Names and Student Numbers: Minn Khant (Scotty) Maw (23478063), Silvi Claudia (24108363)

We were unfortunately only able to complete Part A, B, C, and E. However, we were able to get Part D to work partially; the diffuse setting is working.

A lot of the implementations we were able to do were similar to some of the lab work. A lot of the thought process behind the implementations were taken from the labs.

The following describes the changes made to the skeleton code in order to make the functionalities work as intended.

Part A

Rotation Around Focus Point, Panning (Middle Mouse Drag), Zoom (Scroll Wheel), Reset Functionality,Projection Matrix, ImGui Controls

1) We started by using the yaw and pitch angles to calculate a rotated direction vector. The camera's position is calculated relative to a focus point, and the view matrix is created using glm::lookAt.

```
glm::mat4 rotation_matrix = glm::rotate(yaw, {0,1,0}) * glm::rotate(pitch, {-1,0,0});
glm::vec3 forward = glm::vec3{rotation_matrix[2]};
glm::vec3 position = focus_point - forward * distance;
view_matrix = glm::lookAt(position, focus_point, UP);
```

2) Dragging the middle mouse button shifts the focus point based on the current camera orientation:

```
focus_point += right_vector * pan.x * pan_scale;
focus_point += up_vector * pan.y * pan_scale;
```

3) Scroll input adjusts the distance from the camera to the focus point:

```
distance -= ZOOM_SCROLL_MULTIPLIER * ZOOM_SPEED * scroll;
```

4) Pressing the R key calls reset(), restoring initial values of yaw, pitch, distance, etc.
5) Replaced glm::infinitePerspective with a manual perspective matrix that uses both near and far:

```
projection_matrix = glm::perspective(fov, aspect_ratio, near, far);
```

6) Integrated the following UI controls for real-time control of all camera parameters and help with debugging and visualization.

```
ImGui::SliderFloat("Near Plane", &near, 0.001f, 1.0f, "%.5f", ImGuiSliderFlags_Logarithmic);
ImGui::SliderFloat("Far Plane", &far, 1.0f, 2000.0f);
ImGui::SliderFloat("Pitch", &pitch_deg, -89.99f, 89.99f);
ImGui::DragFloat("Yaw", &yaw_deg);
```

Part B

- Need scene objects to rotate around their local axes using Euler angles, with rotation applied in a sensible order and visible via the UI.

1) Euler Rotation Field

We store the rotation as a glm::vec3 in radians:

glm::vec3 euler_rotation;

2) ImGui Integration for Rotation

We display rotation sliders in degrees, then convert them to radians before applying:

```
glm::vec3 euler_rotation_degrees = glm::degrees(euler_rotation);
ImGui::DragFloat3("Rotation", &euler_rotation_degrees[0]);
euler_rotation = glm::radians(glm::mod(euler_rotation_degrees, 360.0f));
```

3) Matrix Composition in calc_model_matrix()

The model matrix includes translation, rotation (Z to Y to X), and scaling:

```
glm::mat4 translation_matrix = glm::translate(glm::mat4(1.0f), position);
glm::mat4 rotation_x = glm::rotate(glm::mat4(1.0f), euler_rotation.x, {-1, 0, 0});
glm::mat4 rotation_y = glm::rotate(glm::mat4(1.0f), euler_rotation.y, { 0, 1, 0});
glm::mat4 rotation_z = glm::rotate(glm::mat4(1.0f), euler_rotation.z, { 0, 0, -1});
glm::mat4 scale_matrix = glm::scale(glm::mat4(1.0f), scale);

return translation_matrix * rotation_z * rotation_y * rotation_x * scale_matrix;
```

4) Rotation Axes Behavior
   X-axis: tilts forward/backward (roll)
   Y-axis: rotates around vertical (yaw)
   Z-axis: spins around screen axis (pitch)
5) Real-Time Transformation

As the user modifies the sliders, the entity's appearance updates instantly in the scene.

Part C

We started to fix the issue where objects near the camera would get clipped and disappear when zoomed in. This was due to the near clipping plane being too far from the camera in the default projection matrix.

By modifying the PanningCamera class and adding near constants in every possible method of FlyingCamera.cpp, we allowed the user to zoom in very closely to objects without them being clipped.

Also, rendering a triangle entity can be through directly in front of the camera, and it remains visible at close range.

In BasicSceneStatic.cpp:

```
std::vector<EntityRenderer::VertexData> triangle_vertices = {
  {{-1.0f, 0.0f, 1.0f}, {0, 1, 0}, {0, 0}},   // position, normal, UV
  {{ 0.0f, 1.0f, 0.0f}, {0, 1, 0}, {0.5f, 1}},
  {{ 1.0f, 0.0f, 1.0f}, {0, 1, 0}, {1, 0}},
};
```

In PanningCamera.cpp:

We replaced glm::infinitePerspective with glm::perspective, which allows us to set both near and far clipping planes: This can change:

1. Allows zooming in very closely or small near.

2. Improves depth accuracy across near and far ranges.

The code:
float aspect_ratio = window.get_framebuffer_aspect_ratio();
float far = 1000.0f;
projection_matrix = glm::perspective(fov, window.get_framebuffer_aspect_ratio(), near, far);
inverse_projection_matrix = glm::inverse(projection_matrix);

Moreover, prevent rendering errors from extremely small values to maintain a stable depth of buffer.
near = clamp(near, 0.00001f, 10.0f);

ImGui Integration (control the near and far planes during runtime):
ImGui::SliderFloat("Near Plane", &near, 0.001f, 1.0f, "%.5f", ImGuiSliderFlags_Logarithmic);
ImGui::SliderFloat("Far Plane", &far, 1.0f, 2000.0f, "%.1f");

Part E
Step 1: Add texture_scale field with default
In Instance Data wherever you define per-entity material data (likely in EntityMaterial or InstanceData struct):
glm::vec2 texture_scale = glm::vec2(1.0f, 1.0f); // default scale = no change

Step 2: Apply texture scaling in your shader
In your fragment shader (lights.glsl or similar):
vec2 scaled_uv = texture_coordinate * material.texture_scale;
vec3 texture_colour = texture(diffuse_texture, scaled_uv).rgb;
vec3 specular_colour = texture(specular_map, scaled_uv).rgb;

CITS3003 - Project Part E: Texture Scaling Implementation
Goal:
Add the ability to scale the textures (diffuse and specular) for EntityElement and AnimatedEntityElement.
Step 1: Data Structure
glm::vec2 diffuse_texture_scale;
glm::vec2 specular_texture_scale;
glm::vec2 emission_texture_scale;

Step 2: ImGui UI Controls
ImGui::DragFloat2("Diffuse Texture Scale", &material.diffuse_texture_scale[0], ...);
ImGui::DragFloat2("Specular Texture Scale", &material.specular_texture_scale[0], ...);
ImGui::DragFloat2("Emission Texture Scale", &material.emission_texture_scale[0], ...);

Step 3: JSON Save & Load
// Save

{"diffuse_texture_scale", material.diffuse_texture_scale},
{"specular_texture_scale", material.specular_texture_scale},
{"emission_texture_scale", material.emission_texture_scale},

// Load
if (m.contains("diffuse_texture_scale")) ...
if (m.contains("specular_texture_scale")) ...

Step 4: SceneElement & Transform Support

Handled properly in the LocalTransformComponent and the generic structure. This supports use within entities and across children recursively if needed.