CS 405 Project 2 Report

Batur Karakaya - 28881

December 25, 2023

Task 1

For this task it was asked to complete the setTexture function to handle non power of 2 sized textures. To do so, gl.texParameteri function is used to set texture parameters.

- The first parameter of this function is the target texture (in our case "gl.TEXTURE_2D").
- The second parameter of this function specifies the parameter to set. These parameters in our case are: "gl.TEXTURE_WRAP_S", "gl.TEXTURE_WRAP_T" and "gl.TEXTURE_MIN_FILTER". The first two parameters set the texture wrapping behavior along the horizontal and vertical axes respectively. The last parameter is used when the texture is minified.
- The last parameter of this function specifies the value to set for the specified parameter. gl.CLAMP_TO_EDGE ensures that if the texture coordinates go beyond the range [0, 1], the texture will be clamped to the edge color.
  - 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);

Task 2

For this task it was asked to implement basic lighting for the scene by modifying certain parts of the code.

- Constructor: Here, we initialize the variables related to lighting. To do so, we retrieve the locations of the attributes and the uniforms specified as ”normal, ambient, lightPos and enableLighting” in the shader program, and they are set for the variables. In addition, a normal buffer is created to store the normal coordinates.
  - this.normalbuffer = gl.createBuffer();
  - this.normalLoc = gl.getAttribLocation(this.prog, ’normal’);
  - this.ambientLoc = gl.getUniformLocation(this.prog, ’ambient’);
  - this.lightPosLoc = gl.getUniformLocation(this.prog, ’lightPos’);
  - this.enableLightingLoc = gl.getUniformLocation(this.prog, ’enableLighting’);
- setMesh: Here, we handle the normal coordinates for lighting. To do so, we set up a buffer ”this.normalbuffer” and populate it with normal coordinate data ”normalCoords” by binding the buffer to gl.ARRAY_BUFFER, it’s designated as the source for attribute data. Afterwards, we use ”gl.bufferData” to initialize the buffer with the provided normal coordinates.
  - gl.bindBuffer(gl.ARRAY_BUFFER, this.normalbuffer);
  - gl.bufferData(gl.ARRAY_BUFFER, new Float32Array(normalCoords), gl.STATIC_DRAW);

- draw: Here, we bind and enable the normal buffer attributes at the beginning. Then, we calculate the normalized light direction "lightDir" based on the user input which could be changed by the arrow keys. Lastly, we set the light position uniform in the shader.
  - gl.bindBuffer(gl.ARRAY_BUFFER, this.normalbuffer);
  - gl.enableVertexAttribArray(this.normalLoc);
  - gl.vertexAttribPointer(this.normalLoc, 3, gl.FLOAT, false, 0, 0);
  - let lightDir = normalize([lightX, lightY, 0.5]);
  - gl.uniform3f(this.lightPosLoc, lightDir[0], lightDir[1], lightDir[2]);
- enableLighting: Here, we modify the "enableLighting" function to control whether the lighting is enabled or not. To do so, we enable or disable the lighting by setting the value of the "enableLighting" uniform in the current program.
  - gl.useProgram(this.prog);
  - gl.uniform1i(this.enableLightingLoc, show);
- setAmbientLight: Here, we modify the "setAmbientLight" function to set the ambient light intensity. To do so, we set the ambient light intensity by updating the value of the ambient uniform in the current program.
  - gl.useProgram(this.prog);
  - gl.uniform1f(this.ambientLoc, ambient);
- Fragment Shader: Here we calculate lighting contributions "diffuse and ambient" only if both texture rendering "showTex" and lighting "enableLighting" are enabled. First, we normalize the normal vector and the light direction vector for lighting calculations. Then, we compute the cosine of the angle between the normal and light direction. We also clamp the result to 0 using max function to ensure non-negative values, which would mean the light is coming from the opposite direction of the normal. Afterwards, we calculate the ambient lighting by scaling the ambient color with a constant white

color. We also calculate the diffuse lighting based on the dot product that was calculated previously. Lastly, we combine the texture color with the sum of diffuse and ambient lighting.

- vec3 normal = normalize(v_normal);
- lightDir = normalize(lightPos);
- float diff = max(dot(normal, lightDir), 0.0);
- vec3 ambientLighting = ambient * vec3(1.0, 1.0, 1.0);
- diffuseLighting = diff * vec3(1.0, 1.0, 1.0);
- gl_FragColor = texture2D(tex, v_texCoord) * vec4(vec3(diffuseLighting + ambientLighting), 1.0);

GitHub Link: https://github.com/Exion007/CS405/tree/main/Project2