SPRAWOZDANIE

Zajęcia: Grafika komputerowa
Prowadzący: prof. dr hab. Vasyl Martsenyuk

Laboratorium II
Data
Temat: Zadanie_OpenGL2 Zadanie_Swiatlo

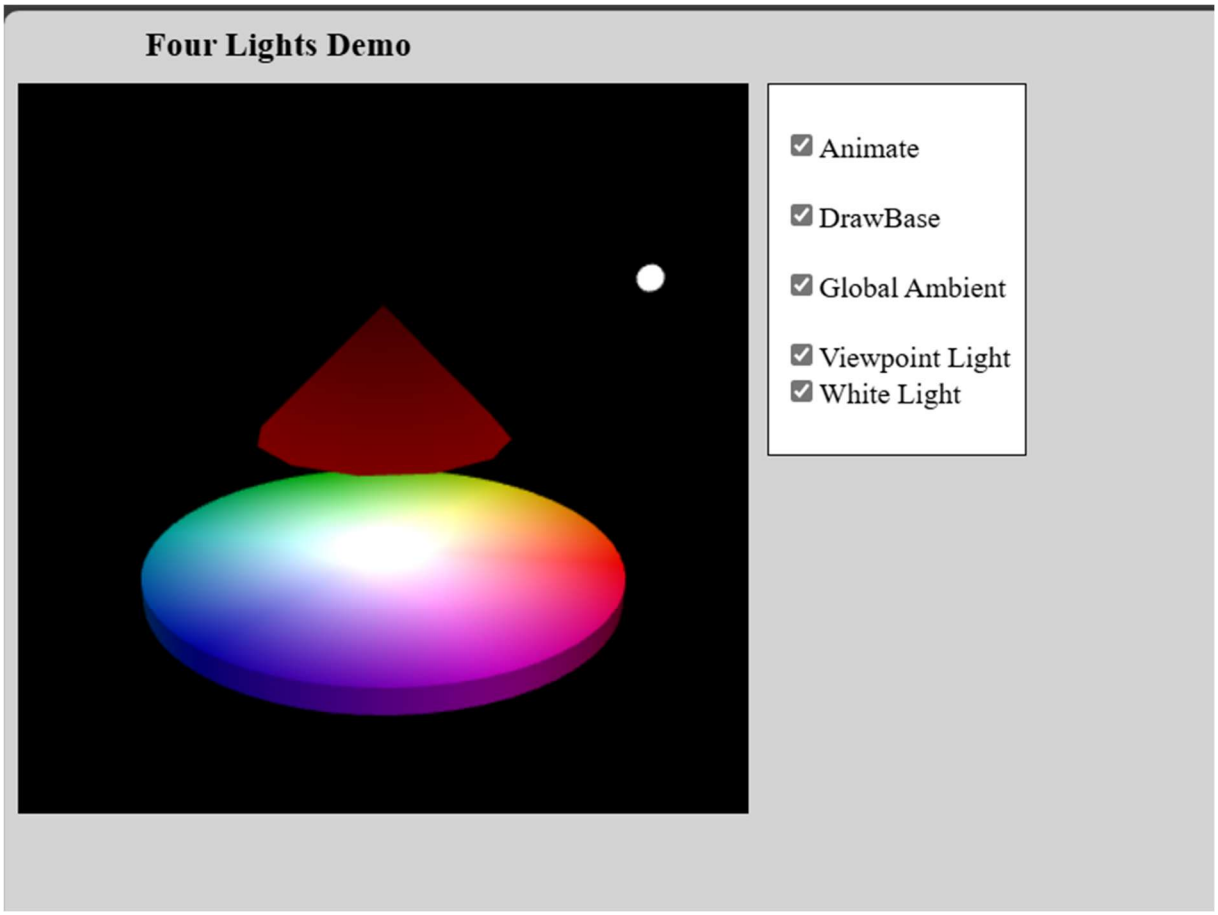
Wariant 8 + 4

Jakub Bąk
Informatyka I stopień,
stacjonarne,
4 semestr,
Gr.3b

1. Polecenie

Celem jest stworzenie pyramidy z użyciem różnych materiałów okrelonych wariantem zadania i umieszczenie jej na „podstawie”. Użytkownik może obracać podstawę wokół osi Y, przeciągając mysz w poziomie. Scena wykorzystuje globalne światło otoczenia (ambient) oraz źródło światła o kształcie kuli z możliwością animacji obrotu wokół pyramidy.
Aby wykonać laboratorium w JavaScript polecane jest zapoznanie z plikami .html: four-lights-demo.html oraz materials-demo.html

