Scientific Visualization

OpenGL – Robot

Anand Kamble
Department of Scientific Computing
Florida State University

1 Introduction

In this project, we will be creating a robot arm in OpenGL using the transformation matrices developed in the first homework. We also add the functionality to rotate the arm using key presses.

2 Execution and Usage

To compile and run the code, you can use the Makefile included with the project. You can directly start the program by using the following command which will compile and run the binary.

```
    make run
```

You can control the robot arm by following keys:

- `'S'` - Rotate the lower arm.
- `'E'` - Rotate the upper arm.
- `'O'` - Open the finger.
- `'C'` - Close the finger.

And to rotate the camera around the Y axis use the `'R'` key.

3 Implementation

3.1 Projection Matrix

Here we are using the function provided by the GLM library to create the projection matrix. We are creating a perspective Projection matrix. The code implementation is like below:

```
mat4 projection = perspective(radians(45.0f), SCR_WIDTH/SCR_HEIGHT,
   0.1f, 100.0f);
ourShader.setMat4("projection", projection);
```

3.2 Camera Position

We are setting camera position using the view matrix, which is created by using `lookAt` function from GLM. We are also calculating the X and Z position of the camera since we are rotating it around the Y axis.

```
    mat4 view = mat4(1.0f);
    float radius = 10.0f;
    float camX = static_cast<float>(sin(cameraAngle) * radius);
    float camZ = static_cast<float>(cos(cameraAngle) * radius);
    view = lookAt(vec3(camX, 0.0f, camZ), vec3(0.0f, 0.0f, 0.0f),
       vec3(0.0f, 1.0f, 0.0f));
    ourShader.setMat4("view", view);
```

3.3 Lower Arm, Upper Arm, and Fingers

For each part of the robot we are following the same structure of code to render and calculate the transformation matrix.

The structure is as follows;

```
model = mat4(1.0f); // Identity Matrix
// ... Perform all the required transformations.
ourShader.setMat4("model", model);// Send updated matrix to shader.
glDrawArrays(GL_TRIANGLES, 0, 36);// Render the arm/finger (Cube)
```

3.4 User Inputs

We are handling all the user inputs from the function `processInput`. In this function, we check if a key is pressed and if it is, we are updating the respective angles.

In this part, we are also making sure that the angle between the fingers doesn't exceed 90°. This is done by adding following conditions in the code:

```
    if (FingerAngle < 45.0f)
    if (FingerAngle > 0.0f)
```

4 Results

The program is able to render the robot arm, and can also handle user inputs and update the robot arm accordingly.

