0.0f;` `GLfloat startSpeed = 8.8f;` `GLfloat speed = 0.0f;` `//light box rotation` `GLfloat lightSourceRotation = 0.0f;` `GLfloat lightStartSpeed = 0.5f;` `GLfloat lightSpeed = 0.0f;` `//const values for helicopter predefined angles.` `const GLdouble DEG_TO_RAD = PI / 180.0f;` `const GLdouble MAX_PITCH_ANGLE = 15.0f;` `const GLdouble HELI_SPEED = 28.0f;` `const GLdouble RPM = 90.0f;` `const GLdouble MAX_ANGLE = 360.0f;` When the helicopter is turning left or right, the speed is determined by adding the RPM (revolutions per minute) value. If the rotation angle reaches 360	I had to manually assign the x, y, and z rotation values individually, but the propeller still does not rotate. Refactoring is necessary as the code lacks consistency and readability. Despite these efforts, the propeller rotation issue remains unresolved.  still stays idle.	

Student Name: Hanul Rheem
Student ID: 20109218

degrees, the y angle will be decreased.

```
if (keyboardMotion.Yaw != MOTION_NONE) {
    //heli y position will be added if the yaw angle is enabled.
    heliYawAngleY += RPM * FRAME_TIME_SEC * keyboardMotion.Yaw;
    //if heli angle is less than max angle.
    if (heliYawAngleY > MAX_ANGLE)
        //it will substract the max value.
        heliYawAngleY -= MAX_ANGLE;
        //other wise it will added
    else if (heliYawAngleY < 0.0f)
        heliYawAngleY += MAX_ANGLE;
}
```

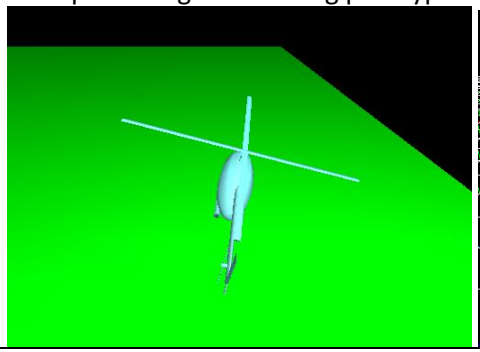
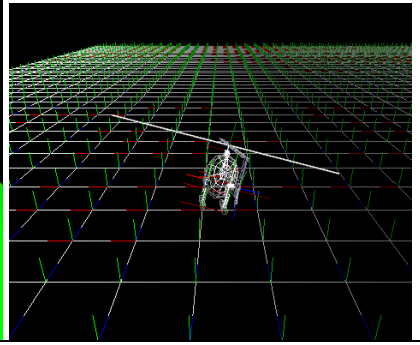
When the helicopter is surging forward, the x and z values will be incremented to simulate the forward movement. If the pitch angle reaches its maximum value, it will be maintained to create the appearance of surging forward or backward.

```
if (keyboardMotion.Surge != MOTION_NONE) {
    /* TEMPLATE: Move your object backward if .Surge < 0, or forward if .Surge > 0 */
    //heli position x will be added with the yaw angle value if the helicopter surges.
    heliPosition.x += (GLfloat)(sinf((GLfloat)heliYawAngleY * (GLfloat)DEG_TO_RAD) * keyboardMotion.Surge * FRAME_TIME_SEC * HELI_SPEED);
    //heli position z will be added with yaw angle value if the helicopter surges.
    heliPosition.z += (GLfloat)(cosf((GLfloat)heliYawAngleY * (GLfloat)DEG_TO_RAD) * keyboardMotion.Surge * FRAME_TIME_SEC * HELI_SPEED);
    //heli position x will be added when the helicopeter surges.
    heliPitchAngleX += 1.0f * keyboardMotion.Surge;
    // if max ptictch reached it it will limit the angle positions
    if (heliPitchAngleX > MAX_PITCH_ANGLE) { heliPitchAngleX = MAX_PITCH_ANGLE; }
    if (heliPitchAngleX < -MAX_PITCH_ANGLE) { heliPitchAngleX = -MAX_PITCH_ANGLE; }
}
else {
    //other wise it will add the pitch x value or subtract the pitch x angle.
    if (heliPitchAngleX < 0 && heliPitchAngleX >= -limitedTurningPoint) {
        heliPitchAngleX += 0.5f;
    }

    if (heliPitchAngleX > 0 && heliPitchAngleX <= limitedTurningPoint) {
        heliPitchAngleX -= 0.5f;
    }
}
```

When the helicopter is swaying, the position of the helicopter will be adjusted by adding values to both the x and y coordinates. This is similar to the surge conditions. To create a swaying motion that moves the helicopter left and up, or right and down, the midpoint for swaying needs to be determined. If the helicopter reaches a limited turning point, it will maintain that position while continuing in the same direction it is heading.