2. Wyniki zadania:



Na podstawie podanych plików należało wykonać zadanie z polecenia

3. Wykorzystane komendy:

Link do github: https://github.com/Szeladin/grafika.git

Kod Programu:

```
1. <!DOCTYPE html>
2. <html>
```

```
<head>
<meta charset="UTF-8">
<title>Four Lights</title>
<link rel="stylesheet" href="../demo.css">
<script src="../script/demo-core.js"></script>
<script src="../script/glsim.js"></script>
<script src="../script/pyramid-model-IFS.js"></script>
<script>

var camera;

var animate;
var drawBase;
var ambientLight;
var viewpointLight, whiteLight;

var animating = false;
var frameNumber = 0;

const redPlasticMaterial = {
    ambient: [0.0, 0.0, 0.0, 1.0],
    diffuse: [0.5, 0.0, 0.0, 1.0],
    specular: [0.7, 0.6, 0.6, 1.0],
    shininess: 0.25 * 128,
};

function applyMaterial(material) {
    glMaterialfv(GL_FRONT_AND_BACK, GL_AMBIENT, material.ambient);
    glMaterialfv(GL_FRONT_AND_BACK, GL_DIFFUSE, material.diffuse);
    glMaterialfv(GL_FRONT_AND_BACK, GL_SPECULAR, material.specular);
    glMateriali(GL_FRONT_AND_BACK, GL_SHININESS, material.shininess);
}

function uvSphere(radius, slices, stacks) {
    var i, j;
    for (j = 0; j < stacks; j++) {
        var latitude1 = (Math.PI / stacks) * j - Math.PI / 2;
        var latitude2 = (Math.PI / stacks) * (j + 1) - Math.PI / 2;
        var sinLat1 = Math.sin(latitude1);
        var cosLat1 = Math.cos(latitude1);
        var sinLat2 = Math.sin(latitude2);
        var cosLat2 = Math.cos(latitude2);
        glBegin(GL_TRIANGLE_STRIP);
        for (i = 0; i <= slices; i++) {
            var longitude = (2 * Math.PI / slices) * i;
            var sinLong = Math.sin(longitude);
            var cosLong = Math.cos(longitude);
            var x1 = cosLong * cosLat1;
            var y1 = sinLong * cosLat1;
            var z1 = sinLat1;
            var x2 = cosLong * cosLat2;
            var y2 = sinLong * cosLat2;
            var z2 = sinLat2;
            glNormal3d(x2, y2, z2);
            glVertex3d(radius * x2, radius * y2, radius * z2);
            glNormal3d(x1, y1, z1);
            glVertex3d(radius * x1, radius * y1, radius * z1);
        }
        glEnd();
    }
}

function lights() {
    glColor3d(0.5, 0.5, 0.5);
    var zero = [0, 0, 0, 1];
    glMaterialfv(GL_FRONT_AND_BACK, GL_SPECULAR, zero);

    if (viewpointLight.checked)
        glEnable(GL_LIGHT0);
    else
```

```
        glDisable(GL_LIGHT0);

    if (whiteLight.checked) {
        glMaterialfv(GL_FRONT_AND_BACK, GL_EMISSION, [1, 1, 1, 1]);
        glEnable(GL_LIGHT1);
    } else {
        glMaterialfv(GL_FRONT_AND_BACK, GL_EMISSION, zero);
        glDisable(GL_LIGHT1);
    }
    glPushMatrix();
    glRotated(-frameNumber, 0, 1, 0);
    glTranslated(10, 7, 0, 1);
    glLightfv(GL_LIGHT1, GL_POSITION, zero);
    uvSphere(0.5, 16, 8);
    glPopMatrix();

    glMaterialfv(GL_FRONT_AND_BACK, GL_EMISSION, zero);
}

