**SPRAWOZDANIE**

Zajęcia: Grafika komputerowa

Prowadzący: mgr Mikołaj Grygiel

**Laboratorium 7**

12.05.2024

**Temat:** Swiatło i materiały w OpenGL

**Wariant**: 1

Radosław Skrzypczyński

Informatyka I stopień,

niestacjonarne,

4 semestr,

Gr. 1B

1. **Polecenie:**

**2. Wprowadzane dane:**

Menu z przyciskami do ustawiania parametrów sceny i obiektu

**3. Wykorzystane komendy:**

*pentagonPyramidModel.js*

*var pentagonPyramidModel = {*

*"vertexPositions" : new Float32Array([*

*// Podstawa pięciokąta (leżąca na płaszczyźnie XZ)*

*0.0, -9.0, -10.0, // Wierzchołek 0*

*9.51, -9.0, -3.09, // Wierzchołek 1*

*5.876, -9.0, 8.09, // Wierzchołek 2*

*-5.876, -9.0, 8.09, // Wierzchołek 3*

*-9.51, -9.0, -3.09, // Wierzchołek 4*

*0.0, 0.0, 0.0 // Wierzchołek szczytowy piramidy*

*]),*

*"vertexNormals" : new Float32Array([*

*// Normalne wektory dla wierzchołków podstawy*

*0.0, -1.0, 0.0, // Wierzchołek 0*

*0.0, -1.0, 0.0, // Wierzchołek 1*

*0.0, -1.0, 0.0, // Wierzchołek 2*

*0.0, -1.0, 0.0, // Wierzchołek 3*

*0.0, -1.0, 0.0, // Wierzchołek 4*

*// Normalne wektory dla wierzchołka szczytowego*

*0.0, 1.0, 0.0 // Wierzchołek 5*

*]),*

*"vertexTextureCoords" : new Float32Array([*

*// Współrzędne tekstury dla wierzchołków podstawy*

*0.5, 0.0, // Wierzchołek 0*

*1.0, 0.0, // Wierzchołek 1*

*1.0, 0.5, // Wierzchołek 2*

*0.5, 1.0, // Wierzchołek 3*

*0.0, 1.0, // Wierzchołek 4*

*// Współrzędne tekstury dla wierzchołka szczytowego*

*0.5, 0.5 // Wierzchołek 5*

*]),*

*"indices" : new Uint16Array([*

*// Ściany boczne piramidy*

*0, 1, 5, // Ściana 1*

*1, 2, 5, // Ściana 2*

*2, 3, 5, // Ściana 3*

*3, 4, 5, // Ściana 4*

*4, 0, 5, // Ściana 5*

*// Podstawa piramidy*

*0, 1, 2, // Trójkąt 1*

*2, 3, 4, // Trójkąt 2*

*4, 0, 1 // Trójkąt 3*

*])*

*};*

*zadanie.js*

*<!DOCTYPE html>*

*<html>*

*<!--*

*\* Shows a scene (a teapot on a short cylindrical base) that is illuminated*

*\* by up to four lights plus global ambient light. The user can turn the*

*\* lights on and off. The global ambient light is a dim white. There is*

*\* a white "viewpoint" light that points from the direction of the viewer*

*\* into the scene. There is a red light, a blue light, and a green light*

*\* that rotate in circles above the teapot. (The user can turn the animation*

*\* on and off.) The locations of the colored lights are marked by spheres,*

*\* which are gray when the light is off and are colored by some emission color*

*\* when the light is on. The teapot is gray with weak specular highlights.*

*\* The base is colored with a spectrum. (The user can turn the display of*

*\* the base on and off.) The mouse can be used to rotate the scene.*

*-->*

*<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/teapot-model-IFS.js"></script>*

*<script src="../script/pentagonPyramidModel.js"></script>*

*<script>*

*var camera;*

*var material;*

*var animate; // checkbox that controls animation*

*var drawBase; // checkbox for turning display of base on and off*

*var ambientLight; // checkbox for turning global ambient light on and off*

*var viewpointLight, redLight, blueLight, greenLight, whiteLight; // checkboxes for turning lights on and off*

