**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,

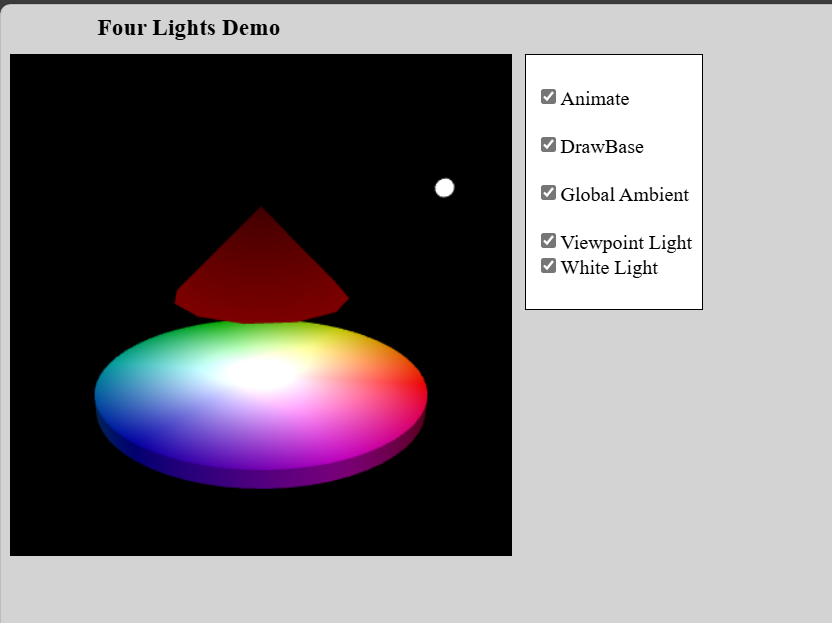
Gr.3b

# 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

# Wyniki zadania:

****

Na podstawie podanych plików należało wykonać zadanie z polecenia

# Wykorzystane komendy:

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

Kod Programu:

1. <!DOCTYPE html>

2. <html>

3. <head>

4. <meta charset="UTF-8">

5. <title>Four Lights</title>

6. <link rel="stylesheet" href="../demo.css">

7. <script src="../script/demo-core.js"></script>

8. <script src="../script/glsim.js"></script>

9. <script src="../script/pyramid-model-IFS.js"></script>

10. <script>

11.

12. var camera;

13.

14. var animate;

15. var drawBase;

16. var ambientLight;

17. var viewpointLight, whiteLight;

18.

19. var animating = false;

20. var frameNumber = 0;

21.

22. const redPlasticMaterial = {

23.     ambient: [0.0, 0.0, 0.0, 1.0],

24.     diffuse: [0.5, 0.0, 0.0, 1.0],

25.     specular: [0.7, 0.6, 0.6, 1.0],

26.     shininess: 0.25 \* 128,

27. };

28.

29. function applyMaterial(material) {

30.     glMaterialfv(GL\_FRONT\_AND\_BACK, GL\_AMBIENT, material.ambient);

31.     glMaterialfv(GL\_FRONT\_AND\_BACK, GL\_DIFFUSE, material.diffuse);

32.     glMaterialfv(GL\_FRONT\_AND\_BACK, GL\_SPECULAR, material.specular);

33.     glMateriali(GL\_FRONT\_AND\_BACK, GL\_SHININESS, material.shininess);

34. }

35.

36. function uvSphere(radius, slices, stacks) {

37.     var i, j;

38.     for (j = 0; j < stacks; j++) {

39.         var latitude1 = (Math.PI / stacks) \* j - Math.PI / 2;

40.         var latitude2 = (Math.PI / stacks) \* (j + 1) - Math.PI / 2;

41.         var sinLat1 = Math.sin(latitude1);

42.         var cosLat1 = Math.cos(latitude1);

43.         var sinLat2 = Math.sin(latitude2);

44.         var cosLat2 = Math.cos(latitude2);

45.         glBegin(GL\_TRIANGLE\_STRIP);

46.         for (i = 0; i <= slices; i++) {

47.             var longitude = (2 \* Math.PI / slices) \* i;

48.             var sinLong = Math.sin(longitude);

49.             var cosLong = Math.cos(longitude);

50.             var x1 = cosLong \* cosLat1;

51.             var y1 = sinLong \* cosLat1;

52.             var z1 = sinLat1;

53.             var x2 = cosLong \* cosLat2;

54.             var y2 = sinLong \* cosLat2;

55.             var z2 = sinLat2;

56.             glNormal3d(x2, y2, z2);

57.             glVertex3d(radius \* x2, radius \* y2, radius \* z2);

58.             glNormal3d(x1, y1, z1);

59.             glVertex3d(radius \* x1, radius \* y1, radius \* z1);

60.         }

61.         glEnd();

62.     }

63. }

64.