function colorArrayForHue(h, s, v) {
    var r, g, b;
    var c, x;
    h = h * 359;
    c = v * s;
    x = (h < 120) ? h / 60 : (h < 240) ? (h - 120) / 60 : (h - 240) / 60;
    x = c * (1 - Math.abs(x - 1));
    x += (v - c);
    switch (Math.floor(h / 60)) {
        case 0: r = v; g = x; b = v - c; break;
        case 1: r = x; g = v; b = v - c; break;
        case 2: r = v - c; g = v; b = x; break;
        case 3: r = v - c; g = x; b = v; break;
        case 4: r = x; g = v - c; b = v; break;
        case 5: r = v; g = v - c; b = x; break;
    }
    var array = new Array(4);
    array[0] = r;
    array[1] = g;
    array[2] = b;
    array[3] = 1;
    return array;
}

function drawCylinder() {
    var i;
    var rgba;
    glBegin(GL_TRIANGLE_STRIP);
    for (i = 0; i <= 64; i++) {
        var angle = 2 * Math.PI / 64 * i;
        var x = Math.cos(angle);
        var y = Math.sin(angle);
        rgba = colorArrayForHue(i / 64.0, 1, 0.6);
        glColor3dv(rgba);
        glNormal3d(x, y, 0);
        glVertex3d(x, y, 1);
        glVertex3d(x, y, -1);
    }
    glEnd();
    glNormal3d(0, 0, 1);
    glBegin(GL_TRIANGLE_FAN);
    glColor3d(1, 1, 1);
    glVertex3d(0, 0, 1);
    for (i = 0; i <= 64; i++) {
        var angle = 2 * Math.PI / 64 * i;
        var x = Math.cos(angle);
        var y = Math.sin(angle);
        rgba = colorArrayForHue(i / 64.0, 1, 0.6);
        glColor3dv(rgba);
        glVertex3d(x, y, 1);
    }
```

```
    glEnd();
    glNormal3f(0, 0, -1);
    glBegin(GL_TRIANGLE_FAN);
    glColor3d(1, 1, 1);
    glVertex3d(0, 0, -1);
    for (i = 64; i >= 0; i--) {
        var angle = 2 * Math.PI / 64 * i;
        var x = Math.cos(angle);
        var y = Math.sin(angle);
        rgba = colorArrayForHue(i / 64.0, 1, 0.6);
        glColor3dv(rgba);
        glVertex3d(x, y, -1);
    }
    glEnd();
}

function display() {
    glClearColor(0, 0, 0, 1);
    glClear(GL_COLOR_BUFFER_BIT | GL_DEPTH_BUFFER_BIT);

    camera.apply();

    lights();

    if (ambientLight.checked) {
        glLightModelfv(GL_LIGHT_MODEL_AMBIENT, [0.15, 0.15, 0.15, 1]);
    } else {
        glLightModelfv(GL_LIGHT_MODEL_AMBIENT, [0, 0, 0, 1]);
    }

    if (drawBase.checked) {
        glMaterialfv(GL_FRONT_AND_BACK, GL_SPECULAR, [0, 0, 0, 1]);

        glPushMatrix();
        glTranslated(0, -5, 0);
        glRotated(-90, 1, 0, 0);
        glScaled(10, 10, 0.5);
        drawCylinder();
        glPopMatrix();
    }

    glPushMatrix();
    glTranslatef(0, 0.68, 0);
    glScalef(0.65, 0.65, 0.65);
    glsimDrawModel(pyramidModel);
    glPopMatrix();
}

function initGL() {
    glClearColor(0, 0, 0, 1);
    glEnable(GL_DEPTH_TEST);
    glEnable(GL_LIGHTING);
    glEnable(GL_LIGHT0);
    glEnable(GL_NORMALIZE);
    glEnable(GL_COLOR_MATERIAL);
    glLightModeli(GL_LIGHT_MODEL_LOCAL_VIEWER, 1);
    glMateriali(GL_FRONT_AND_BACK, GL_SHININESS, 32);

    var dim = [0.5, 0.5, 0.5, 1];
