**SPRAWOZDANIE**

Zajęcia: Grafika komputerowa

Prowadzący: prof. dr hab. Vasyl Martsenyuk

# **Laboratorium**

Data: 25.03.2024

**Temat:**

"Geometria trójwymiarowa OpenGL"

Mateusz Żelazo

Informatyka I stopnia

stacjonarne, 4 semestr

Gr.3a

### 

1. **Polecenie:**

Stworzyć dwa obiekty przy użyciu OpenGL (w języku JavaScript). Po uruchomieniu zakończonego programu naciśnięcie jednego z klawiszy numerycznych 1 lub 2 spowoduje wybranie wyświetlanego obiektu. Program 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.

Obiekt 1. Korkociąg wokół osi {x | y | z} zawierający N obrotów. Punkty są stopniowo powiększane. Ustalić aktualny kolor rysujący na {zielony | niebieski | brązowy | … }.

Obiekt 2. Pyramida, wykorzystując dwa wachlarze trójkątów oraz modelowanie hierarchiczne (najpierw tworzymy podprogramę rysowania jednego trójkonta; dalej wykorzystując przekształcenia geometryczne tworzymy pyramidę). Podstawą pyramidy jest wielokąt o N wierzchówkach.

1. **Wprowadzane dane:**
   1. **Rysowanie piramidy:**

**int w = 11;**

**float kat = 360 / w;**

**int wielkosc = 3;**

**for (int i = 1; i <= w; i++) {**

**gl2.glColor3f( 1.0f, i, i );**

**gl2.glBegin( GL2.GL\_TRIANGLES );**

**gl2.glVertex3f( 0.0f, 3.0f, 0.0f );**

**for (int a = 0; a <= 1; a++) {**

**int s = i + a;**

**if(i >= w) s = 1;**

**double rad = ((kat \* s) \* (2 \* Math.PI)) / 360;**

**float y = (float)Math.sin(rad) \* wielkosc;**

**float x = (float)Math.cos(rad) \* wielkosc;**

**gl2.glVertex3f( x, -1.0f, y );**

**}**

**gl2.glEnd();**

**}**

**// rysuj ostatni trojkat**

**gl2.glColor3f( 1.0f, w, w );**

**gl2.glBegin( GL2.GL\_TRIANGLES );**

**gl2.glVertex3f( 0.0f, 3.0f, 0.0f );**

**double rad\_l = ((kat \* 1) \* (2 \* Math.PI)) / 360;**

**float y\_l = (float)Math.sin(rad\_l) \* wielkosc;**

**float x\_l = (float)Math.cos(rad\_l) \* wielkosc;**

**gl2.glVertex3f( x\_l, -1.0f, y\_l );**

**double rad\_r = ((kat \* w) \* (2 \* Math.PI)) / 360;**

**float y\_r = (float)Math.sin(rad\_r) \* wielkosc;**

**float x\_r = (float)Math.cos(rad\_r) \* wielkosc;**

**gl2.glVertex3f( x\_r, -1.0f, y\_r );**

**gl2.glEnd();**

**gl2.glBegin(GL2.GL\_TRIANGLE\_FAN);**

**gl2.glColor3f( 0.5f, 0.0f, 1.0f );**

**for (int i = 1; i <= w; i++) {**

**double rad = ((kat \* i) \* (2 \* Math.PI)) / 360;**

**float y = (float)Math.sin(rad) \* wielkosc;**

**float x = (float)Math.cos(rad) \* wielkosc;**

**gl2.glVertex3f( x, -1.0f, y );**

**}**

**gl2.glEnd();**

* 1. **Rysowanie korkociągu**