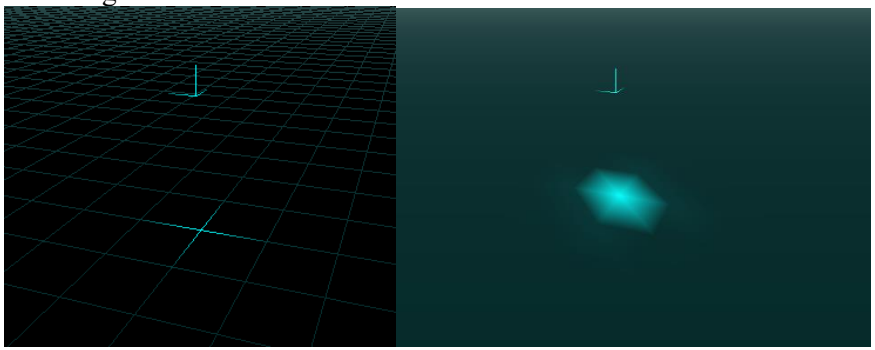
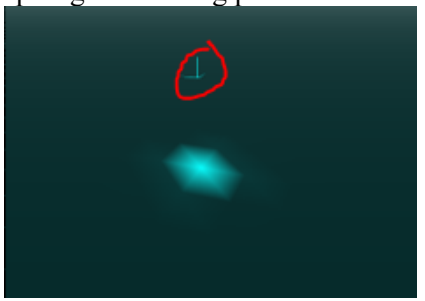
Student Name: Hanul Rheem
Student ID: 20109218

			`` if (keyboardMotion.Sway != MOTION_NONE) { /* TEMPLATE: Move (strafe) your object left if .Sway < 0, or right if .Sway > 0 */ /* TEMPLATE: Move (strafe) your object left if .Sway < 0, or right if .Sway > 0 */ // need to flip inverse somewhere in here //check and add x position if the heliyawangle y is sway. heliPosition.x += (GLfloat)(- cos(heliYawAngleY * (PI / 180.0f)) * keyboardMotion.Sway * FRAME_TIME_SEC * 32.0f); //check and add z position if the heliyawangle y is sway heliPosition.z += (GLfloat)(sin(heliYawAngleY * (PI / 180.0f)) * keyboardMotion.Sway * FRAME_TIME_SEC * 32.0f); //add helirollangle when the sway is enabled. heliRollAngleZ += 1.0f * keyboardMotion.Sway; //if the angle is limited set to the the limited turning point. will stay in that particular angle. if (heliRollAngleZ > limitedTurningPoint) { heliRollAngleZ = limitedTurningPoint; } if (heliRollAngleZ < -limitedTurningPoint) { heliRollAngleZ = -limitedTurningPoint; } } else { //if the angle is not limited start adding angle Z axis if (heliRollAngleZ < 0 && heliRollAngleZ >= -limitedTurningPoint) { heliRollAngleZ += 0.5f; } //if the angle is not limited start subtracting angle Z axis. if (heliRollAngleZ > 0 && heliRollAngleZ <= limitedTurningPoint) { heliRollAngleZ -= 0.5f; } } if (keyboardMotion.Heave != MOTION_NONE) { /* TEMPLATE: Move your object down if .Heave < 0, or up if .Heave > 0 */ //add y positon if the heave is enabled. heliPosition.y += speed * FRAME_TIME_SEC * keyboardMotion.Heave; //set the ground detection to be -0.5f it will not go below that value. if (heliPosition.y - 0.5f < 0 && heliPosition.y != 0) { //if the position is 0.5f it will set speed to 0. heliPosition.y = 0.5f; if (speed >= 0) { speed = 0.0f; } } } ``` Example footage of working protoype.  		
15/5/2023	-fixing helicopter motion control	-the helicopter’s propellers rotation.	Now the helicopter has functioning propeller rotations, and the same applies to the tail rotor. The issue I encountered was that the "glRotatef()" function cannot receive all values at once; they must be inserted individually, with different angle axis values. Here is the code I implemented, which includes a method that takes the parameter of real-time rotation on the y-axis.	There are no errors remaining for the rotating object; the rotation issue has finally been resolved.	6hours

