SPRAWOZDANIE
Zajęcia: Grafika komputerowa
Prowadzący: prof. dr. hab. Vasyl Martsenyuk

Laboratorium 5
25 III 2021 r.
Temat: "Geometria trójwymiarowa OpenGL"
Liczba kątów:5

Robert Laszczak
Informatyka I stopień
Stacjonarne, 4 semestr
Grupa 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. (Miejsce jest oznaczone TODO.) Dodaj również kilka nowych podprogramów do programu.

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. Piramida, wykorzystując dwa wachlarze trójkątów oraz modelowanie hierarchiczne (najpierw tworzymy podprogramę rysowania jednego trójkąta; dalej wykorzystując przekształcenia geometryczne tworzymy piramidę). Podstawą piramidy jest wielokąt o N wierzchówkach.

2. Wprowadzone dane:

Liczba kątów n = 5

3. Wykorzystane komendy:

a) Kod źródłowy:

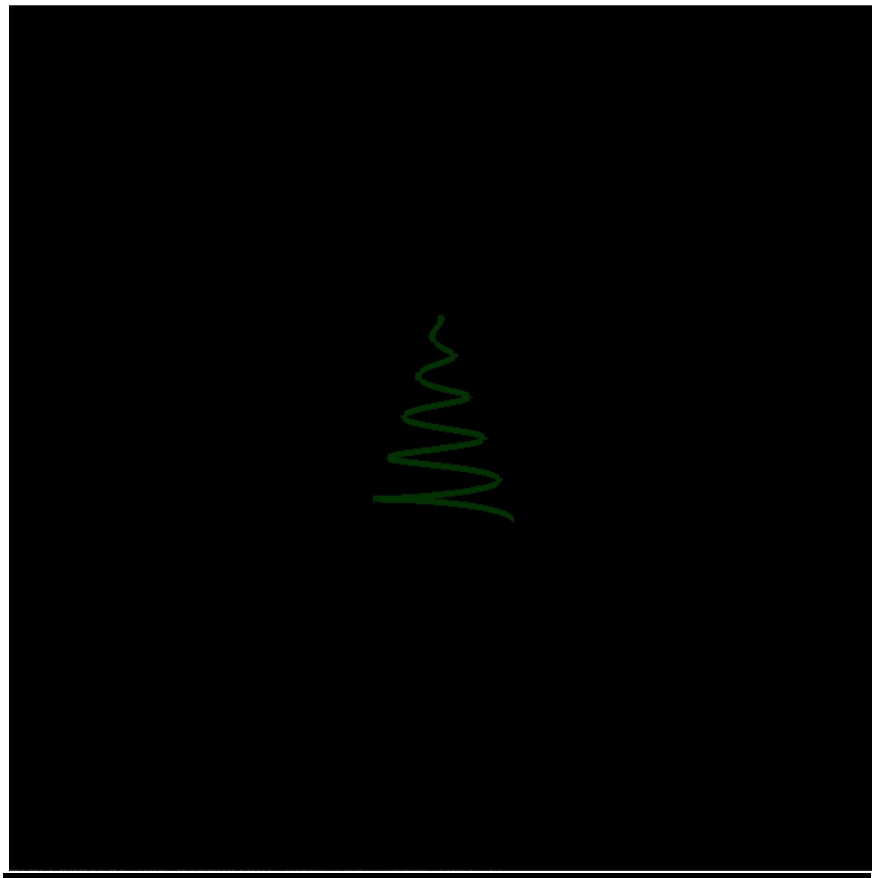
```
package gk;

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;

/**
 * CPSC 424, Fall 2015, 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 SubroutineHierarchy 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");
      SubroutineHierarchy panel = new SubroutineHierarchy();
      window.setContentPane(panel);
      window.pack();
      window.setResizable(false);
      window.setLocation(50,50);
```