**gl2.glBegin(GL2.GL\_LINE\_STRIP);**

**gl2.glColor3f(1.0f, 1.0f, 0.0f); // Red**

**float x = -0.01f;**

**float n = 11.0f;**

**for (double angle = 0.0f; angle <= (2.0f \* 3.14)\* n; angle += 0.1f)**

**{**

**float z = 0.5f \* (float)Math.sin(angle);**

**float y = 0.5f \* (float)Math.cos(angle);**

**gl2.glVertex3f(x, y, z);**

**x += 0.01f;**

**}**

**gl2.glEnd();**

1. **Wykorzystane komendy:**
   1. **Kod źródłowy**

**import java.awt.\*;**

**import javax.swing.\*;**

**import java.awt.event.\*;**

**import com.jogamp.opengl.\*;**

**import com.jogamp.opengl.awt.\*;**

**import com.jogamp.opengl.util.gl2.GLUT;**

**/\*\***

**\* Lab 4: Some objects in 3D. The arrow keys**

**\* can be used to rotate the object. The number keys 1 through 6**

**\* select the object. The space bar toggles the use of anaglyph**

**\* stereo.**

**\*/**

**public class Lab5 extends GLJPanel implements GLEventListener, KeyListener{**

**/\*\***

**\* A main routine to create and show a window that contains a**

**\* panel of type Lab5. The program ends when the user closes the**

**\* window.**

**\*/**

**public static void main(String[] args) {**

**JFrame window = new JFrame("Some Objects in 3D");**

**Lab5 panel = new Lab5();**

**window.setContentPane(panel);**

**window.pack();**

**window.setResizable(false);**

**window.setLocation(50,50);**

**window.setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);**

**window.setVisible(true);**

**}**

**/\*\***

**\* Constructor for class Lab4.**

**\*/**

**public Lab5() {**

**super( new GLCapabilities(null) ); // Makes a panel with default OpenGL "capabilities".**

**setPreferredSize( new Dimension(700,700) );**

**addGLEventListener(this); // This panel will respond to OpenGL events.**

**addKeyListener(this); // The panel will respond to key events.**

**}**

**//------------------- TODO: Complete this section! ---------------------**

**private int objectNumber = 1; // Which object to draw (1 ,2, 3, 4, 5, or 6)?**

**// (Controlled by number keys.)**

**private boolean useAnaglyph = false; // Should anaglyph stereo be used?**

**// (Controlled by space bar.)**

**private int rotateX = 0; // Rotations of the cube about the axes.**

**private int rotateY = 0; // (Controlled by arrow, PageUp, PageDown keys;**

**private int rotateZ = 0; // Home key sets all rotations to 0.)**

**private GLUT glut = new GLUT(); // An object for drawing GLUT shapes.**

**/\*\***

**\* The method that draws the current object, with its modeling transformation.**

**\*/**

**private void draw(GL2 gl2) {**

**gl2.glRotatef(rotateZ,0,0,1); // Apply rotations to complete object.