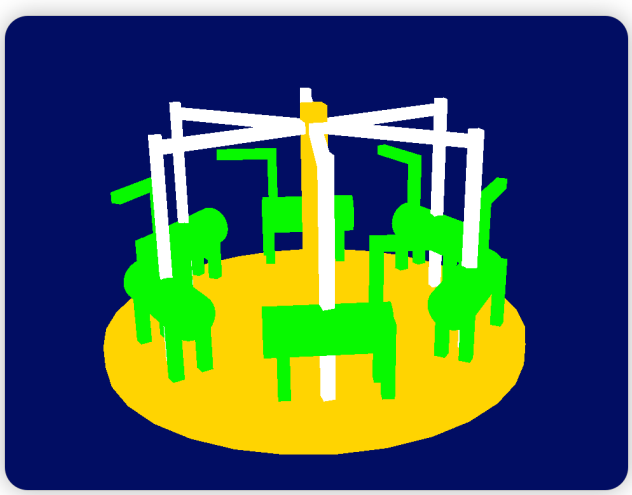
Carousel Project Report
12204400, 장철희

1. Screenshots and Implementation Details

OpenGL was used to implement the basic carousel project. This included the construction of the carousel's structure (legs, head, body, handrails), the creation of the carousel scene (central pillar, base), and the implementation of the functionality for the carousel to rotate around the pillar. Additionally, camera control using wasdqe to manipulate the camera's viewpoint was implemented (refer to camera.hpp).

Images for reference:



2. Description and Location of Key Code

main.cpp

- **Function**: `main.cpp` serves as the main entry point of the program. It initializes the window, shaders, models, and contains the game loop responsible for rendering the scene and handling user input.
- **Code Location and Description**:
  1. **Time Control** - Used to calculate the time difference between frames for controlling animation speed.
     - Code Location: Line 6
     - Code Snippet: `float deltaTime = 0.1f; // Time difference between the current frame and the last frame`
  2. **Window Creation** - Initializes the OpenGL window.
     - Code Location: Line 13
     - Code Snippet: `cg::Window window{}; // Create a window object`
  3. **Shader Loading** - Loads vertex and fragment shaders.
     - Code Location: Line 15
     - Code Snippet: `cg::Shader forward_shader("res/shaders/forward.vert", "res/shaders/forward.frag");`
  4. **Model Loading** - Loads cube and cylinder models, which might be part of the carousel.
     - Code Location: Lines 18 and 19
     - Code Snippet:

        - `cg::Model cube_model("res/models/cube.obj");  // Load cube model`
        - `cg::Model cylinder_model("res/models/cylinder.obj");  // Load cylinder model`
    5. **Model Matrix Initialization** - Initializes the position and orientation of different models.
        - Code Location: Lines 21 to 34
        - Code Snippet:
            - `cg::ModelMatrix column_model_matrix{ glm::vec3{0.0f}, ...`
            - `std::vector<cg::ModelMatrix> hz_model_matrix(3, cg::ModelMatrix{ glm::vec3(0.0f, 1.0f, 0.0), ...`
            - `cg::ModelMatrix const panel_model_matrix{ ...`
            - `cg::ModelMatrix support_model_matrix{ ...`

    These sections are core components directly related to the implementation of the carousel in the `main.cpp` file. Each code snippet plays a crucial role in creating the entire scene, from creating the window and loading shaders to initializing and placing different models.

shader: forward.vert and forward.frag

- forward.vert (Vertex Shader)
    1. **Input Vertex Attributes** - Defines the inputs of the vertex shader.
        - Code Location: Lines 3-5
        - Code Snippet:
            - `layout (location = 0) in vec3 pos;` - Vertex position.
            - `layout (location = 1) in vec3 normal;` - Vertex normal.
            - `layout (location = 2) in vec2 texcoord;` - Texture coordinates.
    2. **Uniform Variables** - Defines the uniform variables passed to the vertex shader.
        - Code Location: Lines 11-14
        - Code Snippet:
            - `uniform mat4 model;` - Model matrix.
            - `uniform mat4 viewProjection;` - View-projection matrix.
            - `uniform mat3 normMat33;` - Normal matrix.
    3. **Main Function** - The main execution part of the vertex shader.
        - Code Location: Line 17
        - Code Snippet: `void main()`
- forward.frag (Fragment Shader)
    1. **Input Variables** - Inputs from the vertex shader.
        - Code Location: Lines 3-5
        - Code Snippet:
            - `in vec2 v2fTexCoord;` - Passed texture coordinates.
            - `in vec3 v2fNormal;` - Passed normal vector.
            - `in vec3 v2fPos;` - Passed vertex position.

2. **Output Color** - Defines the output color of the fragment shader.
    - Code Location: Line 7
    - Code Snippet: `out vec4 fragColor;` - Final fragment color.
3. **Uniform Variables** - Defines the uniform variables passed to the fragment shader.
    - Code Location: Lines 9-11
    - Code Snippet:
        - `uniform vec3 camPos;` - Camera position.
        - `uniform vec3 diffuseColor;` - Diffuse color.
4. **Main Function** - The main execution part of the fragment shader.
    - Code Location: Line 13
    - Code Snippet: `void main()`
        - In the main function, lighting calculations and color blending are performed, ultimately determining the color of each pixel.

3. Implementation of Additional Feature: Camera Viewpoint Control in camera.hpp

1. **Camera Class Definition** - Defines the camera class along with its properties and methods.
    - Code Location: Line 11
    - Code Snippet: `class Camera { ... }`
2. **CameraMovement Enumeration** - Defines possible camera movement directions.
    - Code Location: Line 20
    - Code Snippet: `enum class CameraMovement { FORWARD, BACKWARD, LEFT, RIGHT, UP, DOWN };`
3. **processKeyboard Method** - Handles keyboard input to control camera movement.
    - Code Location: Line 53 and subsequent lines
    - Code Snippet:
        - `void processKeyboard(CameraMovement direction, float delta_time) { ... }`
        - This method updates the camera's position based on the input direction and time difference.
        - Different directions such as `FORWARD`, `BACKWARD`, `LEFT`, `RIGHT`, `UP`, `DOWN` have different handling logic.