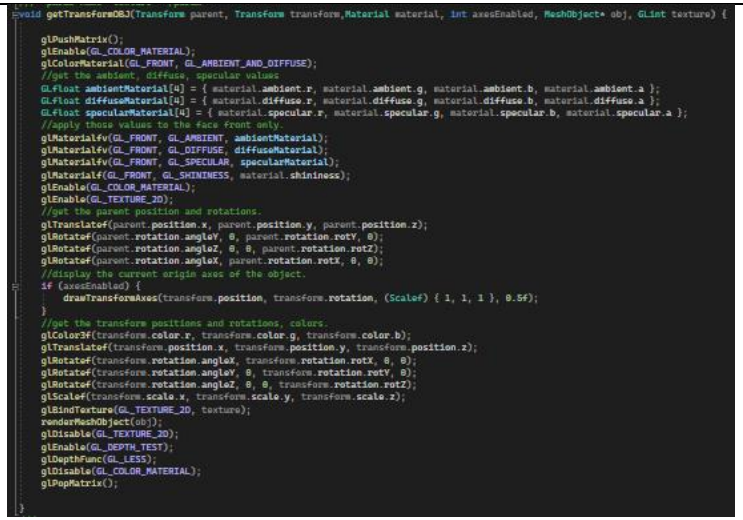
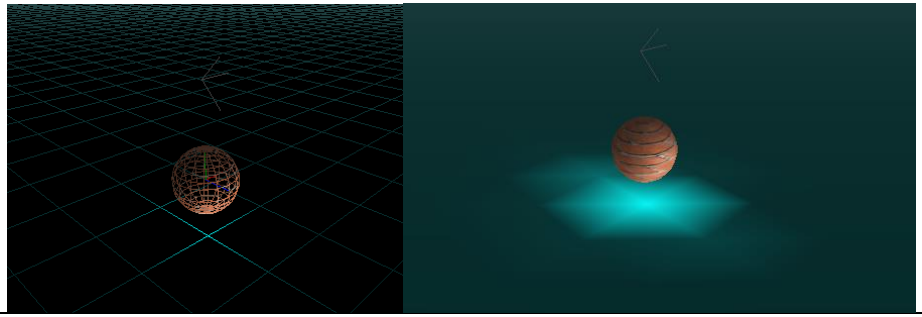
Student Name: Hanul Rheem
Student ID: 20109218

					
16/5/2023	-Making custom lightening prefab 3d object.	-light object implementation.	The spotlight has material structs, which consist of ambient, diffuse, and specular values. These values are applied to the spotlight, affecting any objects that come into proximity with it. Depending on the spotlight, the material object will change its diffuse or specular properties. This method is similar to the getTransformCustomCube method, allowing for adjustments in the rotation and position of the light through parameters. By default, the spotlight will always be rendered in a downward direction.	The color of the spotlight should be rendered in white, but it remains unchanged. There might be an issue with the "glPushMatrix" and "glPopMatrix" calls during the	6 Hours

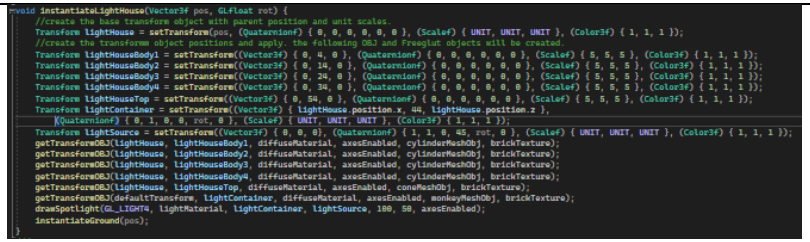
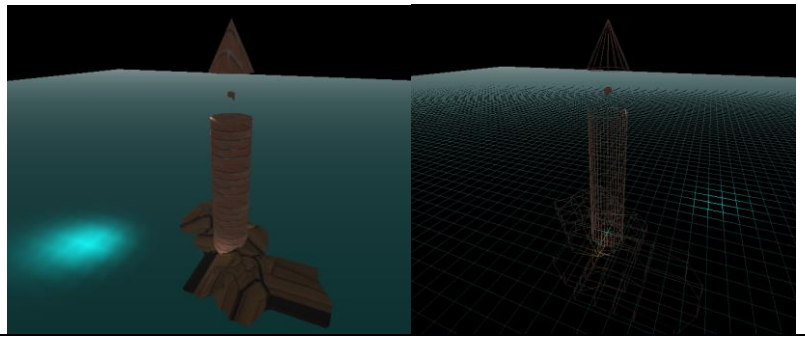
Student Name: Hanul Rheem
Student ID: 20109218