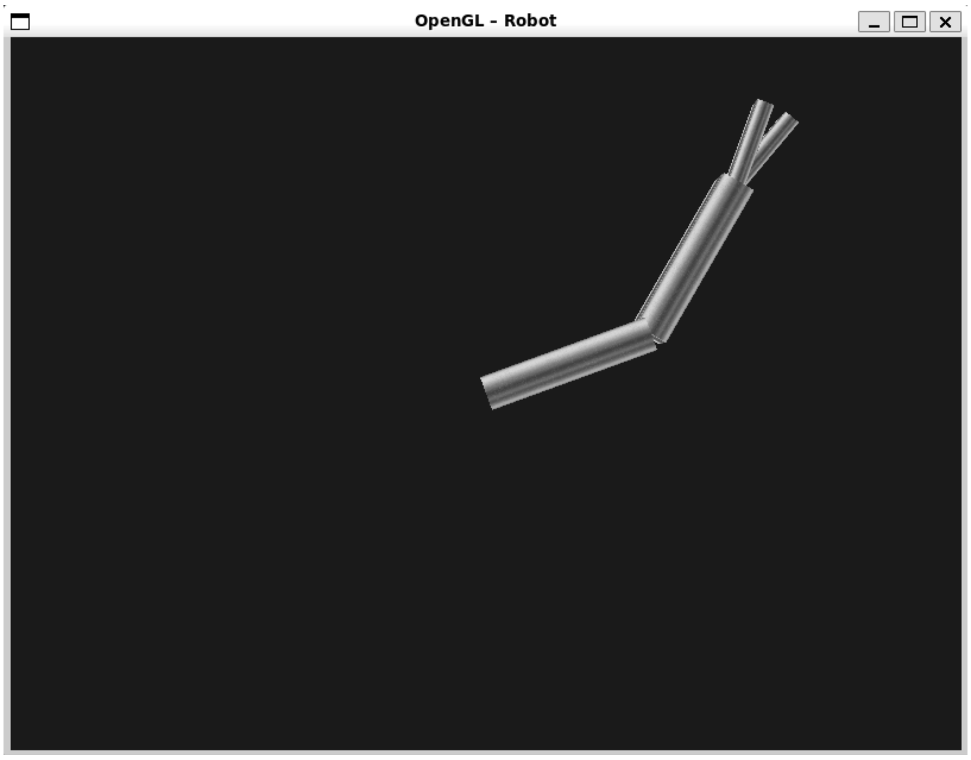
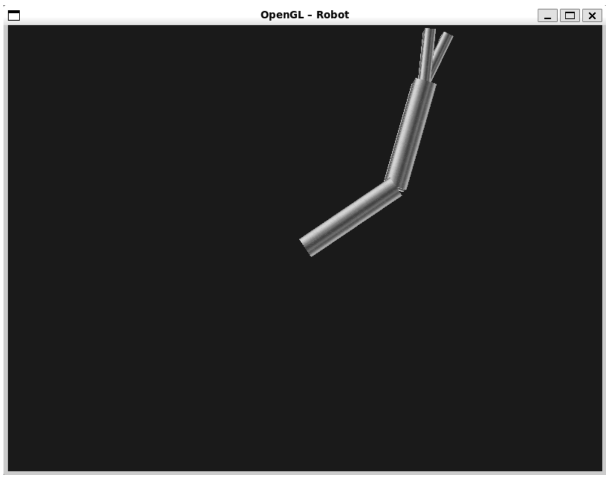
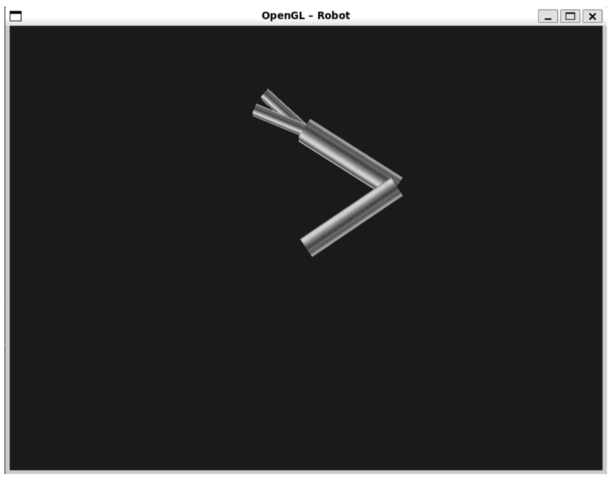


Figure 1: Output Window

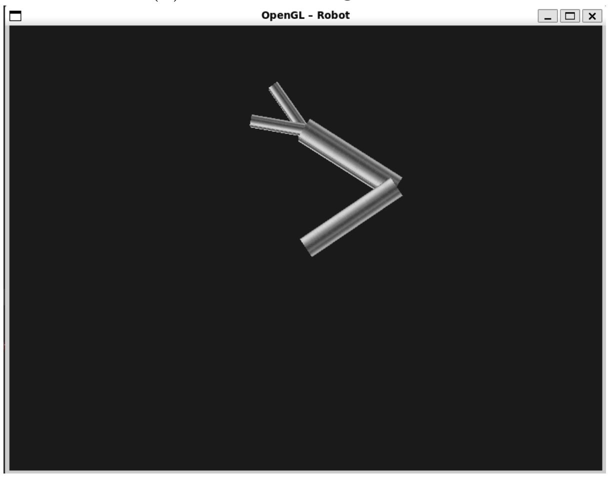
Here is how the output looks after providing the user inputs.



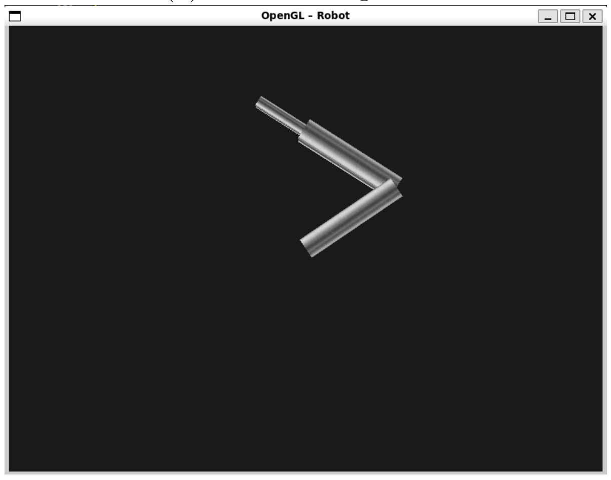
(a) After rotating shoulder



(b) After rotating elbow



(c) Fingers opened completed



(d) Fingers closed

Table 1: Results after user input

References

[1] LearnOpenGL. *Getting-started - Camera* , `https://learnopengl.com/Getting-started/Camera`.

[2] GLFW. *Keyboard key tokens* , `https://www.glfw.org/docs/latest/group__keys.html`. Dec 13 2023.