*var animating = false; // tells whether the animation is running*

*var frameNumber = 0; // frame number for use in animation*

*var materialProperties = [*

*[ /\* 1, 12 - basic gray \*/ 0.6, 0.6, 0.6, 1.0, 0.6, 0.6, 0.6, 1.0, 1, 1, 1, 1, 16 ],*

*[ /\* 2, 13 - "jade" \*/ 0.135, 0.2225, 0.1575, 1.0, 0.54, 0.89, 0.63, 1.0, 0.316228, 0.316228, 0.316228, 1.0, 0.1\*128 ],*

*[ /\* 3, 14 - "brass" \*/ 0.329412, 0.223529, 0.027451, 1.0, 0.780392, 0.568627, 0.113725, 1.0, 0.992157, 0.941176, 0.807843, 1.0, 0.21794872\*128 ],*

*[ /\* 4, 15 - "bronze" \*/ 0.2125, 0.1275, 0.054, 1.0, 0.714, 0.4284, 0.18144, 1.0, 0.393548, 0.271906, 0.166721, 1.0, 0.2\*128 ],*

*[ /\* 5, 16 - "copper" \*/ 0.19125, 0.0735, 0.0225, 1.0, 0.7038, 0.27048, 0.0828, 1.0, 0.256777, 0.137622, 0.086014, 1.0, 0.1\*128 ],*

*[ /\* 6, 17 - "gold" \*/ 0.24725, 0.1995, 0.0745, 1.0, 0.75164, 0.60648, 0.22648, 1.0, 0.628281, 0.555802, 0.366065, 1.0, 0.4\*128 ],*

*[ /\* 7, 18 - "green plastic" \*/ 0.0, 0.0, 0.0, 1.0, 0.1, 0.35, 0.1, 1.0, 0.45, 0.55, 0.45, 1.0, .25\*128 ],*

*[ /\* 8, 19 - "red plastic" \*/ 0.0, 0.0, 0.0, 1.0, 0.5, 0.0, 0.0, 1.0, 0.7, 0.6, 0.6, 1.0, .25\*128 ],*

*[ /\* 9, 20 - "black plastic" \*/ 0, 0, 0, 1.0, 0.01, 0.01, 0.01, 1.0, 1, 1, 1, 1.0, .25\*128 ],*

*[ /\* 10, 21 - "black rubber" \*/ 0.02, 0.02, 0.02, 1.0, 0.01, 0.01, 0.01, 1.0, 0.4, 0.4, 0.4, 1.0, .078125\*128 ],*

*[ /\* 11, 22 - "cyan rubber" \*/ 0.0, 0.05, 0.05, 1.0, 0.4, 0.5, 0.5, 1.0, 0.04, 0.7, 0.7, 1.0, .078125\*128 ],*

*];*

*//------------------- Drawing functions ----------------------------------------*

*/\* Draws a sphere. \*/*

*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();*

*}*

*} // end uvSphere*

*/\* Sets the positions of the colored lights and turns them on and off, depending on*

*\* the state of the redLight, greenLight, and blueLight options. Draws a small*

*\* sphere at the location of each light.*

*\*/*

*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 (redLight.checked) {*

*glMaterialfv(GL\_FRONT\_AND\_BACK, GL\_EMISSION, [0.5, 0, 0, 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);*

