Yavuz Mehmet Oğuz Şimşek
27920

PROJECT-2 REPORT

TASK1:

For non power of 2 images, I set the texture parameters different than power of 2 sized images. Specifically, I have used ‘gl.linear’ for the filter, which controls how the texture is sampled when a pixel covers more than one texture element. In addition to that, I have set the texture wrapping mode to ‘gl.clamp_to_edge’ for both the S and T coordinates. This change was necessary because these images do not support wrapping modes that require textures to be repeated, like ‘gl.repeat’.

Screenshot of the implementation:

```
} 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);
}
```

This approach enabled to extend the functionality of the existing texture mapping system, allowing for greater variety of texture sizes used in GL applications.

TASK2:

In TASK2, my focus was to apply lighting into the scene by performing ambient and diffuse lighting effects. In order to achieve this, I have to edit various aspects of the program such as the constructor, setMesh, draw, enableLight, setAmbientLight and also the fragment shader.

Implementation on these are listed below with screenshots and explanations.

Constructor():

```
constructor() {
    this.prog = InitShaderProgram(meshVS, meshFS);
    this.mvpLoc = gl.getUniformLocation(this.prog, 'mvp');
    this.showTexLoc = gl.getUniformLocation(this.prog, 'showTex');

    this.colorLoc = gl.getUniformLocation(this.prog, 'color');

    this.vertPosLoc = gl.getAttribLocation(this.prog, 'pos');
    this.texCoordLoc = gl.getAttribLocation(this.prog, 'texCoord');
    this.normalLoc = gl.getAttribLocation(this.prog, 'normal');


    this.vertbuffer = gl.createBuffer();
    this.texbuffer = gl.createBuffer();
    this.normalbuffer = gl.createBuffer();

    this.numTriangles = 0;

    this.ambientLightColorLoc = gl.getUniformLocation(this.prog, 'ambient');
    this.isAmbientSetLoc = gl.getUniformLocation(this.prog, 'isAmbientSet');
    this.lightPosLoc = gl.getUniformLocation(this.prog, 'lightPos'); // Light position

    this.enableLightingLoc = gl.getUniformLocation(this.prog, 'enableLighting');


}
```

Yavuz Mehmet Oğuz Şimşek
27920

- Added variables: normalLoc, normalbuffer, ambientLightColorLoc, isAmbientSetLoc, lightPosLoc
- The first two variables are for determining normals of the object, the last three for calculating lighting

setMesh():

```
setMesh(vertPos, texCoords, normalCoords) {
    gl.bindBuffer(gl.ARRAY_BUFFER, this.vertbuffer);
    gl.bufferData(gl.ARRAY_BUFFER, new Float32Array(vertPos), gl.STATIC_DRAW);

    // update texture coordinates
    gl.bindBuffer(gl.ARRAY_BUFFER, this.texbuffer);
    gl.bufferData(gl.ARRAY_BUFFER, new Float32Array(texCoords), gl.STATIC_DRAW);

    this.numTriangles = vertPos.length / 3;

    gl.bindBuffer(gl.ARRAY_BUFFER, this.normalbuffer);
    gl.bufferData(gl.ARRAY_BUFFER, new Float32Array(normalCoords), gl.STATIC_DRAW);
}
```

- Variables for normals are updated and bond to the buffers

draw():

```
draw(trans) {
    gl.useProgram(this.prog);

    gl.uniformMatrix4fv(this.mvpLoc, false, trans);

    gl.bindBuffer(gl.ARRAY_BUFFER, this.vertbuffer);
    gl.enableVertexAttribArray(this.vertPosLoc);
    gl.vertexAttribPointer(this.vertPosLoc, 3, gl.FLOAT, false, 0, 0);

    gl.bindBuffer(gl.ARRAY_BUFFER, this.texbuffer);
    gl.enableVertexAttribArray(this.texCoordLoc);
    gl.vertexAttribPointer(this.texCoordLoc, 2, gl.FLOAT, false, 0, 0);

    gl.bindBuffer(gl.ARRAY_BUFFER, this.normalbuffer);
    gl.enableVertexAttribArray(this.normalLoc);
    gl.vertexAttribPointer(this.normalLoc, 3, gl.FLOAT, false, 0, 0);


    updateLightPos();

    gl.uniform3fv(this.lightPosLoc, [lightX, lightY, 0.5]);
    gl.drawArrays(gl.TRIANGLES, 0, this.numTriangles);


}
```

- Normals and light position are rendered and updated

Yavuz Mehmet Oğuz Şimşek
27920

enableLight():

```
enableLighting(show) {
    gl.useProgram(this.prog);
    gl.uniform1i(this.enableLightingLoc, show ? 1 : 0);
}
```

- enableLight uniform are updated according to the whether the corresponding button is checked or not

setAmbientLight():

```
setAmbientLight(ambient) {
    gl.useProgram(this.prog);
    gl.uniform1f(this.ambientLightColorLoc, ambient);
    gl.uniform1f(this.isAmbientSetLoc, 1);
}
```

- Ambient parameter set according to slider value, whenever slider is updated the ambient value is also updated for later use in the fragment shader.
- Additional variable is also set to 1 in order to avoid uninitialized ambient value which corresponds to 0 leading to lighting to be zero when first enabled.

Fragment Shader:

```
const meshFS = `
        precision mediump float;

        uniform bool showTex;
        uniform bool enableLighting;
        uniform sampler2D tex;
        uniform vec3 color;
        uniform vec3 lightPos;
        uniform float ambient;
        uniform bool isAmbientSet;

        varying vec2 v_texCoord;
        varying vec3 v_normal;

        void main() {
            vec4 texColor = texture2D(tex, v_texCoord);

            float theAmbient = isAmbientSet ? ambient : 1.0;

            vec3 ambientColor = vec3(1.0, 1.0, 1.0) * theAmbient;
            vec3 diffuseColor = vec3(1.0, 1.0, 1.0) * theAmbient;


            if (enableLighting) {

                vec3 normal = normalize(v_normal);
                vec3 lightDir = normalize(lightPos);

                // Ambient component
                vec3 ambientVec = ambientColor;

                // Diffuse component
                float diff = max(dot(normal, -lightDir), 0.0);
                vec3 diffuseVec = diff * diffuseColor;

                gl_FragColor = texColor * vec4(ambientVec + diffuseVec, 1.0);

            } else if (showTex) {
                gl_FragColor = texture2D(tex, v_texCoord);
            } else {
                // No texture, no lighting, just the base color
                gl_FragColor = vec4(1.0, 0, 0, 1.0);
            }
        }`;
```

Yavuz Mehmet Oğuz Şimşek
27920

- The ambient variable is checked whether it is initialized or not. If not, the parameter assigned as 1 which is default, otherwise the ambient variable pass as it is
- Ambient and diffuse colors value are calculated according to ambient value.
- Diffuse light calculated
- Total shading calculated and assigned