CS405 PROJECT 2 REPORT

Name Surname: Ahmethan Özcan
ID Number: 28483

Task 1:

Thank you for sharing your code. I will provide a detailed explanation of "Task 1" and "Task 2" from the code.

If the texture is not a power of 2, the texture parameters are set differently:
- `gl.TEXTURE_WRAP_S` and `gl.TEXTURE_WRAP_T` are set to `gl.CLAMP_TO_EDGE`. This setting prevents wrapping and clamps the texture coordinates, which is necessary for non-power of 2 textures.
- `gl.TEXTURE_MIN_FILTER` and `gl.TEXTURE_MAG_FILTER` are set to `gl.LINEAR` for linear filtering without generating mipmaps.

Task 2:

```
/**
         * @Task2 : You should initialize the required variables for
lighting here
         */
        this.lightPos = gl.getUniformLocation(this.prog, 'lightPos');
        this.normalAttribLocation = gl.getAttribLocation(this.prog,
'normal');
        this.normalbuffer = gl.createBuffer();
        this.ambientLoc = gl.getUniformLocation(this.prog, 'ambient');
        this.enableLightingLoc = gl.getUniformLocation(this.prog,
'enableLighting');



////////////////////////////////////////////////////////////////////////
///////////
```

LightPos and AmbientPos and Enable LightningPos and normalAttribLocation are all positions that will be used later and buffers are memory locations that store several vertices.

```
/**
         * @Task2 : You should update the rest of this function to handle
the lighting
```

```
         */
        // this.normalbuffer = gl.createBuffer();
        gl.bindBuffer(gl.ARRAY_BUFFER, this.normalbuffer);
        gl.bufferData(gl.ARRAY_BUFFER, new Float32Array(normalCoords),
gl.STATIC_DRAW);

////////////////////////////////////////////////////////////////////////
//////////////
```

This part binds buffer to the actual gl buffer (can be think as a pointer) to use it and states the data type of the buffer.

```
/**
         * @Task2 : You should update this function to handle the lighting
         */
        gl.uniform3fv(this.lightPos, [lightX, lightY, 5.0]);
        gl.enableVertexAttribArray(this.normalAttribLocation);
        gl.vertexAttribPointer(this.normalAttribLocation, 3, gl.FLOAT,
false, 0, 0);

        ////////////////////////////////
```

First line updates the position of light with new values in shader. Second and third enables the usage of normal and states the position in buffer.

```
/**
         * @Task2 : You should implement the lighting and implement this
function
         */
        gl.useProgram(this.prog);
        gl.uniform1f(this.ambientLoc, 0.5);
        gl.uniform1i(this.enableLightingLoc, show ? 1 : 0);
```

This code applies the lighting choice as yes or no and basically applies default ambient value

```
/**
         * @Task2 : You should implement the lighting and implement this
function
         */
        gl.useProgram(this.prog);
        gl.uniform1f(this.ambientLoc, ambient);
```

This code updates the ambient value

```
if(showTex && enableLighting){
```

```
                // UPDATE THIS PART TO HANDLE LIGHTING
                vec3 norm = normalize(v_normal);
                vec3 lightDir = normalize(lightPos);
                float diffuse = max(dot(norm, lightDir), 0.0);
                vec4 finalColor = texture2D(tex, v_texCoord);
                gl_FragColor = finalColor * (diffuse + ambient);
            }
```

Lastly this part make the calculations create a diffuser (which handles the basic reflection based on the position of light) and apply the colors to the original color.