			`` void drawSpotlight(GLenum lightName, Material material, Transform parent, Transform transform,GLfloat spotExponent, GLfloat spotCutoff, int axesEnabled) { glPushMatrix(); GLfloat spotDirection[] = { 0.0, -1.0, 0.0 }; // Spotlight direction (pointing downwards) // Apply parent's position and rotation glTranslatef(parent.position.x, parent.position.y, parent.position.z); glRotatef(parent.rotation.angleY, 0, parent.rotation.rotY, 0); glRotatef(parent.rotation.angleZ, 0, 0, parent.rotation.rotZ); glRotatef(parent.rotation.angleX, parent.rotation.rotX, 0, 0); if (axesEnabled) { drawTransformAxes(transform.position, transform.rotation, (Scalef) { 1, 1, 1 }, 0.5f); } //get the transform positions and rotations, colors. glRotatef(transform.rotation.angleX, transform.rotation.rotX, 0, 0); glRotatef(transform.rotation.angleY, 0, transform.rotation.rotY, 0); glRotatef(transform.rotation.angleZ, 0, 0, transform.rotation.rotZ); glScalef(transform.scale.x, transform.scale.y, transform.scale.z); // Set spotlight material properties GLfloat ambientMaterial[] = { material.ambient.r, material.ambient.g, material.ambient.b, material.ambient.a }; GLfloat diffuseMaterial[] = { material.diffuse.r, material.diffuse.g, material.diffuse.b, material.diffuse.a }; GLfloat specularMaterial[] = { material.specular.r, material.specular.g, material.specular.b, material.specular.a }; glLightfv(lightName, GL_AMBIENT, ambientMaterial); glLightfv(lightName, GL_DIFFUSE, diffuseMaterial); glLightfv(lightName, GL_SPECULAR, specularMaterial); // Enable lighting and spotlight glEnable(GL_LIGHTING); glEnable(lightName); glEnable(GL_NORMALIZE); glEnable(GL_DEPTH_TEST); glEnable(GL_COLOR_MATERIAL); // Set spotlight properties glLightfv(lightName, GL_SPOT_DIRECTION, spotDirection); glTranslatef(transform.position.x, transform.position.y, transform.position.z); GLfloat lightPosition[] = { transform.position.x, transform.position.y, transform.position.z, 1.0 }; glLightfv(lightName, GL_POSITION, lightPosition); // Rotate the spotlight direction GLfloat lightDir[] = { 0.0, -1.0, 0.0 }; // Initialize the spotlight direction glMultMatrixf(lightDir); // Apply the rotation to the spotlight direction vector glLightf(lightName, GL_SPOT_EXPONENT, spotExponent); glLightf(lightName, GL_SPOT_CUTOFF, spotCutoff); // Restore previous matrix state glPopMatrix(); } ``` You can observe that the light is rendered on the ground plane. However, due to the ground plane's lack of smoothness and insufficient number of vertices, the light resolution appears to be quite low, resulting in a less visually pleasing rendering. 	spotlight rendering process. 	
17/5/2023	-Object that contains textures -Custom texture mapping. -Custom object material		I utilized the lecturer's Obj Loader to load a custom 3D object, which also includes material preferences in the method. Implementing this method was relatively straightforward and didn't require much time compared to the previous method. The process of applying material settings is similar to the drawspotlight method. The only distinction is enabling GL_COLOR_MATERIAL and GL_AMBIENT_AND_DIFFUSE to obtain the diffuse and ambient values.	There were no significant issues to report, as most of the problems encountered were related to memory type mismatches and uninitialized variables.	3hours

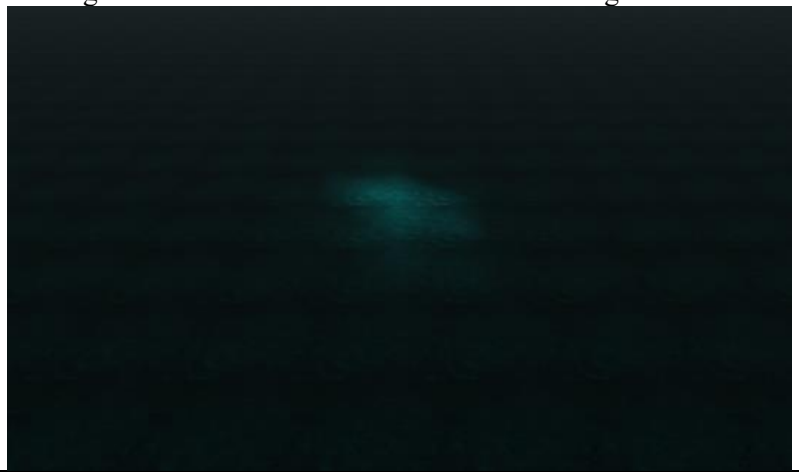
Student Name: Hanul Rheem
Student ID: 20109218