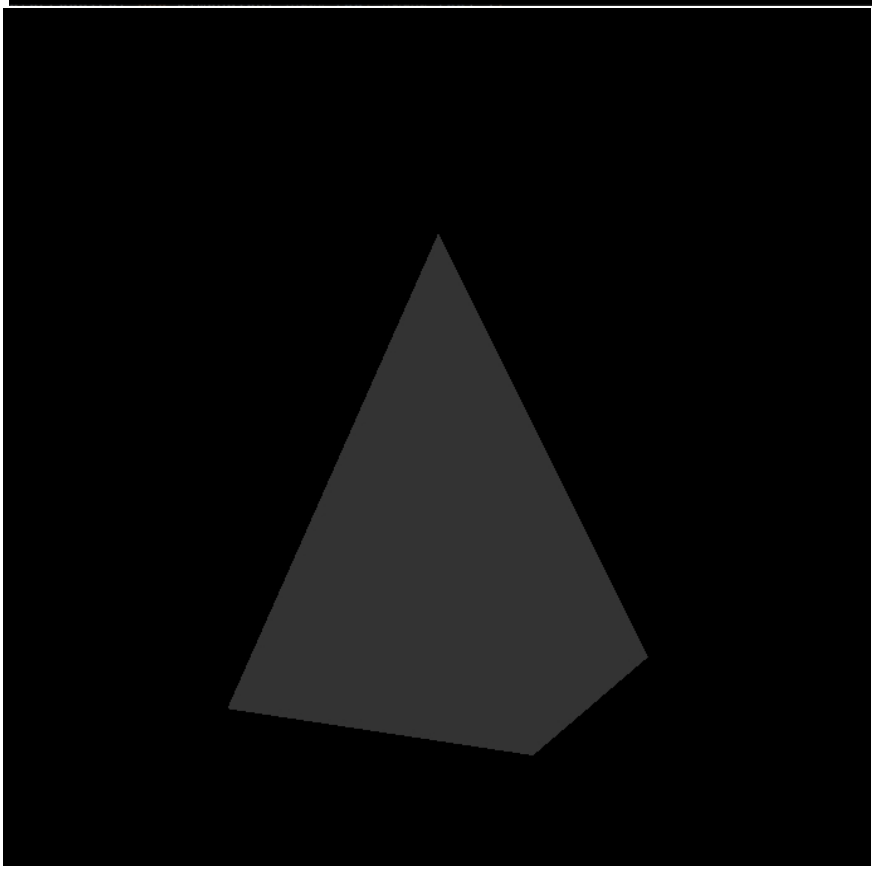
```
      window.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
      window.setVisible(true);
   }

   /**
    * Constructor for class Lab4.
    */
   public SubroutineHierarchy() {
      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 pyramidWalls(float n, GL2 gl2)
   {
      float deg = 360/n;
      double x,y;
      gl2.glBegin(GL.GL_TRIANGLE_FAN);
      gl2.glVertex3d(0,0,0);
      for(float i=1;i<=n+1;i++)
      {
         x= Math.cos(Math.toRadians(deg*i));
         y= Math.sin(Math.toRadians(deg*i));
         gl2.glVertex3d(x, y, 2);
      }
      gl2.glEnd();
   }

   private void pyramidBase(float n, GL2 gl2)
   {
      float deg = 360/n;
      double x,y;
      gl2.glBegin(GL.GL_TRIANGLE_FAN);
      gl2.glVertex3d(0,0,2);
      for(float i=1;i<=n+1;i++)
      {
         x= Math.cos(Math.toRadians(deg*i));
         y= Math.sin(Math.toRadians(deg*i));
         gl2.glVertex3d(x, y, 2);
      }
      gl2.glEnd();
   }

   private void pyramid(float n, float scale, GL2 gl2)
   {
      gl2.glColor3f(1,1,1);
      gl2.glScalef(scale,scale,scale);
      gl2.glRotatef(90,1,0,0);
      gl2.glTranslatef(0,0,-1);

      pyramidWalls(n,gl2);
      pyramidBase(n,gl2);
   }

   private void  corkscrew(int n, float scale,GL2 gl2)
   {
      gl2.glColor3f(0,1,0);
      gl2.glScalef(scale,scale,scale);
      gl2.glLineWidth(5);
      gl2.glRotatef(90,1,0,0);
      gl2.glTranslatef(0,0,-1);

      gl2.glBegin(GL.GL_LINE_STRIP);
      int res = 36;
      float deg = 360/res;
      double x,y;
      for(float i=1;i<=n*res;i++)
```