65. function lights() {

66.     glColor3d(0.5, 0.5, 0.5);

67.     var zero = [0, 0, 0, 1];

68.     glMaterialfv(GL\_FRONT\_AND\_BACK, GL\_SPECULAR, zero);

69.

70.     if (viewpointLight.checked)

71.         glEnable(GL\_LIGHT0);

72.     else

73.         glDisable(GL\_LIGHT0);

74.

75.     if (whiteLight.checked) {

76.         glMaterialfv(GL\_FRONT\_AND\_BACK, GL\_EMISSION, [1, 1, 1, 1]);

77.         glEnable(GL\_LIGHT1);

78.     } else {

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

80.         glDisable(GL\_LIGHT1);

81.     }

82.     glPushMatrix();

83.     glRotated(-frameNumber, 0, 1, 0);

84.     glTranslated(10, 7, 0, 1);

85.     glLightfv(GL\_LIGHT1, GL\_POSITION, zero);

86.     uvSphere(0.5, 16, 8);

87.     glPopMatrix();

88.

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

90. }

91.

92. function colorArrayForHue(h, s, v) {

93.     var r, g, b;

94.     var c, x;

95.     h = h \* 359;

96.     c = v \* s;

97.     x = (h < 120) ? h / 60 : (h < 240) ? (h - 120) / 60 : (h - 240) / 60;

98.     x = c \* (1 - Math.abs(x - 1));

99.     x += (v - c);

100.     switch (Math.floor(h / 60)) {

101.         case 0: r = v; g = x; b = v - c; break;

102.         case 1: r = x; g = v; b = v - c; break;

103.         case 2: r = v - c; g = v; b = x; break;

104.         case 3: r = v - c; g = x; b = v; break;

105.         case 4: r = x; g = v - c; b = v; break;

106.         case 5: r = v; g = v - c; b = x; break;

107.     }

108.     var array = new Array(4);

109.     array[0] = r;

110.     array[1] = g;

111.     array[2] = b;

112.     array[3] = 1;

113.     return array;

114. }

115.

116. function drawCylinder() {

117.     var i;

118.     var rgba;

119.     glBegin(GL\_TRIANGLE\_STRIP);

120.     for (i = 0; i <= 64; i++) {

121.         var angle = 2 \* Math.PI / 64 \* i;

122.         var x = Math.cos(angle);

123.         var y = Math.sin(angle);

124.         rgba = colorArrayForHue(i / 64.0, 1, 0.6);

125.         glColor3dv(rgba);

126.         glNormal3d(x, y, 0);

127.         glVertex3d(x, y, 1);

128.         glVertex3d(x, y, -1);

129.     }

130.     glEnd();

131.     glNormal3d(0, 0, 1);

132.     glBegin(GL\_TRIANGLE\_FAN);

133.     glColor3d(1, 1, 1);

134.     glVertex3d(0, 0, 1);

135.     for (i = 0; i <= 64; i++) {

136.         var angle = 2 \* Math.PI / 64 \* i;

137.         var x = Math.cos(angle);

138.         var y = Math.sin(angle);

139.         rgba = colorArrayForHue(i / 64.0, 1, 0.6);

140.         glColor3dv(rgba);

141.         glVertex3d(x, y, 1);

142.     }

143.     glEnd();

144.     glNormal3f(0, 0, -1);

145.     glBegin(GL\_TRIANGLE\_FAN);

146.     glColor3d(1, 1, 1);

147.     glVertex3d(0, 0, -1);

148.     for (i = 64; i >= 0; i--) {

149.         var angle = 2 \* Math.PI / 64 \* i;

150.         var x = Math.cos(angle);

151.         var y = Math.sin(angle);

152.         rgba = colorArrayForHue(i / 64.0, 1, 0.6);

153.         glColor3dv(rgba);

154.         glVertex3d(x, y, -1);

155.     }

156.     glEnd();

157. }

158.

159. function display() {

160.     glClearColor(0, 0, 0, 1);

161.     glClear(GL\_COLOR\_BUFFER\_BIT | GL\_DEPTH\_BUFFER\_BIT);

162.