    glLightfv(GL_LIGHT0, GL_DIFFUSE, dim);
    glLightfv(GL_LIGHT0, GL_SPECULAR, dim);

    var white = [1, 1, 1, 1];
    var whitea = [0.2, 0.2, 0.2, 1];
    glLightfv(GL_LIGHT1, GL_AMBIENT, whitea);
    glLightfv(GL_LIGHT1, GL_DIFFUSE, white);
    glLightfv(GL_LIGHT1, GL_SPECULAR, white);
}

function doFrame() {
```

```
    if (animating) {
        frameNumber++;
        display();
        setTimeout(doFrame, 30);
    }
}

function init() {
    try {
        glsimUse("maincanvas");
    } catch (e) {
        document.getElementById("canvas-holder").innerHTML =
            "<p><b>Sorry, an error occurred:<br>" + e + "</b></p>";
        return;
    }
    animate = document.getElementById("animate");
    drawBase = document.getElementById("drawBase");
    ambientLight = document.getElementById("ambientLight");
    viewpointLight = document.getElementById("viewpointLight");
    whiteLight = document.getElementById("whiteLight");
    animate.checked = false;
    drawBase.checked = true;
    ambientLight.checked = true;
    viewpointLight.checked = true;
    whiteLight.checked = true;
    drawBase.onchange = display;
    ambientLight.onchange = display;
    viewpointLight.onchange = display;
    whiteLight.onchange = display;
    animate.onchange = function () {
        if (animate.checked) {
            animating = true;
            doFrame();
        } else {
            animating = false;
        }
    };
    initGL();
    camera = new Camera();
    camera.lookAt(5, 10, 30, 0, 0, 0, 0, 1, 0);
    camera.setScale(15);
    camera.installTrackball(display);
    display();
}

</script>
</head>
<body onload="init()">

<div id="content">

<h3 id="headline">Four Lights Demo</h3>

<div id="canvas-holder">
<canvas id="maincanvas" width="400" height="400"></canvas>
</div>

<div id="tools">
<p>
<label><input type="checkbox" id="animate">Animate</label><br><br>
<label><input type="checkbox" id="drawBase">DrawBase</label><br><br>
<label><input type="checkbox" id="ambientLight">Global Ambient</label><br><br>
<label><input type="checkbox" id="viewpointLight">Viewpoint Light</label><br>
<label><input type="checkbox" id="whiteLight">White Light</label><br>
</p>
</div>
</div>
</body>
</html>

```

Program to interaktywna demonstracja graficzna w WebGL, która przedstawia scenę z różnymi źródłami światła. Użytkownik może włączać/wyłączać animację, rysowanie podstawy oraz różne typy oświetlenia (ambient, viewpoint, white light) za pomocą pól wyboru. Scena zawiera obiekty 3D, takie jak piramida i podstawę renderowane z wykorzystaniem materiałów i świateł. Kamera umożliwia manipulację widokiem za pomocą trackballa.

4. Wnioski:

OpenGL umożliwia realistyczne odwzorowanie oświetlenia w scenach 3D poprzez różne typy świateł, takie jak ambientowe, punktowe i emitujące. OpenGL w połączeniu z WebGL umożliwia tworzenie interaktywnych aplikacji graficznych dostępnych w przeglądarce.