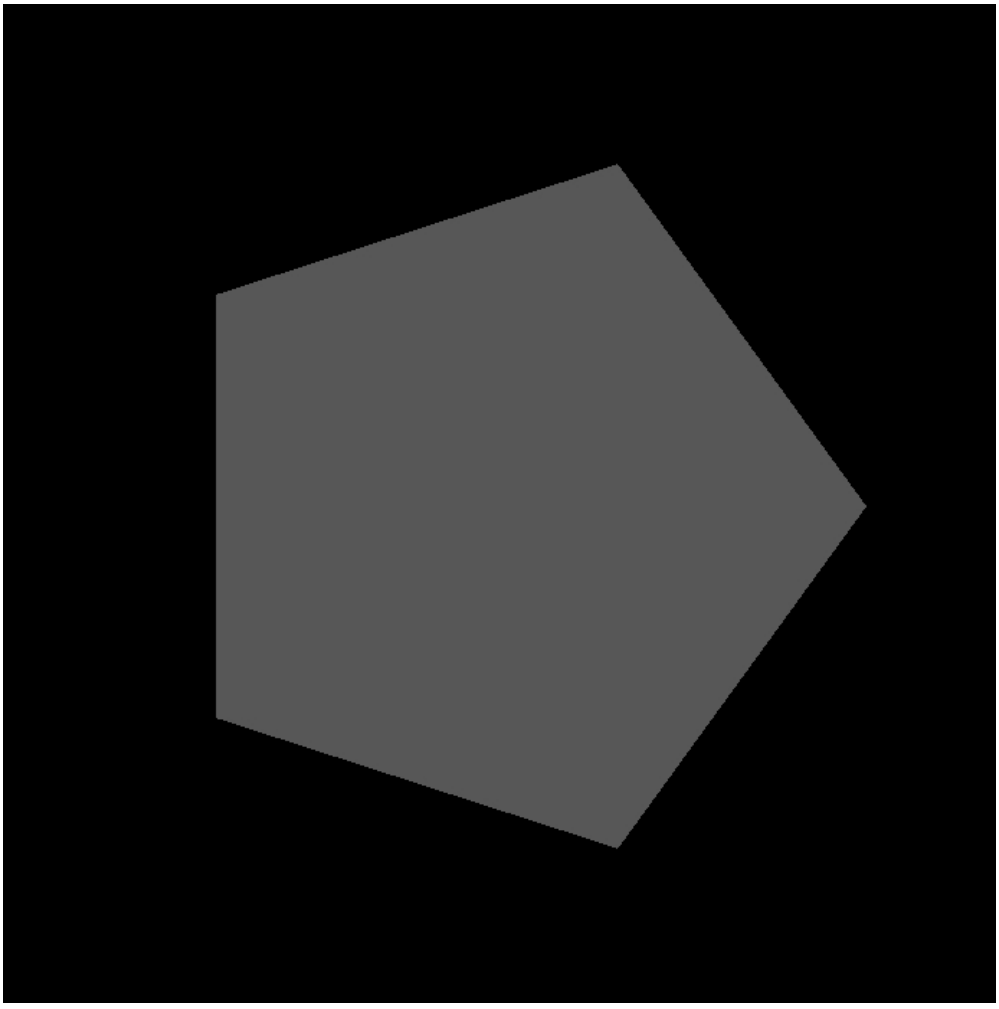
```
      {
         x= Math.cos(Math.toRadians(deg*i));
         y= Math.sin(Math.toRadians(deg*i));
         gl2.glVertex3d(x*(0.01f*i), y*(0.01f*i), (i/res)-(n/2));
      }
      gl2.glEnd();
   }

   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.)

      switch(objectNumber)
      {
         case 1:
            corkscrew(5,1,gl2);
            break;
         case 2:
            pyramid(5,5,gl2);
            break;

      }
   }

   //--------------------- 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
```

4. Wyniki działania:







5. Wnioski

Na podstawie otrzymanego wyniku można stwierdzić, że:

a) Biblioteka udostępnia metody do tworzenia obiektów 3D

b) Dzięki temu, że polecenia są realizowane przez sprzęt (procesor graficzny), tworzenie grafiki następuje szybciej niż innymi sposobami.