LAB1_assignment_report

OpenGL Code Walkthrough - Cake on a Plate

This code demonstrates how to render a scene with a "cake on a plate" using OpenGL. The "plate" is represented by a triangleVerticesArray, and the "cake" by a squareVerticesArray.

1. Setting Up OpenGL and GLFW

- **Libraries**: The program includes OpenGL libraries (`glad` for loading functions, `GLFW` for window management) and `glm` for transformations.
- **Constants**:
  - `SCR_WIDTH` and `SCR_HEIGHT` set the screen width and height.
- **Transformation Variables**:
  - Variables control the **rotation**, **translation**, and **scaling** of the plate (triangle) and cake (square) shapes.

2. Shader Code

Two shaders are defined:

- **Vertex Shader**:
  - Accepts vertex position data (`aPos`).
  - Applies a transformation matrix (`transform`) to the position of each vertex.
- **Fragment Shader**:
  - Assigns a uniform color (`objectColor`) to each shape.

3. Initialization in `main` Function

- **GLFW Initialization**:
  - Sets the OpenGL version and profile, then creates a window with the specified width and height.
- **Shader Compilation**:
  - Vertex and fragment shaders are created and compiled.
  - The shaders are linked into a shader program, which is then activated for use.

4. Defining Shape Vertices

- **Triangle (Plate) Vertices**:

- The `triangleVertices` array contains coordinates for a circular shape (simulating a plate).

- **Square (Cake) Vertices**:
  - The `squareVertices` array contains vertices for a smaller square shape, representing a cake.

5. Generating Buffers and Vertex Arrays

- **VAOs and VBOs**:
  - Two VAOs (Vertex Array Objects) and two VBOs (Vertex Buffer Objects) are created for the cake and plate.
- **Triangle VAO**:
  - Links the triangle vertex data to the shader.
- **Square VAO**:
  - Links the square vertex data to the shader.

6. Render Loop

The main rendering loop updates the screen continuously while the window is open.

- **Screen Clearing**:
  - Sets and clears the background color on each loop iteration.

Drawing the Plate (Triangle Shape)

- **Triangle Transformations**:
  - A 4x4 transformation matrix (`transformTriangle`) is created using `glm`.
  - This matrix applies translation, rotation, and scaling transformations based on the plate transformation variables.
- **Color and Drawing**:
  - The shader program sets the triangle color to red (`glUniform3f(colorLoc, 1.0f, 0.0f, 0.0f)`).
  - The transformation matrix is passed to the vertex shader, and the plate is drawn as a `GL_TRIANGLE_FAN`.

Drawing the Cake (Square Shape)

- **Square Transformations**:
  - Another transformation matrix (`transformSquare`) is created for the square with separate transformations.
- **Color and Drawing**:

- The shader program sets the square color to yellow (`glUniform3f(colorLoc, 1.0f, 1.0f, 0.0f)`).
- The transformation matrix is passed to the shader, and the cake is drawn with `GL_TRIANGLE_FAN`.

7. Input Handling in `processInput` Function

- **Close Window**:
  - Pressing `ESC` closes the window.
- **Plate (Triangle) Controls**:
  - **Rotation**: Pressing `R` or `T` Rotates plate.
  - **Translation**: Uses arrow keys for up, down, left, or right movement.
  - **Scaling**: Uses `=` to enlarge, `-` to shrink.
- **Cake (Square) Controls**:
  - **Rotation**: `Y` and `U` rotate.
  - **Translation**: `W`, `A`, `S`, `D` move the square.
  - **Scaling**: `O` enlarges, `P` shrinks.

8. Framebuffer Resize Callback

The `framebuffer_size_callback` function updates the viewport to match new window dimensions, ensuring rendering adapts to resizing.

9. Cleanup

Once the rendering loop exits, the program deletes all buffers and terminates GLFW, freeing resources.

Summary

This code draws two shapes:

1. **Red triangle (plate)**: Configurable transformations for positioning and size.
2. **Yellow square (cake)**: Positioned above the plate with independent transformations.

The user can move, rotate, and scale both shapes using keyboard controls, providing a simple example of OpenGL rendering with transformation matrices from `glm`.