Task 1:

Methodology for Handling Non-Power-of-2 Sized Textures:

Create WebGL Texture:

Used WebGL's gl.createTexture() to create a new texture object.

Bind the texture using gl.bindTexture(gl.TEXTURE_2D, texture).

Set Texture Image Data:

Utilize gl.texImage2D() to set the texture image data from the HTML IMG element.

Specify parameters such as format, type, and the image itself.

Set Texture Parameters:

For power-of-2 textures:

Generate mipmaps using gl.generateMipmap(gl.TEXTURE_2D).

For both power-of-2 and non-power-of-2 textures:

Set texture wrapping to gl.CLAMP_TO_EDGE to avoid edge artifacts.

Set texture filtering for both minification and magnification to gl.LINEAR.

Handle Non-Power-of-2 Textures:

Check if the width and height are not powers of 2.

If non-power-of-2, consider using the OES_texture_float_linear extension:

Check for the extension using gl.getExtension('OES_texture_float_linear').

If supported, set appropriate texture parameters for non-power-of-2 textures.

Bind Texture to Shader:

Activate a texture unit using gl.activeTexture(gl.TEXTURE0).

Bind the texture to the unit with gl.bindTexture(gl.TEXTURE_2D, texture).

Pass the texture unit to the shader using gl.uniform1i(sampler, 0).

Task 2:

1. Initialize Lighting Variables:

Location: Constructor of the MeshDrawer class.

Purpose: Initialize variables such as light direction (lightPos), ambient light intensity (ambientLight), and a flag to enable/disable lighting (enableLightingFlag).

2. Update setMesh Function:

Location: setMesh method in the MeshDrawer class.

Purpose: Update the function to handle normal coordinates for each vertex. Create a buffer for normal coordinates and bind it to the shader program.

3. Implement enableLighting Function:

Location: enableLighting method in the MeshDrawer class.

Purpose: Toggle the lighting on/off by setting the enableLightingFlag and trigger a redraw of the scene.

4. Implement setAmbientLight Function:

Location: setAmbientLight method in the MeshDrawer class.

Purpose: Set the ambient light intensity and trigger a redraw of the scene.

5. Update Fragment Shader (meshFS):

Location: The fragment shader in your WebGL application.

Purpose: Modify the fragment shader to include lighting calculations. This involves computing ambient and diffuse lighting components and combining them with the texture color.

6. Update draw Method:

Location: draw method in the MeshDrawer class.

Purpose: Update the method to include the necessary lighting uniforms. Enable/disable lighting based on the enableLightingFlag and set the ambient light intensity.

7. Testing and Adjustments:

Location: Throughout your application.

Purpose: Test the lighting implementation, toggle lighting on/off, adjust ambient light intensity, and ensure that the lighting behaves as expected. Make adjustments as needed.

9. Documentation:

Location: Code comments and documentation.

Purpose: Document the changes made for lighting implementation, including shader modifications and any additional functions or variables introduced.