BBM414 Computer Graphics Lab.
Assignment 4 - Report

Berk Bubuş
21945939
Department of Computer Engineering
Hacettepe University
Ankara, Turkey
b21945939@cs.hacettepe.edu.tr

Overview

In this assignment, we have 2 parts. At the first part, we have learned perspective and how to use pointer lock API.

1 Part 1

In Part 1, I have changed quad function and added some parts to init() where is in perspective2.js. quad is changed for true colors according to the assignment pdf. mousemove and keydown events are added for adding pointer lock and changing theta and phi variables according to the mouse movement.

```
function quad(a, b, c, d) {
     pointsArray.push(vertices[a]);
     colorsArray.push(vertexColors[a]);
     pointsArray.push(vertices[b]);
     colorsArray.push(vertexColors[a]);
     pointsArray.push(vertices[c]);
     colorsArray.push(vertexColors[a]);
     pointsArray.push(vertices[a]);
     colorsArray.push(vertexColors[b]);
     pointsArray.push(vertices[c]);
     colorsArray.push(vertexColors[b]);
     pointsArray.push(vertices[d]);
     colorsArray.push(vertexColors[b]);

}
```

```
document.addEventListener("mousemove", function (event) {
        if (lock) {
            console.log(event.movementX, event.movementY);
            const sensitivity = 0.01;
            theta -= event.movementX * sensitivity;
            phi -= event.movementY * sensitivity;
        }
    });
    document.addEventListener("keydown", function (event) {
        if (!lock && event.key === "p"){
            lock = true;
```

```
            canvas.requestPointerLock({
                unadjustedMovement: true,
              });
            console.log("Pointer lock engaged");
        }
        else if (lock && event.key === "p"){
            lock = false;
            document.exitPointerLock();
            console.log("Pointer lock disengaged");
        }
      });
```

2 Part 2 - HTML

I added a little part to HTML to add nocursor class to the style.

```
<html>
    <head>
        <title>Assignment 4 - Berk Bubu </title>
        <meta charset="UTF-8">
        <meta name="viewport" content="width=device-width, initial-scale=1.0">
        <script src="js/initialize.js" defer></script>
        <script src="js/shaders.js" defer></script>
        <script src="js/app.js" defer></script>
        <style>
            .nocursor {
                position: relative;
                overflow: hidden;
                cursor: none;
            }
        </style>
    </head>
    <body>
        <canvas class="nocursor" id="glCanvas" width="512" height="512">
            Oops ... your browser doesn't support the HTML5 canvas element
            </canvas>
    </div>
    </body>
</html>
```

3 Part 2 - app.js

At the first part of app.js file, I have got lots of variables that I need to calculate all matrices.

```
var gl;
const numOfComponents = 3;
const stride = 0;
const offset = 0;
var level = 10;
var catProgram;
var terrainProgram;
var cat;

var terrain;

var catLoaded = false;
var terrainLoaded = false;

var positionAttributeCat;
var projectionCat;
var viewCat;
```

```
var modelCat;

var positionAttributeTerrain;
var projectionTerrain;
var viewTerrain;
var modelTerrain;

let cameraPosition = [5.8, 5.1, 3.3, 1.0];
let cameraRotation = [-29, 58, 0.0, 1.0];
let colorCat = [0.2, 0.2, 0.2, 1.0];
let colorTerrain = [0.7, 0.95, 0.45, 1.0];

var colorCatLocation;
var colorTerrainLocation;

var angleX;
var angleY;
var fov;
var f;
var near;
var far;
var aspect;
var topValue;
var bottom;
var right;
var left;
var cosX;
var sinX;
var cosY;
var sinY;
var xaxis;
var yaxis;
var zaxis;
var view;
var projection;
var model;
```

I have a dot function to calculate dot products of vectors.

```
function dot(a, b){
    return a[0] * b[0] + a[1] * b[1] + a[2] * b[2];
}
```