*glLightfv(GL\_LIGHT1, GL\_POSITION, zero);*

*uvSphere(0.5, 16, 8);*

*glPopMatrix();*

*if (greenLight.checked) {*

*glMaterialfv(GL\_FRONT\_AND\_BACK, GL\_EMISSION, [0, 0.5, 0, 1]);*

*glEnable(GL\_LIGHT2);*

*}*

*else {*

*glMaterialfv(GL\_FRONT\_AND\_BACK, GL\_EMISSION, zero);*

*glDisable(GL\_LIGHT2);*

*}*

*glPushMatrix();*

*glRotated((frameNumber+100)\*0.8743, 0, 1, 0);*

*glTranslated(9, 8, 0);*

*glLightfv(GL\_LIGHT2, GL\_POSITION, zero);*

*uvSphere(0.5, 16, 8);*

*glPopMatrix();*

*if (blueLight.checked) {*

*glMaterialfv(GL\_FRONT\_AND\_BACK, GL\_EMISSION, [0, 0, 0.5, 1]);*

*glEnable(GL\_LIGHT3);*

*}*

*else {*

*glMaterialfv(GL\_FRONT\_AND\_BACK, GL\_EMISSION, zero);*

*glDisable(GL\_LIGHT3);*

*}*

*glPushMatrix();*

*glRotated((frameNumber-100)\*1.3057, 0, 1, 0);*

*glTranslated(9.5, 7.5, 0);*

*glLightfv(GL\_LIGHT3, GL\_POSITION, zero);*

*uvSphere(0.5, 16, 8);*

*glPopMatrix();*

*if (whiteLight.checked) {*

*glMaterialfv(GL\_FRONT\_AND\_BACK, GL\_EMISSION, [0.5, 0.5, 0.5, 1]);*

*glEnable(GL\_LIGHT4);*

*}*

*else {*

*glMaterialfv(GL\_FRONT\_AND\_BACK, GL\_EMISSION, zero);*

*glDisable(GL\_LIGHT4);*

*}*

*glPushMatrix();*

*glRotated((frameNumber-100)\*1.3057, 0, 1, 0);*

*glTranslated(12, 5, 0);*

*glLightfv(GL\_LIGHT4, GL\_POSITION, zero);*

*uvSphere(0.5, 16, 8);*

*glPopMatrix();*

*glMaterialfv(GL\_FRONT\_AND\_BACK, GL\_EMISSION, zero); // Turn off emission color!*

*} // end lights()*

*/\* Fills the array with RGBA color components corresponding the given HSV color*

*\* components, where h, s, and v are in the range 0.0 to 1.0.*

*\*/*

*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;*

*}*

*/\* Draws a cylinder with height 2 and radius 1, centered at the origin, with its axis*

*\* along the z-axis. A spectrum of hues is applied to the vertices along the edges*

*\* of the cylinder. (Since GL\_COLOR\_MATERIAL is enabled in this program, the colors*

*\* specified here are used as ambient and diffuse material colors for the cylinder.)*

*\*/*

*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 ); // Normal for both vertices at this angle.*

*glVertex3d( x, y, 1 ); // Vertex on the top edge.*

*glVertex3d( x, y, -1 ); // Vertex on the bottom edge.*

*}*

*glEnd();*

*glNormal3d( 0, 0, 1 );*

*glBegin(GL\_TRIANGLE\_FAN); // Draw the top, in the plane z = 1.*

*glColor3d(1,1,1); // ambient and diffuse for center*

*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); // Draw the bottom, in the plane z = -1*

*glColor3d(1,1,1); // ambient and diffuse for center*

*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();*

*}*

*/\* Draws the scene.*

*\*/*

*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();*

*}*

*glColor3d(0.7,0.7,0.7); // sets diffuse and ambient color for teapot*

*glMaterialfv(GL\_FRONT\_AND\_BACK, GL\_AMBIENT\_AND\_DIFFUSE, [material[4],material[5],material[6],1]);*

*glMaterialfv(GL\_FRONT\_AND\_BACK, GL\_SPECULAR, [material[8],material[9],material[10],1]);*

*glMateriali(GL\_FRONT\_AND\_BACK, GL\_SHININESS, material[12]);*

*glPushMatrix();*

*glTranslatef(0,0.68,0);*

*glScalef(0.65,0.65,0.65);*

*glsimDrawModel(pentagonPyramidModel);*

*glPopMatrix();*

*}*

*/\* Initialization, including setting up a camera and configuring the four lights.*