**

**gl2.glRotatef(rotateY,0,1,0);**

**gl2.glRotatef(rotateX,1,0,0);**

**// TODO: Draw the currently selected object, number 1, 2, 3, 4, 5, or 6.**

**// (Objects should lie in the cube with x, y, and z coordinates in the**

**// range -5 to 5.)**

**}**

**//-------------------- Draw the Scene -------------------------**

**/\*\***

**\* The display method is called when the panel needs to be drawn.**

**\* It's called when the window opens and it is called by the keyPressed**

**\* method when the user hits a key that modifies the scene.**

**\*/**

**public void display(GLAutoDrawable drawable) {**

**GL2 gl2 = drawable.getGL().getGL2(); // The object that contains all the OpenGL methods.**

**if (useAnaglyph) {**

**gl2.glDisable(GL2.GL\_COLOR\_MATERIAL); // in anaglyph mode, everything is drawn in white**

**gl2.glMaterialfv(GL2.GL\_FRONT\_AND\_BACK, GL2.GL\_AMBIENT\_AND\_DIFFUSE, new float[]{1,1,1,1}, 0);**

**}**

**else {**

**gl2.glEnable(GL2.GL\_COLOR\_MATERIAL); // in non-anaglyph mode, glColor\* is respected**

**}**

**gl2.glNormal3f(0,0,1); // (Make sure normal vector is correct for object 1.)**

**gl2.glClearColor( 0, 0, 0, 1 ); // Background color (black).**

**gl2.glClear( GL2.GL\_COLOR\_BUFFER\_BIT | GL2.GL\_DEPTH\_BUFFER\_BIT );**

**if (useAnaglyph == false) {**

**gl2.glLoadIdentity(); // Make sure we start with no transformation!**

**gl2.glTranslated(0,0,-15); // Move object away from viewer (at (0,0,0)).**

**draw(gl2);**

**}**

**else {**

**gl2.glLoadIdentity(); // Make sure we start with no transformation!**

**gl2.glColorMask(true, false, false, true);**

**gl2.glRotatef(4,0,1,0);**

**gl2.glTranslated(1,0,-15);**

**draw(gl2); // draw the current object!**

**gl2.glColorMask(true, false, false, true);**

**gl2.glClear(GL2.GL\_DEPTH\_BUFFER\_BIT);**

**gl2.glLoadIdentity();**

**gl2.glRotatef(-4,0,1,0);**

**gl2.glTranslated(-1,0,-15);**

**gl2.glColorMask(false, true, true, true);**

**draw(gl2);**

**gl2.glColorMask(true, true, true, true);**

**}**

**} // end display()**

**/\*\* The init method is called once, before the window is opened, to initialize**

**\* OpenGL. Here, it sets up a projection, turns on some lighting, and enables**

**\* the depth test.**

**\*/**

**public void init(GLAutoDrawable drawable) {**

**GL2 gl2 = drawable.getGL().getGL2();**

**gl2.glMatrixMode(GL2.GL\_PROJECTION);**

**gl2.glFrustum(-3.5, 3.5, -3.5, 3.5, 5, 25);**

**gl2.glMatrixMode(GL2.GL\_MODELVIEW);**

**gl2.glEnable(GL2.GL\_LIGHTING);**

**gl2.glEnable(GL2.GL\_LIGHT0);**

**gl2.glLightfv(GL2.GL\_LIGHT0,GL2.GL\_DIFFUSE,new float[] {0.7f,0.7f,0.7f},0);**

**gl2.glLightModeli(GL2.GL\_LIGHT\_MODEL\_TWO\_SIDE, 1);**

**gl2.glEnable(GL2.GL\_DEPTH\_TEST);**

**gl2.glLineWidth(3); // make wide lines for the stellated dodecahedron.