SPRAWOZDANIE

Zajęcia: Grafika komputerowa

Prowadzący: prof. dr hab. Vasyl Martsenyuk

Laboratorium: 5

Data: 20.04

Temat: "Geometria tr´ojwymiarowa biblioteki OpenGL";

Wariant: black; z

Przemysław Holisz

Informatyka I stopień, stacjonarne,

4 semestr, Gr.4/2b

1. Polecenie:

Stworzyć dwa obiekty przy użyciu OpenGL (w języku C lub Java). Po uruchomieniu zakończonego programu naciśnięcie jednego z klawiszy numerycznych 1 lub 2 spowoduje wybranie wyświetlanego obiektu. Program już ustawia wartość zmiennej globalnej, objectNumber, aby powiedzieć, który obiekt ma zostać narysowany. Użytkownik może obracać obiekt za pomocą klawiszy strzałek, PageUp, PageDown i Home. Podprogram display() jest wywoływany, aby narysować obiekt. Podprogram ten z kolei wywołuje draw() i właśnie w draw() powinieneś wykonać podstawową pracę.

2. Wprowadzane dane:

```
3.  package lab5;
4.
5.  import java.awt.*;
6.  import javax.swing.*;
7.  import java.awt.event.*;
8.  import com.jogamp.opengl.*;
9.  import com.jogamp.opengl.awt.*;
10.
11. public class lab5 extends GLJPanel implements GLEventListener, KeyListener
    {
12.
13.       private static final long serialVersionUID = 1L;
14.
15.       public static void main(String[] args) {
16.             JFrame window = new JFrame("Some Objects in 3D");
17.             lab5 panel = new lab5();
18.             window.setContentPane(panel);
19.             window.pack();
20.             window.setResizable(false);
21.             window.setLocation(50, 50);
22.             window.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
23.             window.setVisible(true);
24.       }
25.
26.       public lab5() {
27.             super(new GLCapabilities(null));
28.             setPreferredSize(new Dimension(700, 700));
29.             addGLEventListener(this);
30.             addKeyListener(this);
31.       }
32.
33.       private int objectNumber = 1;
34.
35.       private boolean useAnaglyph = false;
36.
37.       private int rotateX = 0;
38.       private int rotateY = 0;
39.       private int rotateZ = 0;
40.
41.       private void pyramidWalls(float n, GL2 gl2) {
42.             float deg = 360 / n;
43.             gl2.glBegin(GL.GL_TRIANGLE_FAN);
44.             gl2.glVertex3d(0, 0, 0);
45.             for (float i = 1; i <= n + 1; i++) {
```

```
46.                   gl2.glVertex3d(Math.cos(Math.toRadians(i * deg)),
   Math.sin(Math.toRadians(i * deg)), 2);
47.             }
48.             gl2.glEnd();
49.       }
50.
51.       private void pyramidBase(float n, GL2 gl2) {
52.             float deg = 360 / n;
53.             gl2.glBegin(GL.GL_TRIANGLE_FAN);
54.             gl2.glVertex3d(0, 0, 2);
55.             for (float i = 1; i <= n + 1; i++) {
56.                   gl2.glVertex3d(Math.cos(Math.toRadians(i * deg)),
   Math.sin(Math.toRadians(i * deg)), 2);
57.             }
58.             gl2.glEnd();
59.       }
60.
61.       private void pyramid(float n, float scale, GL2 gl2) {
62.             gl2.glColor3f(0, 0, (float) 0);
63.             gl2.glScalef(scale, scale, scale);
64.             gl2.glRotatef(0, 0, 90, 0);
65.             gl2.glTranslatef(0, 0, -1);
66.
67.             pyramidWalls(n, gl2);
68.             pyramidBase(n, gl2);
69.       }
70.
71.       private void corkscrew(int n, float scale, GL2 gl2) {
72.             gl2.glColor3f(0, 0, (float) 0);
73.             gl2.glScalef(scale, scale, scale);
74.             gl2.glLineWidth(5);
75.             gl2.glRotatef(0, 0, 90, 0);
76.             gl2.glTranslatef(0, 0, -1);
77.
78.             gl2.glBegin(GL.GL_LINE_STRIP);
79.             int res = 36;
80.             float deg = 360 / res;
81.
82.             for (float i = 1; i <= n * res; i++) {
83.                   double x = Math.cos(Math.toRadians(i * deg));
84.                   double y = Math.sin(Math.toRadians(i * deg));
85.                   gl2.glVertex3d(x * (0.01f * i), y * (0.01f * i), (i /
   res) - (n / 2));
86.             }
87.             gl2.glEnd();
88.       }
89.
90.       private void draw(GL2 gl2) {
91.
92.             gl2.glRotatef(rotateZ, 0, 0, 1);
93.             gl2.glRotatef(rotateY, 0, 1, 0);
94.             gl2.glRotatef(rotateX, 1, 0, 0);
95.
96.             switch (objectNumber) {
97.             case 1:
98.                   corkscrew(15, 1, gl2);
99.                   break;
100.                  case 2:
```

```
101.                         pyramid(15, 3, gl2);
102.                         break;
103.                   }
104.
105.            }
106.
107.            public void display(GLAutoDrawable drawable) {