	-Enabling fog for my environment scene.		 The light is rotated at a 45-degree angle, and the sphere object is textured with a brick texture map. The default material setting is configured as a diffuse material. 		
23/5/2023	-making animated light house.		The lighthouse features an animated light object with a flying monkey statue at the top of the lightbox. The light is rendered in a downward direction and rotates 360 degrees every frame. Implementing this method was not overly difficult; it mainly involved adjusting the scale and rotation of the object. The method includes a "rot" parameter that takes the real-time rotation value.		3hours

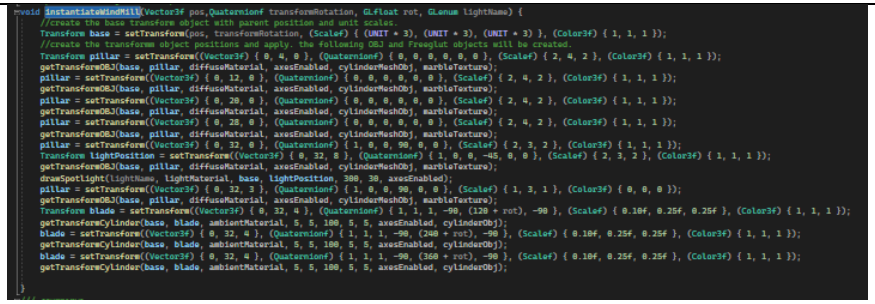
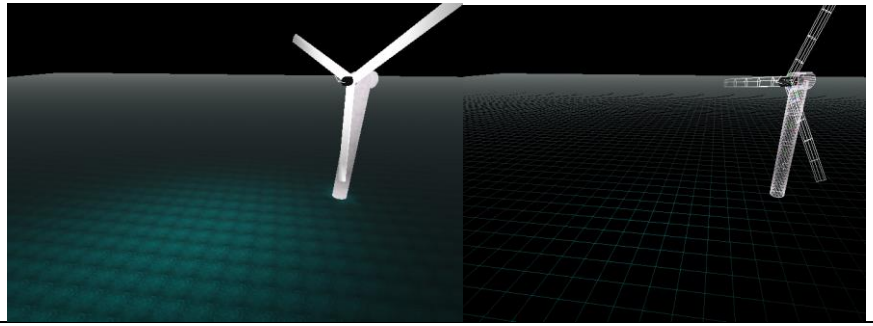
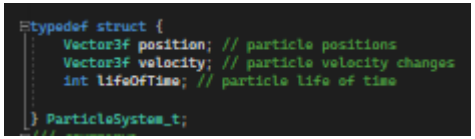
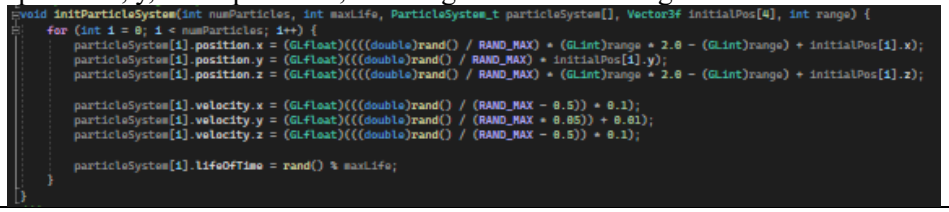
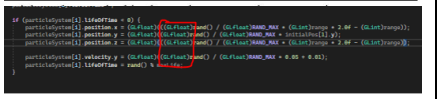
Student Name: Hanul Rheem
Student ID: 20109218

			 Example footage of the light house. If I used the OBJ model to render it, would have the best result to make it look decent. 		
24/5/2023	-making animated texture map ground.		The ground is animated with texture, which includes material struct values. The UV coordinates of the texture have rotation values added and divided by 360 degrees, allowing it to move left and downwards. As depicted, the ground panel has its normals calculated, and the material values are applied seamlessly, resulting in a visually pleasing appearance.		3hours