**

**}**

**public void dispose(GLAutoDrawable drawable) {**

**// called when the panel is being disposed**

**}**

**public void reshape(GLAutoDrawable drawable, int x, int y, int width, int height) {**

**// called when user resizes the window**

**}**

**// ---------------- Methods from the KeyListener interface --------------**

**/\*\***

**\* Responds to keypressed events. The four arrow keys control the rotations**

**\* about the x- and y-axes. The PageUp and PageDown keys control the rotation**

**\* about the z-axis. The Home key resets all rotations to zero. The number**

**\* keys 1, 2, 3, 4, 5, and 6 select the current object number. Pressing the space**

**\* bar toggles anaglyph stereo on and off. The panel is redrawn to reflect the**

**\* change.**

**\*/**

**public void keyPressed(KeyEvent evt) {**

**int key = evt.getKeyCode();**

**boolean repaint = true;**

**if ( key == KeyEvent.VK\_LEFT )**

**rotateY -= 6;**

**else if ( key == KeyEvent.VK\_RIGHT )**

**rotateY += 6;**

**else if ( key == KeyEvent.VK\_DOWN)**

**rotateX += 6;**

**else if ( key == KeyEvent.VK\_UP )**

**rotateX -= 6;**

**else if ( key == KeyEvent.VK\_PAGE\_UP )**

**rotateZ += 6;**

**else if ( key == KeyEvent.VK\_PAGE\_DOWN )**

**rotateZ -= 6;**

**else if ( key == KeyEvent.VK\_HOME )**

**rotateX = rotateY = rotateZ = 0;**

**else if (key == KeyEvent.VK\_1)**

**objectNumber = 1;**

**else if (key == KeyEvent.VK\_2)**

**objectNumber = 2;**

**else if (key == KeyEvent.VK\_3)**

**objectNumber = 3;**

**else if (key == KeyEvent.VK\_4)**

**objectNumber = 4;**

**else if (key == KeyEvent.VK\_5)**

**objectNumber = 5;**

**else if (key == KeyEvent.VK\_6)**

**objectNumber = 6;**

**else if (key == KeyEvent.VK\_SPACE)**

**useAnaglyph = ! useAnaglyph;**

**else**

**repaint = false;**

**if (repaint)**

**repaint();**

**}**

**public void keyReleased(KeyEvent evt) {**

**}**

**public void keyTyped(KeyEvent evt) {**

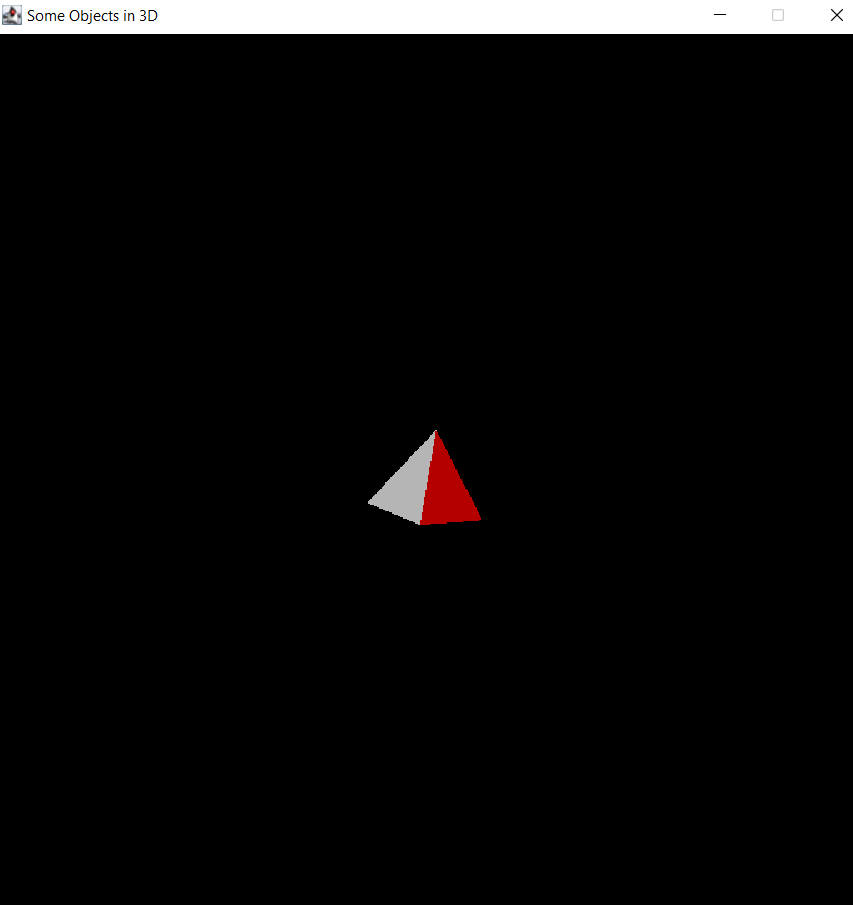
**}**

**} // end class Lab4**

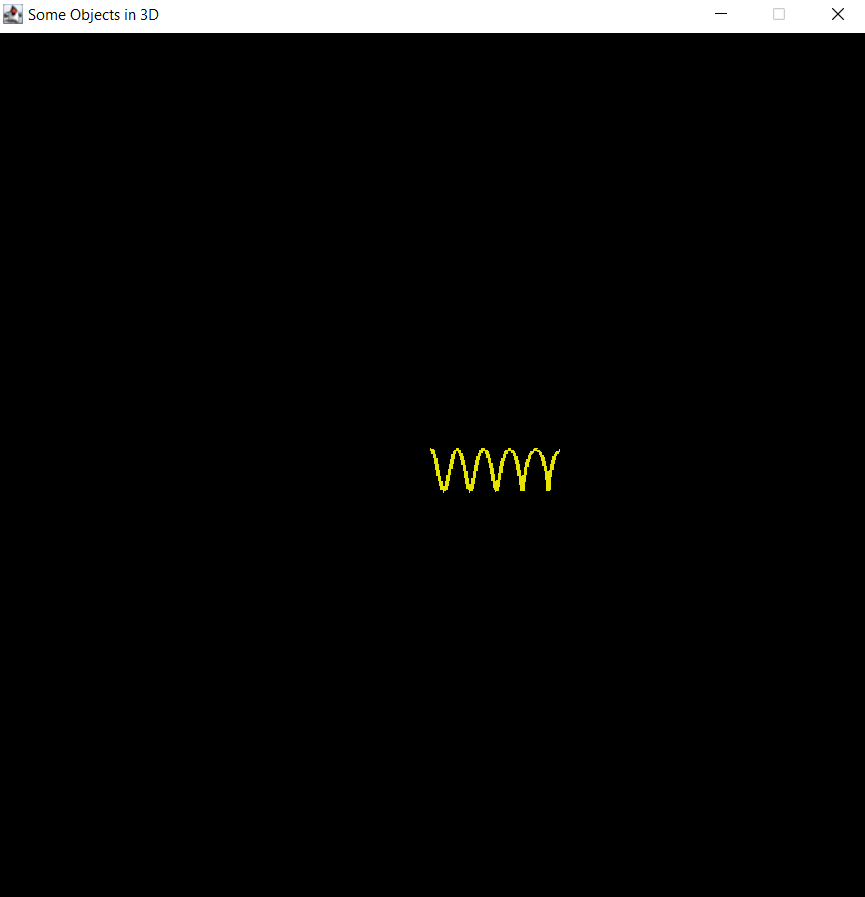
1. **Link do zdalnego repozytorium:**

* https://github.com/Terminalk/GKLab

1. **Wynik działania:** 
   1. **Piramida**

****

* 1. **Korkociąg**

****

1. **Wnioski:**

### Za pomocą OpenGL jesteśmy w względnie łatwo stworzyć proste obiekty 3d, oraz dowolnie je przekształcać.