In init function, I have mousemove and keydown handlers and loadOBJ calls to take models inside my code. And there is an important part, I'm using click and escape handlers to lock my pointer and control the camera more efficiently. You may click mouse1 and lock the pointer and after that when you close the project, you may press escape button and exit the pointer lock.

```
window.onload = function init() {
    document.addEventListener("keydown", function(event){
        if(event.key == "ArrowUp"){
            cameraPosition[2] += 0.1;
        }
        if(event.key == "ArrowDown"){
            cameraPosition[2] -= 0.1;
        }
        if(event.key == "ArrowLeft"){
            cameraPosition[0] -= 0.1;
        }
        if(event.key == "ArrowRight"){
            cameraPosition[0] += 0.1;
        }
```

```
        if(event.key == "PageUp"){
            cameraPosition[1] += 0.1;
        }
        if(event.key == "PageDown"){
            cameraPosition[1] -= 0.1;
        }
    });

    document.addEventListener("mousemove", function(event){
        cameraRotation[1] -=event.movementX;
        cameraRotation[0] -=event.movementY;

    });
    document.addEventListener("click", function () {
        document.body.requestPointerLock({
            unadjustedMovement: true,

        });
    });
    document.addEventListener("escape", function(){
        document.exitPointerLock();
    });
    loadOBJ("objects/cat.obj").then(function(obj){
        cat = obj;
        catLoaded = true;
        if(terrainLoaded){
            const canvas = document.querySelector("#glCanvas");
            gl = canvas.getContext("webgl2");
            if(!gl){
                alert("Unable to initialize WebGL2. Your browser or machine may not suppor
                return;
            }
            calculate();
        }
    });

    loadOBJ("objects/terrain.obj").then(function(obj){
        terrain = obj;
        terrainLoaded = true;
        if(catLoaded){
            const canvas = document.querySelector("#glCanvas");
            gl = canvas.getContext("webgl2");
            if(!gl){
                alert("Unable to initialize WebGL2. Your browser or machine may not suppor
                return;
            }
            calculate();
        }
    });
}
```

On calculate function, my code calculates all variable values and calls draw function.

```
function calculate(){
        angleX = cameraRotation[0] * Math.PI / 180.0;
        angleY = cameraRotation[1] * Math.PI / 180.0;
        fov = 70.0;
        f = 1.0 / Math.tan(fov * 0.5);
        near = 0.1;
        far = 100.0;
        aspect = 1.0;
        topValue = near * Math.tan(Math.PI/180.0 * fov * 0.5);
        bottom = -topValue;
        right = topValue * aspect;
        left = -right;
```

```
        cosX = Math.cos(angleX);
        sinX = Math.sin(angleX);
        cosY = Math.cos(angleY);
        sinY = Math.sin(angleY);
        xaxis = [cosY, 0.0, -sinY];
        yaxis = [sinY * sinX, cosX, cosY * sinX];
        zaxis = [sinY * cosX, -sinX, cosY * cosX];
        view = [
            xaxis[0], yaxis[0], zaxis[0], 0.0,
            xaxis[1], yaxis[1], zaxis[1], 0.0,
            xaxis[2], yaxis[2], zaxis[2], 0.0,
            -dot(xaxis, cameraPosition),-dot(yaxis, cameraPosition), -dot(zaxis, cameraPos
        ];
        projection = [
            2.0*near/(right-left), 0.0, 0.0, 0.0,
            0.0, 2.0*near/(topValue-bottom), 0.0, 0.0,
            (right+left)/(right-left), (topValue+bottom)/(topValue-bottom), -(far+near)/(
            0.0, 0.0, -2.0*far*near/(far-near),0.0
        ];
        model = [
            1.0, 0.0, 0.0, 0.0,
            0.0, 1.0, 0.0, 0.0,
            0.0, 0.0, 1.0, 0.0,
            0.0, 0.0, 0.0, 1.0
        ];

        gl.enable(gl.DEPTH_TEST);

        catProgram = initShaderProgram(gl, vsSource, fsSource);
        positionAttributeCat = gl.getAttribLocation(catProgram, 'aPosition');
        projectionCat = gl.getUniformLocation(catProgram, 'projection');
        viewCat = gl.getUniformLocation(catProgram, 'view');
        modelCat = gl.getUniformLocation(catProgram, 'model');


        colorCatLocation = gl.getUniformLocation(catProgram, 'color');


        terrainProgram = initShaderProgram(gl, vsSource, fsSource);
        positionAttributeTerrain = gl.getAttribLocation(terrainProgram, 'aPosition');
        projectionTerrain = gl.getUniformLocation(terrainProgram, 'projection');
        viewTerrain = gl.getUniformLocation(terrainProgram, 'view');
        modelTerrain = gl.getUniformLocation(terrainProgram, 'model');
        colorTerrainLocation = gl.getUniformLocation(terrainProgram, 'color');

        draw();
}
```

