**SPRAWOZDANIE**

Zajęcia: Grafika komputerowa

Prowadzący: prof. dr hab. Vasyl Martsenyuk

**Laboratorium II**

Data

**Temat: Zadanie\_teksturyGL**

**Wariant 8 + 4**

Jakub Bąk

Informatyka I stopień,

stacjonarne,

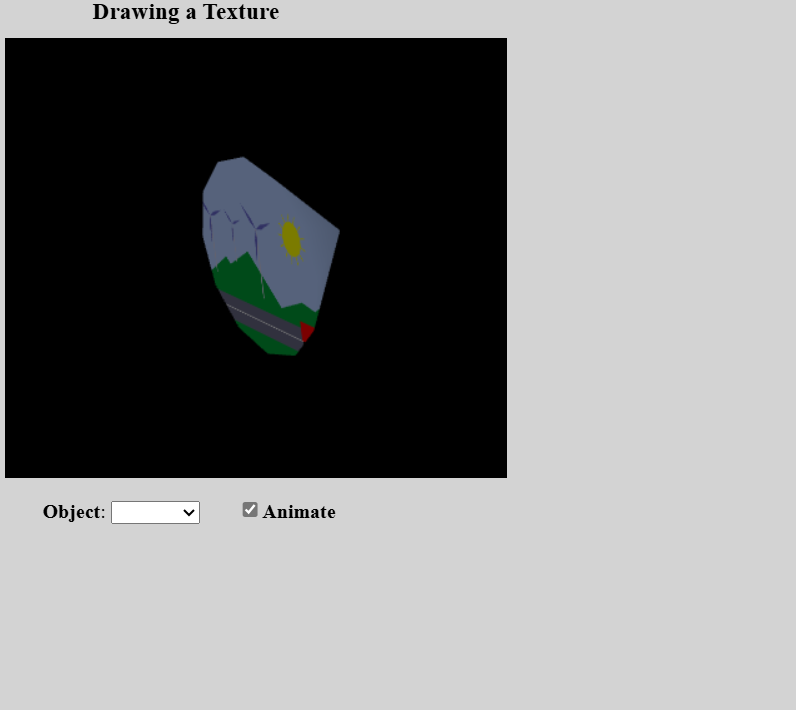
4 semestr,

Gr.3b

# Polecenie

Celem jest teksturowanie piramidy z użyciem dwóch sposobów ładowania tekstur:  użycie tekstury z buforu kolorów  (rysowanie w Panel); ładowanie tekstury z pliku (trzy pliki przykładowe do pobrania)

# Wyniki zadania:

****

# 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>Texture From Color Buffer</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 src="../script/basic-object-models-IFS.js"></script>

11. <script>

12. var camera;

13. var canvas;

14. var frameNumber = 0;

15. var pyramid;

16.

17. function draw() {

18.

19. var objectNumber = Number(document.getElementById("object").value);

20. glDisable(GL\_LIGHTING);

21. glDisable(GL\_DEPTH\_TEST);

22. glDisable(GL\_TEXTURE\_2D);

23. glViewport(0,0,256,256);

24. glMatrixMode(GL\_PROJECTION);

25. glLoadIdentity();

26. glOrtho( 0,7, -1,5, -1,1 );

27. glMatrixMode(GL\_MODELVIEW);

28. draw2DScene();

29. if (objectNumber == 6) {

30. return;

31. }

32. glCopyTexImage2D(GL\_TEXTURE\_2D, 0, GL\_RGBA, 0, 0, 256, 256, 0);

33. glTexParameteri(GL\_TEXTURE\_2D, GL\_TEXTURE\_MIN\_FILTER, GL\_LINEAR);

34. glEnable(GL\_LIGHTING);

35. glEnable(GL\_DEPTH\_TEST);

36. glEnable(GL\_TEXTURE\_2D);

37. glViewport(0,0,canvas.width,canvas.height);

38. camera.apply();

39. glClearColor( 0, 0, 0, 1 );

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

41. switch(objectNumber) {

42. case 0:

43. glScalef(0.06, 0.06, 0.06);

44. drawModel(pyramid);

45. break;

46. }

47. }

48. function drawModel(model) {

49. glEnableClientState(GL\_VERTEX\_ARRAY);

50. glVertexPointer(3,GL\_FLOAT,0,model.vertexPositions);

51. glEnableClientState(GL\_NORMAL\_ARRAY);

52. glNormalPointer(GL\_FLOAT, 0, model.vertexNormals);