108.
109.                  GL2 gl2 = drawable.getGL().getGL2();
110.
111.                  if (useAnaglyph) {
112.                        gl2.glDisable(GL2.GL_COLOR_MATERIAL);
113.                        gl2.glMaterialfv(GL2.GL_FRONT_AND_BACK,
    GL2.GL_AMBIENT_AND_DIFFUSE, new float[] { 1, 1, 1, 1 }, 0);
114.                  } else {
115.                        gl2.glEnable(GL2.GL_COLOR_MATERIAL);
116.                  }
117.                  gl2.glNormal3f(0, 0, 1);
118.
119.                  gl2.glClearColor(0, 0, 0, 1);
120.                  gl2.glClear(GL2.GL_COLOR_BUFFER_BIT |
    GL2.GL_DEPTH_BUFFER_BIT);
121.
122.                  if (useAnaglyph == false) {
123.                        gl2.glLoadIdentity();
124.                        gl2.glTranslated(0, 0, -15);
125.                        draw(gl2);
126.                  } else {
127.                        gl2.glLoadIdentity();
128.                        gl2.glColorMask(true, false, false, true);
129.                        gl2.glRotatef(4, 0, 1, 0);
130.                        gl2.glTranslated(1, 0, -15);
131.                        draw(gl2);
132.                        gl2.glColorMask(true, false, false, true);
133.                        gl2.glClear(GL2.GL_DEPTH_BUFFER_BIT);
134.                        gl2.glLoadIdentity();
135.                        gl2.glRotatef(-4, 0, 1, 0);
136.                        gl2.glTranslated(-1, 0, -15);
137.                        gl2.glColorMask(false, true, true, true);
138.                        draw(gl2);
139.                        gl2.glColorMask(true, true, true, true);
140.                  }
141.
142.            }
143.
144.            public void init(GLAutoDrawable drawable) {
145.                  GL2 gl2 = drawable.getGL().getGL2();
146.                  gl2.glMatrixMode(GL2.GL_PROJECTION);
147.                  gl2.glFrustum(-3.5, 3.5, -3.5, 3.5, 5, 25);
148.                  gl2.glMatrixMode(GL2.GL_MODELVIEW);
149.                  gl2.glEnable(GL2.GL_LIGHTING);
150.                  gl2.glEnable(GL2.GL_LIGHT0);
151.                  gl2.glLightfv(GL2.GL_LIGHT0, GL2.GL_DIFFUSE, new float[]
    { 0.7f, 0.7f, 0.7f }, 0);
152.                  gl2.glLightModeli(GL2.GL_LIGHT_MODEL_TWO_SIDE, 1);
153.                  gl2.glEnable(GL2.GL_DEPTH_TEST);
154.                  gl2.glLineWidth(3);
155.            }
```

```
156.
157.            public void dispose(GLAutoDrawable drawable) {
158.            }
159.
160.            public void reshape(GLAutoDrawable drawable, int x, int y, int
    width, int height) {
161.            }
162.
163.            public void keyPressed(KeyEvent evt) {
164.                  int key = evt.getKeyCode();
165.                  boolean repaint = true;
166.                  if (key == KeyEvent.VK_LEFT)
167.                        rotateY -= 6;
168.                  else if (key == KeyEvent.VK_RIGHT)
169.                        rotateY += 6;
170.                  else if (key == KeyEvent.VK_DOWN)
171.                        rotateX += 6;
172.                  else if (key == KeyEvent.VK_UP)
173.                        rotateX -= 6;
174.                  else if (key == KeyEvent.VK_PAGE_UP)
175.                        rotateZ += 6;
176.                  else if (key == KeyEvent.VK_PAGE_DOWN)
177.                        rotateZ -= 6;
178.                  else if (key == KeyEvent.VK_HOME)
179.                        rotateX = rotateY = rotateZ = 0;
180.                  else if (key == KeyEvent.VK_1)
181.                        objectNumber = 1;
182.                  else if (key == KeyEvent.VK_2)
183.                        objectNumber = 2;
184.                  else if (key == KeyEvent.VK_3)
185.                        objectNumber = 3;
186.                  else if (key == KeyEvent.VK_4)
187.                        objectNumber = 4;
188.                  else if (key == KeyEvent.VK_5)
189.                        objectNumber = 5;
190.                  else if (key == KeyEvent.VK_6)
191.                        objectNumber = 6;
192.                  else if (key == KeyEvent.VK_SPACE)
193.                        useAnaglyph = !useAnaglyph;
194.                  else
195.                        repaint = false;
196.                  if (repaint)
197.                        repaint();
198.            }
199.
200.            public void keyReleased(KeyEvent evt) {
201.            }
202.
203.            public void keyTyped(KeyEvent evt) {
204.            }
205.
206.      }
```

207. Wynik działania:

208. Wnioski:

Na podstawie otrzymanego wyniku można stwierdzić, że