

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»

ЗВІТ

ПРО ЛАБОРАТОРНУ РОБОТУ №5

ТЕМА: «ОСВІТЛЕННЯ ТА ЗАТІНЕННЯ»

Виконав:
Студент групи ІП-15
Мешков А.І.

Перевірив:
доц. каф. ІПІ
Родіонов П.Ю.

Київ 2023

ХІД РОБОТИ

1. Створити програму WebGL та використати фоновий колір.
2. Створити та розмістити у графічній сцені об'єкт довільної форми у перспективній проекції.
3. Реалізувати освітлення графічної сцени за допомогою моделі віддзеркалення Фонга
4. Внести зміни в освітлення сцени.
5. Скласти звіт про виконану роботу.

Лістинг 1 - Програмний код у файлі HTML

```
<!DOCTYPE <!DOCTYPE html>
<html>
<head>
  <meta charset="utf-8" />
  <meta http-equiv="X-UA-Compatible" content="IE=edge">
  <title>Лабораторна робота №5</title>
  <meta name="viewport" content="width=device-width, initial-scale=1">

  <style>
    body {
      margin: 0;
      background: #000;
    }

    canvas {
      width: 100%;
      height: 100%;
    }
  </style>
  <script src="script.js" defer></script>
  <script src="https://cdnjs.cloudflare.com/ajax/libs/gl-
matrix/2.4.0/gl-matrix.js"></script>
</head>
<body>

  <canvas id="myCanvas"></canvas>

</body>
</html>
```

Лістинг 2 - Програмний код у файлі JavaScript

```
const canvas = document.getElementById('myCanvas');
const gl = canvas.getContext('webgl');

if (!gl) {
  throw new Error('WebGL not supported');
}

const vertexData = [
  0.5, 0.5, 0.5,
  0.5, -0.5, 0.5,
  -0.5, 0.5, 0.5,
```

```

-0.5, 0.5, 0.5,
0.5, -0.5, 0.5,
-0.5, -0.5, 0.5,

-0.5, 0.5, 0.5,
-0.5, -0.5, 0.5,
-0.5, 0.5, -0.5,
-0.5, 0.5, -0.5,
-0.5, -0.5, 0.5,
-0.5, -0.5, -0.5,

-0.5, 0.5, -0.5,
-0.5, -0.5, -0.5,
0.5, 0.5, -0.5,
0.5, 0.5, -0.5,
-0.5, -0.5, -0.5,
0.5, -0.5, -0.5,

0.5, 0.5, -0.5,
0.5, -0.5, -0.5,
0.5, 0.5, 0.5,
0.5, 0.5, 0.5,
0.5, -0.5, -0.5,
0.5, -0.5, 0.5,

0.5, 0.5, 0.5,
0.5, 0.5, -0.5,
-0.5, 0.5, 0.5,
-0.5, 0.5, 0.5,
0.5, 0.5, -0.5,
-0.5, 0.5, -0.5,

0.5, -0.5, 0.5,
0.5, -0.5, -0.5,
-0.5, -0.5, 0.5,
-0.5, -0.5, 0.5,
0.5, -0.5, -0.5,
-0.5, -0.5, -0.5,
];

function repeat(n, pattern) {
    return [...Array(n)].reduce(sum => sum.concat(pattern), []);
}

const uvData = repeat(6, [
    1, 1,
    1, 0,
    0, 1,
    0, 1,
    1, 0,
    0, 0,
]);

const normalData = [
    ...repeat(6, [0, 0, 1]),
    ...repeat(6, [-1, 0, 0]),
    ...repeat(6, [0, 0, -1]),
    ...repeat(6, [1, 0, 0]),
    ...repeat(6, [0, 1, 0]),
    ...repeat(6, [0, -1, 0]),
];

const positionBuffer = gl.createBuffer();
gl.bindBuffer(gl.ARRAY_BUFFER, positionBuffer);

```

```

gl.bufferData(gl.ARRAY_BUFFER, new Float32Array(vertexData),
gl.STATIC_DRAW);

const uvBuffer = gl.createBuffer();
gl.bindBuffer(gl.ARRAY_BUFFER, uvBuffer);
gl.bufferData(gl.ARRAY_BUFFER, new Float32Array(uvData), gl.STATIC_DRAW);

const normalBuffer = gl.createBuffer();
gl.bindBuffer(gl.ARRAY_BUFFER, normalBuffer);
gl.bufferData(gl.ARRAY_BUFFER, new Float32Array(normalData),
gl.STATIC_DRAW);

function shaderProgram() {
    const vertexShader = gl.createShader(gl.VERTEX_SHADER);
    gl.shaderSource(vertexShader, `
        precision mediump float;

        const vec3 lightDirection = normalize(vec3(0,1,1));
        const float ambient = 0.1;

        attribute vec3 position;
        attribute vec2 uv;
        attribute vec3 normal;

        varying vec2 vUV;
        varying float vBrightness;

        uniform mat4 matrix;
        uniform mat4 normalMatrix;

        void main() {
            vec3 worldNormal = (normalMatrix * vec4(normal, 1)).xyz;
            float diffuse = max(0.0, dot(worldNormal, lightDirection));

            vec3 reflectDir = reflect(-lightDirection, worldNormal);

            float specular = pow(max(0.0, dot(reflectDir, normalize(-
vec3(0,1,1))))), 32.0);

            vUV = uv;

            vBrightness = ambient + diffuse + specular;

            gl_Position = matrix * vec4(position, 1);
        }
    `);
    gl.compileShader(vertexShader);

    const fragmentShader = gl.createShader(gl.FRAGMENT_SHADER);
    gl.shaderSource(fragmentShader, `
        precision mediump float;

        varying vec2 vUV;
        varying float vBrightness;

        void main() {
            gl_FragColor = vec4(0, 0.8, 0, 1.0) * vBrightness;
        }
    `);
    gl.compileShader(fragmentShader);

    const program = gl.createProgram();
    gl.attachShader(program, vertexShader);
    gl.attachShader(program, fragmentShader);

