CS 405 Project 2: Textures + Illumination - Report

Ebrar Berenay Yiğit

December 21, 2023

Task 1: Handling Non-Power-of-2 Textures

Implementation in `setTexture` Function

To address the limitation of only accepting power-of-2 sized textures, the following modifications were made:

```
// Inside setTexture function
if (isPowerOf2(img.width) && isPowerOf2(img.height)) {
    // Existing code for power-of-2 textures
} else {
    gl.texParameteri(gl.TEXTURE_2D, gl.TEXTURE_WRAP_S, gl.CLAMP_TO_EDGE);
    gl.texParameteri(gl.TEXTURE_2D, gl.TEXTURE_WRAP_T, gl.CLAMP_TO_EDGE);
    gl.texParameteri(gl.TEXTURE_2D, gl.TEXTURE_MIN_FILTER, gl.LINEAR);
}
```

Listing 1: Handling Non-Power-of-2 Textures

Now, the code correctly handles non-power-of-2 sized textures by adjusting the texture parameters accordingly.

Task 2: Implementing Basic Lighting

Implementation in `MeshDrawer` Class

- **Initialization of Lighting Variables:**

```
    this.lightPosLoc = gl.getUniformLocation(this.prog, 'lightPos');
    this.normalLoc = gl.getAttribLocation(this.prog, 'normal');
    this.lightPos = [0.0, 0.0, 1.0];

```

Listing 2: Initialization of Lighting Variables

The necessary variables for handling lighting were initialized.

- **Updating `setMesh` Function:**

```
    // Inside setMesh function
    gl.bindBuffer(gl.ARRAY_BUFFER, this.normalbuffer);
    gl.bufferData(gl.ARRAY_BUFFER, new Float32Array(normalCoords), gl.
    STATIC_DRAW);

```

Listing 3: Updating `setMesh` Function

The normal buffer is now properly populated for lighting calculations.

- **Modifications in `draw` Function:**

```
    // Inside draw function
    gl.bindBuffer(gl.ARRAY_BUFFER, this.normalbuffer);
    gl.enableVertexAttribArray(this.normalLoc);
    gl.vertexAttribPointer(this.normalLoc, 3, gl.FLOAT, false, 0, 0);
```

```
```

Listing 4: Modifications in `draw` Function

The normal buffer is enabled and configured for the lighting shader.

- **Implementation of Lighting in the Fragment Shader:**

```
    // Inside fragment shader (meshFS)
    if (enableLighting) {
        // Lighting calculations
    } else {
        // No lighting, just the base color
    }

```

Listing 5: Fragment Shader for Lighting

The fragment shader now correctly computes ambient and diffuse lighting.

Additional Functionality

- The `enableLighting` function now correctly sets the uniform for enabling/disabling lighting.
- The `setAmbientLight` function sets the ambient intensity for lighting.
- Light position can be adjusted using arrow keys.

Conclusion

With these modifications, the project now supports non-power-of-2 sized textures and implements basic lighting with ambient and diffuse components. The code has been thoroughly tested, and the functionality has been verified against the provided requirements.