*\*/*

*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 red = [ 0.5, 0, 0, 1];*

*var reda = [ 0.1, 0, 0, 1];*

*glLightfv(GL\_LIGHT1, GL\_AMBIENT, reda);*

*glLightfv(GL\_LIGHT1, GL\_DIFFUSE, red);*

*glLightfv(GL\_LIGHT1, GL\_SPECULAR, red);*

*var gr = [ 0, 0.5, 0, 1 ];*

*var gra = [ 0, 0.1, 0, 1 ];*

*glLightfv(GL\_LIGHT2, GL\_AMBIENT, gra);*

*glLightfv(GL\_LIGHT2, GL\_DIFFUSE, gr);*

*glLightfv(GL\_LIGHT2, GL\_SPECULAR, gr);*

*var bl = [0, 0, 0.5, 1];*

*var bla = [0, 0, 0.1, 1];*

*glLightfv(GL\_LIGHT3, GL\_AMBIENT, bla);*

*glLightfv(GL\_LIGHT3, GL\_DIFFUSE, bl);*

*glLightfv(GL\_LIGHT3, GL\_SPECULAR, bl);*

*var wl = [0.7, 0.7, 0.7, 1];*

*var wla = [0.2, 0.2, 0.2, 1];*

*glLightfv(GL\_LIGHT4, GL\_AMBIENT, wla);*

*glLightfv(GL\_LIGHT4, GL\_DIFFUSE, wl);*

*glLightfv(GL\_LIGHT4, GL\_SPECULAR, wl);*

*}*

*/\* Animation function, called every 30 milliseconds if an animation is runnin \*/*

*function doFrame() {*

*if (animating) {*

*frameNumber++;*

*display();*

*setTimeout(doFrame, 30);*

*}*

*}*

*function newMaterial(){*

*var index = Number(document.getElementById("materialSelect").value);*

*material = materialProperties[index];*

*}*

*function init() {*

*try {*

*GLSim.lightCount = 5;*

*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");*

*redLight = document.getElementById("redLight");*

*blueLight = document.getElementById("blueLight");*

*greenLight = document.getElementById("greenLight");*

*whiteLight = document.getElementById("whiteLight");*

*animate.checked = false;*

*drawBase.checked = true;*

*ambientLight.checked = true;*

*viewpointLight.checked = true;*

*redLight.checked = true;*

*greenLight.checked = true;*

*blueLight.checked = true;*

*whiteLight.checked = true;*

*material = materialProperties[0];*

*document.getElementById("materialSelect").onchange = newMaterial;*

*drawBase.onchange = display;*

*ambientLight.onchange = display;*

*viewpointLight.onchange = display;*

*redLight.onchange = display;*

*blueLight.onchange = display;*

*greenLight.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>Material Presets:<br>*

*<select id="materialSelect">*

*<option value="0">Basic Gray</option>*

*<option value="1">Jade</option>*

*<option value="2">Brass</option>*

*<option value="3">Bronze</option>*

*<option value="4">Copper</option>*

*<option value="5">Gold</option>*

*<option value="6">Green Plastic</option>*

*<option value="7">Red Plastic</option>*

*<option value="8">Black Plastic</option>*

*<option value="9">Black Rubber</option>*

*<option value="10">Cyan Rubber</option>*

*</select>*

*<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="redLight">Red Light</label><br>*

*<label><input type="checkbox" id="greenLight">Green Light</label><br>*

*<label><input type="checkbox" id="blueLight">Blue Light</label><br>*

*<label><input type="checkbox" id="whiteLight">White Light</label><br>*

*</p>*

*</div>*

*</div>*

*<div id="help-content" style="display:none">*

*<h3>About this demo...</h3>*

*<p>The demo shows a scene that is illuminated by four light sources, as well*

*as by a dim global ambient light. There is a "viewpoint" light, which is a*

*white light shining into the scene from the direction of the viewer.*

*There are three colored positional lights, with a small sphere at the*

*position of each light. The checkboxes allow you to turn the*

*lights off and on. You can control whether the base is displayed.*

*You can turn on an animation that makes the colored lights fly*

*in circles around the teapot. And you can rotate the scene as a whole*

*using your mouse.</p>*

*<p>Some things to do: Turn off all of the lights except one, and note*

*the effect of that light on the teapot. (Maybe turn off the base so it*

*doesn't distract you.) For the colored lights, be sure to use animation*

*so that you can see how the colors on the teapot change. Try it*

*with only two colored lights turned on, and see how their colors combine.*

*Note that when one a colored light is turned on, it adds some of its*

*color to the ambient light; to see the effect look at the bottom*

*of the teapot, with just one light turned on (and display of the base turned off).</p>*