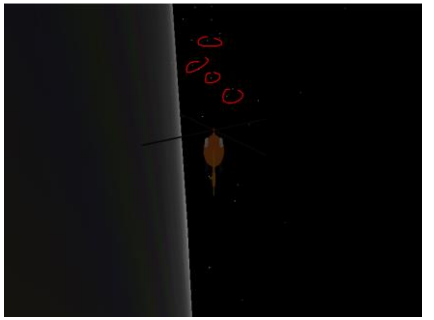
Student Name: Hanul Rheem
Student ID: 20109218

			`` void drawGround(Material material,Color3f color, Vector3f position, int numQuads, GLuint textureID, GLfloat rotationSpeed) { const float unitSize = UNIT * 5.0f; // Unit size in meters const float textureScale = 1.0f; // Scale factor for texture coordinates float startX = position.x - (numQuads / 2) * unitSize; float startZ = position.z - (numQuads / 2) * unitSize; glPushMatrix(); glTranslatef(position.x, position.y, position.z); renderFillEnabled ? glPolygonMode(GL_FRONT_AND_BACK, GL_FILL) : glPolygonMode(GL_FRONT_AND_BACK, GL_LINE); glEnable(GL_TEXTURE_2D); // Enable texturing // Loop through the quad planes and draw them for (int i = 0; i < numQuads; i++) { for (int j = 0; j < numQuads; j++) { // Calculate the position and size of the current quad plane float quadX = startX + i * unitSize; float quadZ = startZ + j * unitSize; float quadSize = unitSize; GLfloat ambientMaterial[4] = { material.ambient.r, material.ambient.g, material.ambient.b, material.ambient.a }; GLfloat diffuseMaterial[4] = { material.diffuse.r, material.diffuse.g, material.diffuse.b, material.diffuse.a }; GLfloat specularMaterial[4] = { material.specular.r, material.specular.g, material.specular.b, material.specular.a }; glMaterialfv(GL_FRONT, GL_AMBIENT, ambientMaterial); glMaterialfv(GL_FRONT, GL_DIFFUSE, diffuseMaterial); glMaterialfv(GL_FRONT, GL_SPECULAR, specularMaterial); glMaterialf(GL_FRONT, GL_SHININESS, material.shininess); glEnable(GL_COLOR_MATERIAL); // Set the color to white (for texture) glColor3f(color.r, color.g, color.b); // Calculate the texture coordinates float texCoordX1 = (i + rotationSpeed / 360.0f) * textureScale; float texCoordX2 = texCoordX1 + textureScale; float texCoordY1 = (j + rotationSpeed / 360.0f) * textureScale; float texCoordY2 = texCoordY1 + textureScale; // Bind the texture for the current quad glBindTexture(GL_TEXTURE_2D, textureID); // Draw the quad plane with texture coordinates glBegin(GL_QUADS); glNormal3f(0.0f, 1.0f, 0.0f); glTexCoord2f(texCoordX1, texCoordY1); glVertex3f(quadX, position.y, quadZ); glTexCoord2f(texCoordX2, texCoordY1); glVertex3f(quadX + quadSize, position.y, quadZ); glTexCoord2f(texCoordX2, texCoordY2); glVertex3f(quadX + quadSize, position.y, quadZ + quadSize); glTexCoord2f(texCoordX1, texCoordY2); glVertex3f(quadX, position.y, quadZ + quadSize); glEnd(); } } glDisable(GL_COLOR_MATERIAL); ``` Example footage of ground plane with the light reflection. If the ground plane was high resolution or smooth it would make the light smoother. 		
30/5/2023	-Making scene assets windmill.		Creating this asset was a straightforward process. I utilized my custom-built asset to construct the windmill, and it involved simply applying rotation values to the quaternionf, which controls the x, y, and z rotation coordinates. The light object is positioned at the center point of the propeller blades.		2hours

Student Name: Hanul Rheem
Student ID: 20109218

			 Here is an example of a windmill with wireframe visualization. Although utilizing a proper OBJ model obtained from the internet would enhance the visual quality, it would be necessary to explore methods for animating the model. 		
31/5/2023	- Implementing 3D Particle system for the helicopter.		This is the structure type for the particle system, which bears resemblance to a 2D snow particle system. The main distinction lies in the inclusion of a Vector3f structure, encompassing position and velocity values for the x, y, and z axes.  Before utilizing the initParticleSystem method, it is crucial to instantiate the particle system locations. These locations are randomly generated within specific x, y, and z positions, allowing for a defined range.. 	In the console, numerous typecast errors occurred, prompting me to address them by adding GLfloat to resolve the warnings..   The implementation of the particle system was relatively straightforward, mainly based on the assignment1 script. However, an issue arose where the particle system renders outside of the	10 hours