53. glEnableClientState(GL\_TEXTURE\_COORD\_ARRAY);

54. glTexCoordPointer(2,GL\_FLOAT,0,model.vertexTextureCoords);

55. glDrawElements(GL\_TRIANGLES, model.indices.length, GL\_UNSIGNED\_BYTE, model.indices);

56. glDisableClientState(GL\_VERTEX\_ARRAY);

57. glDisableClientState(GL\_NORMAL\_ARRAY);

58. glDisableClientState(GL\_TEXTURE\_COORD\_ARRAY);

59. }

60. function initGL() {

61. glEnable(GL\_LIGHT0);

62. glEnable(GL\_NORMALIZE);

63. glMaterialfv(GL\_FRONT\_AND\_BACK, GL\_AMBIENT\_AND\_DIFFUSE, [ 1, 1, 1,1 ]);// white material for texturing.

64.

65. }

66. function drawDisk(radius) {

67. var d;

68. glBegin(GL\_POLYGON);

69. for (d = 0; d < 32; d++) {

70. var angle = 2\*Math.PI/32 \* d;

71. glVertex2d( radius\*Math.cos(angle), radius\*Math.sin(angle));

72. }

73. glEnd();

74. }

75. function drawWheel() {

76. var i;

77. glColor3f(0,0,0);

78. drawDisk(1);

79. glColor3f(0.75, 0.75, 0.75);

80. drawDisk(0.8);

81. glColor3f(0,0,0);

82. drawDisk(0.2);

83. glRotatef(frameNumber\*20,0,0,1);

84. glBegin(GL\_LINES);

85. for (i = 0; i < 15; i++) {

86. glVertex2f(0,0);

87. glVertex2d(Math.cos(i\*2\*Math.PI/15), Math.sin(i\*2\*Math.PI/15));

88. }

89. glEnd();

90. }

91. function drawCart() {

92. glPushMatrix();

93. glTranslatef(-1.5, -0.1, 0);

94. glScalef(0.8,0.8,1);

95. drawWheel();

96. glPopMatrix();

97. glPushMatrix();

98. glTranslatef(1.5, -0.1, 0);

99. glScalef(0.8,0.8,1);

100. drawWheel();

101. glPopMatrix();

102. glColor3f(1,0,0);

103. glBegin(GL\_POLYGON);

104. glVertex2f(-2.5,0);

105. glVertex2f(2.5,0);

106. glVertex2f(2.5,2);

107. glVertex2f(-2.5,2);

108. glEnd();

109. }

110. function drawSun() {

111. var i;

112. glColor3f(1,1,0);

113. for (i = 0; i < 13; i++) { // Draw 13 rays, with different rotations.

114. glRotatef( 360 / 13, 0, 0, 1 ); // Note that the rotations accumulate!

115. glBegin(GL\_LINES);

116. glVertex2f(0, 0);

117. glVertex2f(0.75, 0);

118. glEnd();

119. }

120. drawDisk(0.5);

121. glColor3f(0,0,0);

122. }

123. function drawWindmill() {

124. var i;

125. glColor3f(0.8, 0.8, 0.9);

126. glBegin(GL\_POLYGON);

127. glVertex2f(-0.05, 0);

128. glVertex2f(0.05, 0);

129. glVertex2f(0.05, 3);

130. glVertex2f(-0.05, 3);

131. glEnd();

132. glTranslatef(0, 3, 0);

133. glRotated(frameNumber \* (180.0/46), 0, 0, 1);

134. glColor3f(0.4, 0.4, 0.8);

135. for (i = 0; i < 3; i++) {

136. glRotated(120, 0, 0, 1); // Note: These rotations accumulate.

137. glBegin(GL\_POLYGON);

138. glVertex2f(0,0);

139. glVertex2f(0.5, 0.1);

140. glVertex2f(1.5,0);

141. glVertex2f(0.5, -0.1);

142. glEnd();

143. }

144. }

145. function draw2DScene() {

146. glClearColor( 0.7, 0.8, 1.0, 1.0 );

147. glClear(GL\_COLOR\_BUFFER\_BIT);

148. glLoadIdentity();

149. glColor3f(0, 0.6, 0.2);

150. glBegin(GL\_POLYGON);

151. glVertex2f(-3,-1);

152. glVertex2f(1.5,1.65);

153. glVertex2f(5,-1);

154. glEnd();

155. glBegin(GL\_POLYGON);

156. glVertex2f(-3,-1);

157. glVertex2f(3,2.1);