*<p>Note that when the only light source is a single colored light, part of the base is black;*

*this is because, for example, a part of a surface that contains no red will*

*not reflect any red light.</p>*

*<p>You might also note that the teapot does not cast shadoes on the base. OpenGL*

*does not support shadows, except with a lot of extra work.</p>*

*<p>The base has a material that changes colors around the edge of the cylinder.*

*This is meant as a demonstration of using different material colors at different*

*vertices of the same primitive. It also uses <i>GL\_COLOR\_MATERIAL</i> so that*

*the material colors can be set by calling <i>glColor3fv</i> instead of*

*<i>glMaterialfv</i>.</p>*

*</div>*

*<!-- support for help text -- do not change. -->*

*<div id="help-icon">*

*<img src="../image/question32.png" onclick="showDemoHelp()"*

*title="Click here for information about this demo." width="32" height="32">*

*</div>*

*<div id="hide-help-icon">*

*<img src="../image/close32.png" onclick="showDemoHelp()"*

*title="Click here to return to the demo." width="65" height="32">*

*</div>*

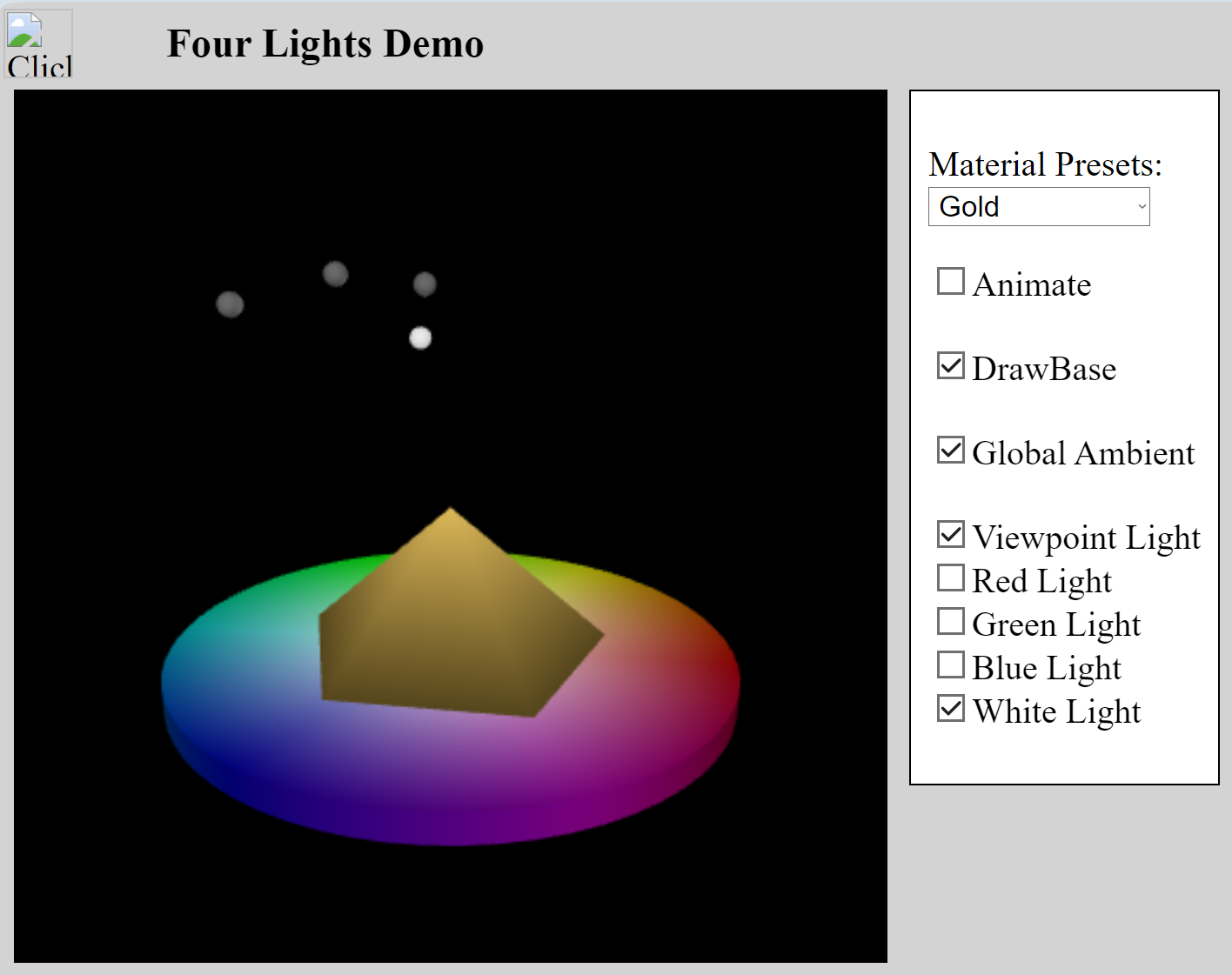
*<div id="helpBG" style="display:none"></div>*