Student Name: Hanul Rheem
Student ID: 20109218

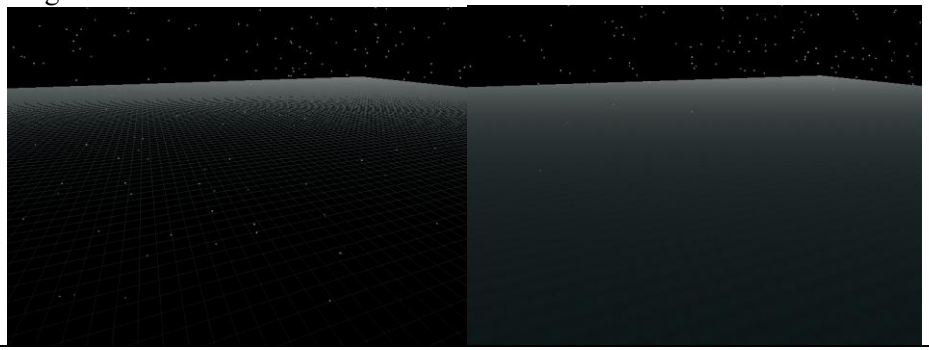
			The particle update method shares similarities with the snow assignment 1 method. It takes the velocity values for the x, y, and z axes and updates the positions of each particle accordingly. When the lifetime of a particle reaches zero, it is immediately regenerated at random positions within the defined range. Each particle will be rendered with its updated position, resulting in a smooth appearance that closely resembles a snow particle effect. Here is an example of the snow particle scenes, including one without the wireframe. In the future, incorporating multiple shapes of objects within the particle system would enhance its visual appeal and provide a more diverse range of effects.	skybox, causing undesired visual effects. 	
5/6/2023	-making skybox.		The process of instantiating the skybox was relatively simple. All that was required was to apply a texture map to the created cube and scale it up accordingly. To achieve the desired effect of a foggy scene, I utilized a white		1 hour

```
void updateParticleSystem(int numParticles, double gravity,int maxLife, Vector3f initialPos[4], ParticleSystem_t particleSystem[], int range) {
    for (int i = 0; i < numParticles; i++) {
        particleSystem[i].position.y += particleSystem[i].velocity.y;
        particleSystem[i].velocity.y += (GLfloat)gravity;
        particleSystem[i].lifeOfTime--;

        if (particleSystem[i].lifeOfTime < 0) {
            particleSystem[i].position.x = (GLfloat)(((GLfloat)rand() / (GLfloat)RAND_MAX * (GLint)range * 2.0f - (GLint)range));
            particleSystem[i].position.y = (GLfloat)((GLfloat)rand() / (GLfloat)RAND_MAX * initialPos[1].y);
            particleSystem[i].position.z = (GLfloat)(((GLfloat)rand() / (GLfloat)RAND_MAX * (GLint)range * 2.0f - (GLint)range));

            particleSystem[i].velocity.y = (GLfloat)((GLfloat)rand() / (GLfloat)RAND_MAX * 0.05 + 0.01);
            particleSystem[i].lifeOfTime = rand() % maxLife;
        }
    }
}
```

```
void drawParticleSystem(int numParticles, Vector3f initialPos[4], ParticleSystem_t particleSystem[], Color3f color,Vector3f pos) {
    glPushMatrix();
    glEnable(GL_COLOR_MATERIAL);
    glEnable(GL_POINT_SMOOTH);
    glPointSize(1.0);
    glBegin(GL_POINTS);
    for (int i = 0; i <= numParticles; i++) {
        glColor3f(color.r, color.g, color.b);
        glVertex3f((particleSystem[i].position.x + pos.x), particleSystem[i].position.y, (particleSystem[i].position.z + pos.z));
    }
    glEnd();
    glDisable(GL_POINT_SMOOTH);
    glDisable(GL_COLOR_MATERIAL);
    glPopMatrix();
}
```