```

```

gl.linkProgram(program);

const positionLocation = gl.getAttribLocation(program, 'position');
gl.enableVertexAttribArray(positionLocation);
gl.bindBuffer(gl.ARRAY_BUFFER, positionBuffer);
gl.vertexAttribPointer(positionLocation, 3, gl.FLOAT, false, 0, 0);

const uvLocation = gl.getAttribLocation(program, 'uv');
gl.enableVertexAttribArray(uvLocation);
gl.bindBuffer(gl.ARRAY_BUFFER, uvBuffer);
gl.vertexAttribPointer(uvLocation, 2, gl.FLOAT, false, 0, 0);

const normalLocation = gl.getAttribLocation(program, 'normal');
gl.enableVertexAttribArray(normalLocation);
gl.bindBuffer(gl.ARRAY_BUFFER, normalBuffer);
gl.vertexAttribPointer(normalLocation, 3, gl.FLOAT, false, 0, 0);

gl.useProgram(program);
gl.enable(gl.DEPTH_TEST);

return {
  program,
  uniformLocations: {
    matrix: gl.getUniformLocation(program, 'matrix'),
    normalMatrix: gl.getUniformLocation(program, 'normalMatrix'),
  },
};
}

const { program, uniformLocations } = shaderProgram();

const modelMatrix = mat4.create();
const viewMatrix = mat4.create();
const projectionMatrix = mat4.create();
mat4.perspective(projectionMatrix,
  75 * Math.PI / 180,
  canvas.width / canvas.height,
  1e-4,
  1e4
);

const mvMatrix = mat4.create();
const mvpMatrix = mat4.create();

mat4.translate(viewMatrix, viewMatrix, [0, 0.1, 2]);
mat4.invert(viewMatrix, viewMatrix);

const normalMatrix = mat4.create();

function render() {
  // requestAnimationFrame(render);

  mat4.rotateY(modelMatrix, modelMatrix, 0.6);
  mat4.rotateY(modelMatrix, modelMatrix, 0.5);

  mat4.multiply(mvMatrix, viewMatrix, modelMatrix);
  mat4.multiply(mvpMatrix, projectionMatrix, mvMatrix);

  mat4.invert(normalMatrix, mvMatrix);
  mat4.transpose(normalMatrix, normalMatrix);

  gl.uniformMatrix4fv(uniformLocations.normalMatrix, false,
normalMatrix);
  gl.uniformMatrix4fv(uniformLocations.matrix, false, mvpMatrix);

```

```
gl.clear(gl.COLOR_BUFFER_BIT | gl.DEPTH_BUFFER_BIT);  
gl.drawArrays(gl.TRIANGLES, 0, vertexData.length / 3);  
}  
render();
```



Рисунок 1 – Результат виконання завдання

ВИСНОВОК

В ході виконання лабораторної роботи №5 "Освітлення та затінення" було досягнуто наступного:

1. Створено програму WebGL та використано фоновий колір.
 - ☐ Визначено основні елементи WebGL, такі як канвас та контекст.
 - ☐ Використано фоновий колір.
2. Створено та розміщено у графічній сцені об'єкт довільної форми у перспективній проекції.
 - ☐ Визначено вершини, текстурні координати та нормалі для графічного об'єкта.
 - ☐ Реалізовано відображення об'єкта у тривимірному просторі з використанням перспективної проекції.
3. Реалізовано освітлення графічної сцени за допомогою моделі віддзеркалення Фонга.
 - ☐ Використано модель освітлення Фонга, яка враховує дифузне та віддзеркалене світло.
 - ☐ Додано розрахунок віддзеркаленого світла (specular) за допомогою вектора віддзеркалення та визначення кута падіння світла на поверхню.
4. Внесено зміни в освітлення сцени.
 - ☐ Здійснено зміни в параметрах освітлення, такі як напрямок світла та ступінь віддзеркалення, для досягнення бажаного візуального ефекту.

Виконана робота дозволила отримати практичні навички роботи з освітленням графічної сцени у середовищі WebGL та впровадити модель віддзеркалення Фонга для реалістичного освітлення об'єктів.