163.     camera.apply();

164.

165.     lights();

166.

167.     if (ambientLight.checked) {

168.         glLightModelfv(GL\_LIGHT\_MODEL\_AMBIENT, [0.15, 0.15, 0.15, 1]);

169.     } else {

170.         glLightModelfv(GL\_LIGHT\_MODEL\_AMBIENT, [0, 0, 0, 1]);

171.     }

172.

173.     if (drawBase.checked) {

174.         glMaterialfv(GL\_FRONT\_AND\_BACK, GL\_SPECULAR, [0, 0, 0, 1]);

175.

176.         glPushMatrix();

177.         glTranslated(0, -5, 0);

178.         glRotated(-90, 1, 0, 0);

179.         glScaled(10, 10, 0.5);

180.         drawCylinder();

181.         glPopMatrix();

182.     }

183.

184.     glPushMatrix();

185.     glTranslatef(0, 0.68, 0);

186.     glScalef(0.65, 0.65, 0.65);

187.     glsimDrawModel(pyramidModel);

188.     glPopMatrix();

189. }

190.

191. function initGL() {

192.     glClearColor(0, 0, 0, 1);

193.     glEnable(GL\_DEPTH\_TEST);

194.     glEnable(GL\_LIGHTING);

195.     glEnable(GL\_LIGHT0);

196.     glEnable(GL\_NORMALIZE);

197.     glEnable(GL\_COLOR\_MATERIAL);

198.     glLightModeli(GL\_LIGHT\_MODEL\_LOCAL\_VIEWER, 1);

199.     glMateriali(GL\_FRONT\_AND\_BACK, GL\_SHININESS, 32);

200.

201.     var dim = [0.5, 0.5, 0.5, 1];

202.     glLightfv(GL\_LIGHT0, GL\_DIFFUSE, dim);

203.     glLightfv(GL\_LIGHT0, GL\_SPECULAR, dim);

204.

205.     var white = [1, 1, 1, 1];

206.     var whitea = [0.2, 0.2, 0.2, 1];

207.     glLightfv(GL\_LIGHT1, GL\_AMBIENT, whitea);

208.     glLightfv(GL\_LIGHT1, GL\_DIFFUSE, white);

209.     glLightfv(GL\_LIGHT1, GL\_SPECULAR, white);

210. }

211.

212. function doFrame() {

213.     if (animating) {

214.         frameNumber++;

215.         display();

216.         setTimeout(doFrame, 30);

217.     }

218. }

219.

220. function init() {

221.     try {

222.         glsimUse("maincanvas");

223.     } catch (e) {

224.         document.getElementById("canvas-holder").innerHTML =

225.             "<p><b>Sorry, an error occurred:<br>" + e + "</b></p>";

226.         return;

227.     }

228.     animate = document.getElementById("animate");

229.     drawBase = document.getElementById("drawBase");

230.     ambientLight = document.getElementById("ambientLight");

231.     viewpointLight = document.getElementById("viewpointLight");

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

233.     animate.checked = false;

234.     drawBase.checked = true;

235.     ambientLight.checked = true;

236.     viewpointLight.checked = true;

237.     whiteLight.checked = true;

238.     drawBase.onchange = display;

239.     ambientLight.onchange = display;

240.     viewpointLight.onchange = display;

241.     whiteLight.onchange = display;

242.     animate.onchange = function () {

243.         if (animate.checked) {

244.             animating = true;

245.             doFrame();

246.         } else {

247.             animating = false;

248.         }

249.     };

250.     initGL();

251.     camera = new Camera();

252.     camera.lookAt(5, 10, 30, 0, 0, 0, 0, 1, 0);

253.     camera.setScale(15);

254.     camera.installTrackball(display);

255.     display();

256. }

257.

258. </script>

259. </head>

260. <body onload="init()">

261.

262. <div id="content">

263.

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

265.

266. <div id="canvas-holder">

267. <canvas id="maincanvas" width="400" height="400"></canvas>

268. </div>

269.

270. <div id="tools">

271. <p>

272. <label><input type="checkbox" id="animate">Animate</label><br><br>

273. <label><input type="checkbox" id="drawBase">DrawBase</label><br><br>

274. <label><input type="checkbox" id="ambientLight">Global Ambient</label><br><br>

275. <label><input type="checkbox" id="viewpointLight">Viewpoint Light</label><br>

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

277. </p>

278. </div>

279. </div>

280. </body>

281. </html>

282.

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.

# 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.