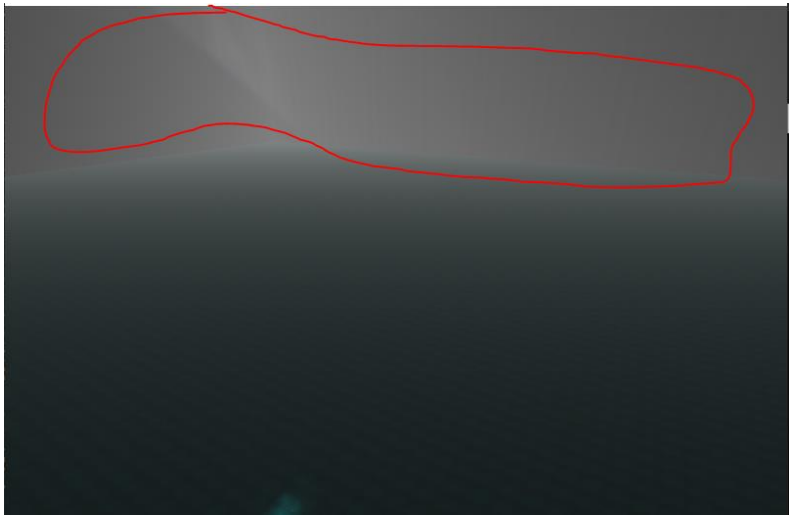


Student Name: Hanul Rheem
Student ID: 20109218

image as the texture.

```
void instantiateSkyBox(Vector3f pos, Scalef scale) {
    //create the base transform object with parent position and unit scales.
    Transform base = setTransform(pos, (Quaternionf) { 0, 0, 0, 0, 0, 0 }, (Scalef) { UNIT, UNIT, UNIT }, (Color3f) { 1, 1, 1 });
    Transform skybox = setTransform((Vector3f) { 0, 0, 0 }, (Quaternionf) { 0, 0, 0, 0, 0, 0 }, scale, (Color3f) { 1, 1, 1 });
    //create cube OBJ.
    getTransformOBJ(base, skybox, diffuseMaterial, axesEnabled, cubeMeshObj, nightSkyBoxTexture);
}
```

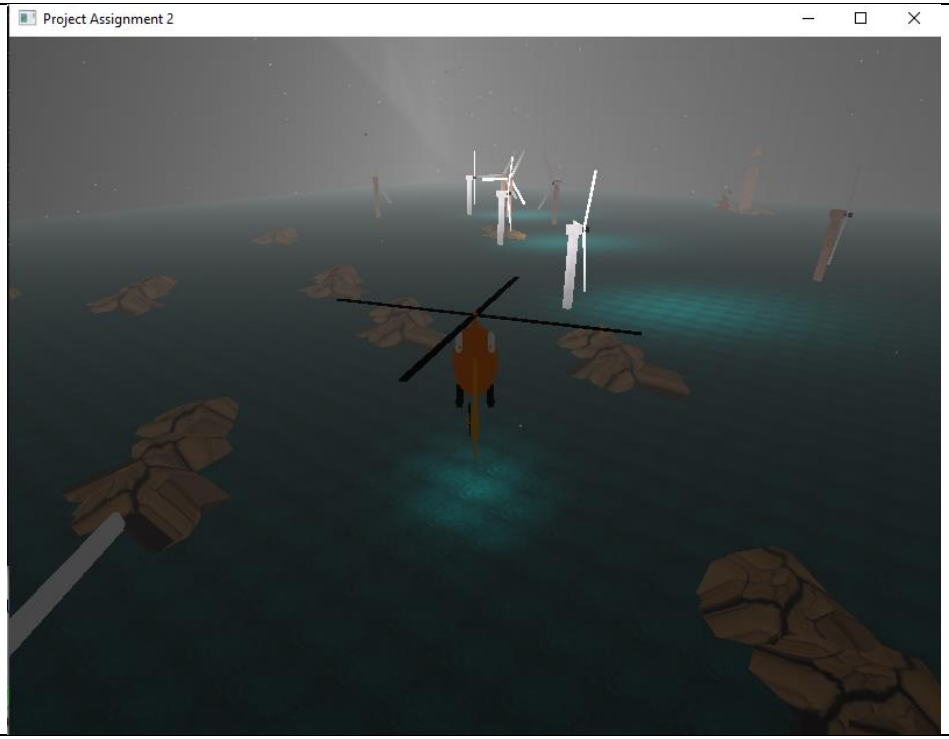
Here is an example scene of the skybox. If the skybox had a higher-resolution texture, it would greatly enhance the visual quality and overall appearance of the scene.



Conclusion:
Overall, Assignment 2 was not particularly straightforward when it came to implementing the objects. However, the use of a modular system for instantiating objects greatly facilitated the process. The particle system, largely derived from Assignment 1, encountered several bugs related to the coordination of velocities in the x, y, and z axes. Nonetheless, incorporating light sources and material preferences proved to be relatively manageable. The main challenge arose in ensuring the helicopter model's visual integrity and achieving flawless functionality for the particle system.

Here’s my final scene of the project.

Student Name: Hanul Rheem
Student ID: 20109218

					
Total					49 Hours