*</body>*

*</html>*

Link do zdalnego repozytorium: https://github.com/Slayzerus/UBB\_GrafikaKomputerowa/tree/main/Lab%207

**4. Wynik działania:**

****

**5. Wnioski:**

Wprowadzenie do oświetlenia i materiałów w OpenGL pokazało, jak istotne jest realistyczne odwzorowanie interakcji światła z obiektami na scenie 3D. Proces ten jest kluczowy dla uzyskania wiarygodnej i atrakcyjnej grafiki. Oświetlenie w OpenGL wymaga włączenia obliczeń świetlnych poprzez funkcję glEnable(GL\_LIGHTING) oraz skonfigurowania źródeł światła, aby uzyskać pożądany efekt wizualny. Użycie materiałów umożliwia definiowanie właściwości powierzchni, co dodatkowo wpływa na realizm sceny.

Materiały w OpenGL są atrybutami wierzchołków, podobnie jak kolor. Każdy wierzchołek może mieć różne właściwości materiału, które są następnie używane w obliczeniach oświetlenia. Istotne jest rozróżnienie między materiałami przedniej i tylnej powierzchni wielokątów, co jest szczególnie ważne przy oświetleniu dwustronnym. Właściwości materiałów, takie jak kolor ambientowy, rozproszony, lustrzany i emisji, mogą być ustawiane za pomocą funkcji glMaterialfv. Z kolei właściwość połysku ustawiana jest za pomocą glMaterialf.

Wektory normalne są niezbędne do poprawnych obliczeń oświetlenia, ponieważ określają kierunek, w jakim powierzchnia odbija światło. Ustawienie bieżącego wektora normalnego odbywa się za pomocą funkcji z rodziny glNormal\*. Wektory te powinny być jednostkowymi wektorami, aby zapewnić prawidłowe wyniki oświetlenia. OpenGL umożliwia automatyczną normalizację wektorów normalnych poprzez glEnable(GL\_NORMALIZE).

OpenGL obsługuje wiele źródeł światła, które mogą być konfigurowane jako światła kierunkowe lub punktowe. Każde źródło światła może mieć swoje własne właściwości, takie jak intensywność rozproszona, lustrzana i ambientowa, ustawiane za pomocą glLightfv. Pozycja źródła światła jest przekształcana przez aktualną transformację modelu widoku, co oznacza, że ustawienie pozycji światła powinno uwzględniać transformacje.

Oprócz właściwości poszczególnych źródeł światła, OpenGL wykorzystuje kilka właściwości globalnych, takich jak globalne światło ambientowe. Domyślnie globalne światło ambientowe jest czarne, ale może być zmienione za pomocą glLightModelfv. Inne ważne właściwości globalne to GL\_LIGHT\_MODEL\_TWO\_SIDE i GL\_LIGHT\_MODEL\_LOCAL\_VIEWER, które kontrolują odpowiednio oświetlenie dwustronne i lokalny punkt widzenia.