My draw function calculates projection, view and model matrices and send them to the vertex shader. I calculated all matrices here for better performance.

```
function draw(){
    angleX = cameraRotation[0] * Math.PI / 180.0;
    angleY = cameraRotation[1] * Math.PI / 180.0;
    cosX = Math.cos(angleX);
    sinX = Math.sin(angleX);
    cosY = Math.cos(angleY);
    sinY = Math.sin(angleY);
    xaxis = [cosY, 0.0, -sinY];
    yaxis = [sinY * sinX, cosX, cosY * sinX];
    zaxis = [sinY * cosX, -sinX, cosY * cosX];
    view = [
        xaxis[0], yaxis[0], zaxis[0], 0.0,
        xaxis[1], yaxis[1], zaxis[1], 0.0,
```

```
        xaxis[2], yaxis[2], zaxis[2], 0.0,
        -dot(xaxis, cameraPosition),-dot(yaxis, cameraPosition), -dot(zaxis, cameraPositi
    ];
    projection = [
        2.0*near/(right-left), 0.0, 0.0, 0.0,
        0.0, 2.0*near/(topValue-bottom), 0.0, 0.0,
        (right+left)/(right-left), (topValue+bottom)/(topValue-bottom), -(far+near)/(far-n
        0.0, 0.0, -2.0*far*near/(far-near),0.0
    ];
    gl.clear(gl.COLOR_BUFFER_BIT);
    //draw my cat and terrain objects
    const positionBuffer = initBuffer(gl, cat.position);
    //position, texcoord, normal
    gl.useProgram(catProgram);

    const vao = gl.createVertexArray();
    gl.bindVertexArray(vao);

    gl.bindBuffer(gl.ARRAY_BUFFER, positionBuffer);
    gl.enableVertexAttribArray(positionAttributeCat);
    gl.vertexAttribPointer(positionAttributeCat, 3, gl.FLOAT, false, 0, 0);

    gl.uniformMatrix4fv( projectionCat,false, projection );
    gl.uniformMatrix4fv( viewCat,false, view );
    gl.uniformMatrix4fv( modelCat,false, model );
    gl.uniform4fv( colorCatLocation, colorCat );
    // Draw the object
    gl.bindVertexArray(vao);
    gl.drawArrays(gl.TRIANGLES, 0, cat.position.length/numOfComponents);

    //draw terrain

    const positionBufferTerrain = initBuffer(gl, terrain.position);

    gl.useProgram(terrainProgram);

    const vaoTerrain = gl.createVertexArray();
    gl.bindVertexArray(vaoTerrain);

    gl.bindBuffer(gl.ARRAY_BUFFER, positionBufferTerrain);
    gl.enableVertexAttribArray(positionAttributeTerrain);
    gl.vertexAttribPointer(positionAttributeTerrain, 3, gl.FLOAT, false, 0, 0);

    gl.uniformMatrix4fv( projectionTerrain,false, projection );
    gl.uniformMatrix4fv( viewTerrain,false, view );
    gl.uniformMatrix4fv( modelTerrain,false, model );
    gl.uniform4fv( colorTerrainLocation, colorTerrain );

    gl.bindVertexArray(vaoTerrain);
    gl.drawArrays(gl.TRIANGLES, 0, terrain.position.length/numOfComponents);

    requestAnimationFrame(draw);
}
```