158. glVertex2f(7,-1);

159. glEnd();

160. glBegin(GL\_POLYGON);

161. glVertex2f(0,-1);

162. glVertex2f(6,1.2);

163. glVertex2f(20,-1);

164. glEnd();

165. glColor3f(0.4, 0.4, 0.5);

166. glBegin(GL\_POLYGON);

167. glVertex2f(0,-0.4);

168. glVertex2f(7,-0.4);

169. glVertex2f(7,0.4);

170. glVertex2f(0,0.4);

171. glEnd();

172. glLineWidth(4);

173. glColor3f(1,1,1);

174. glBegin(GL\_LINES);

175. glVertex2f(0,0);

176. glVertex2f(7,0);

177. glEnd();

178. glLineWidth(1);

179. glPushMatrix();

180. glTranslated(5.8,3,0);

181. glRotated(-frameNumber\*0.7,0,0,1);

182. drawSun();

183. glPopMatrix();

184. glPushMatrix();

185. glTranslated(0.75,1,0);

186. glScaled(0.6,0.6,1);

187. drawWindmill();

188. glPopMatrix();

189. glPushMatrix();

190. glTranslated(2.2,1.6,0);

191. glScaled(0.4,0.4,1);

192. drawWindmill();

193. glPopMatrix();

194. glPushMatrix();

195. glTranslated(3.7,0.8,0);

196. glScaled(0.7,0.7,1);

197. drawWindmill();

198. glPopMatrix();

199. glPushMatrix();

200. glTranslated(-3 + 13\*(frameNumber % 300) / 300.0, 0, 0);

201. glScaled(0.3,0.3,1);

202. drawCart();

203. glPopMatrix();

204. }

205. var animating = false;

206. function frame() {

207. if (animating) {

208. frameNumber++;

209. draw();

210. setTimeout(frame,30);

211. }

212. }

213. function doAnimate() {

214. animating = document.getElementById("animate").checked;

215. if (animating) {

216. frame();

217. }

218. }

219. function init() {

220. try {

221. canvas = document.getElementById("maincanvas");

222. glsimUse(canvas,null);

223. }

224. catch (e) {

225. document.getElementById("canvas-holder").innerHTML="<p><b>Sorry, an error occurred:<br>" +

226. e + "</b></p>";

227. return;

228. }

229. initGL();

230. document.getElementById("object").value = "1";

231. document.getElementById("object").onchange = draw;

232. document.getElementById("animate").checked = false;

233. document.getElementById("animate").onchange = doAnimate;

234. camera = new Camera();

235. camera.setScale(1);

236. camera.lookAt(2,2,5, 0,0,0, 0,1,0);

237. camera.installTrackball(draw);

238. sphere = uvSphere();

239. cubeModel = cube();

240. cylinder = uvCylinder();

241. cone = uvCone();

242. torus = uvTorus();

243. pyramid = pyramidModel;

244. draw();

245. }

246. </script>

247. </head>

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

249. <div id="content">

250. <h3 id="headline">Drawing a Texture</h3>

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

252. <canvas id="maincanvas" width="400" height="350"></canvas>

253. </div>

254. <br clear=all>

255. <p style="text-indent:30px"><b>Object</b>:

256. <select id="object">

257. <option value="0">Pyramid</option>

258. </select>

259. <label><input type="checkbox" id="animate" style="margin-left:30px"><b>Animate</b></label></p>

260. </div>

261. </body>

262. </html>

263.

Program to aplikacja internetowa wykorzystująca WebGL do renderowania sceny 3D i 2D na płótnie HTML `<canvas>`. Umożliwia użytkownikowi wybór obiektu (np. piramidy) do wyświetlenia oraz opcjonalne animowanie sceny. Scena 2D zawiera elementy takie jak słońce, wiatraki, wózek i krajobraz, natomiast scena 3D renderuje wybrany obiekt z teksturą. Program obsługuje kamerę z możliwością manipulacji (np. skalowanie, obrót) oraz animację w pętli.

# Wnioski:

Kod demonstruje wykorzystanie tekstur w WebGL poprzez kopiowanie zawartości bufora kolorów (`glCopyTexImage2D`) i używanie jej jako tekstury na obiektach 3D. Tekstury są generowane dynamicznie na podstawie sceny 2D, co pozwala na tworzenie złożonych efektów wizualnych. Ustawienia takie jak `GL\_LINEAR` dla filtrowania tekstur zapewniają płynne przejścia między pikselami.