4 Part 2 - initialize.js

In this file, I added some functions that is taken from webgl2fundamentals website. They are about fetching obj files and using it in the code. It gives positions, texcoords, and normals but I just used positions for this assignment.

```
async function loadOBJ(url) {
    const response = await fetch(url);
    const text = await response.text();
```

```
    return parseOBJ(text);
}

function parseOBJ(text) {
    // because indices are base 1 let's just fill in the 0th data
    const objPositions = [[0, 0, 0]];
    const objTexcoords = [[0, 0]];
    const objNormals = [[0, 0, 0]];

    // same order as `f` indices
    const objVertexData = [
      objPositions,
      objTexcoords,
      objNormals,
    ];

    // same order as `f` indices
    let webglVertexData = [
      [],   // positions
      [],   // texcoords
      [],   // normals
    ];

    function newGeometry() {
      // If there is an existing geometry and it's
      // not empty then start a new one.
      if (geometry && geometry.data.position.length) {
        geometry = undefined;
      }
      setGeometry();
    }

    function addVertex(vert) {
      const ptn = vert.split('/');
      ptn.forEach((objIndexStr, i) => {
        if (!objIndexStr) {
          return;
        }
        const objIndex = parseInt(objIndexStr);
        const index = objIndex + (objIndex >= 0 ? 0 : objVertexData[i].length);
        webglVertexData[i].push(...objVertexData[i][index]);
      });
    }

    const keywords = {
      v(parts) {
        objPositions.push(parts.map(parseFloat));
      },
      vn(parts) {
        objNormals.push(parts.map(parseFloat));
      },
      vt(parts) {
        // should check for missing v and extra w?
        objTexcoords.push(parts.map(parseFloat));
      },
      f(parts) {
        const numTriangles = parts.length - 2;
        for (let tri = 0; tri < numTriangles; ++tri) {
          addVertex(parts[0]);
          addVertex(parts[tri + 1]);
          addVertex(parts[tri + 2]);
        }
      },
    };
```

```
    const keywordRE = /(\w*)(?: )*(.*)/;
    const lines = text.split('\n');
    for (let lineNo = 0; lineNo < lines.length; ++lineNo) {
      const line = lines[lineNo].trim();
      if (line === '' || line.startsWith('#')) {
        continue;
      }
      const m = keywordRE.exec(line);
      if (!m) {
        continue;
      }
      const [, keyword, unparsedArgs] = m;
      const parts = line.split(/\s+/).slice(1);
      const handler = keywords[keyword];
      if (!handler) {
        console.warn('unhandled keyword:', keyword);
// eslint-disable-line no-console
        continue;
      }
      handler(parts, unparsedArgs);
    }

    return {
      position: webglVertexData[0],
      texcoord: webglVertexData[1],
      normal: webglVertexData[2],
    };
  }
```

5 Part 2 - shaders.js

This file is simple as I calculated all the metrices on app.js part.

```
const vsSource = `#version 300 es
    #define PI 3.1415926538

    in vec4 aPosition;
    uniform mat4 projection;
    uniform mat4 view;
    uniform mat4 model;

    void main()
    {

        gl_Position = projection * view * model * aPosition;

    }
`;

//vec4(0.5, 0.8, 0.9,1.0);
const fsSource =  `#version 300 es
    precision mediump float;
    uniform vec4 color;
    out vec4 fColor;
    void main(){
        fColor = color;
    }
`;
```