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

Laboratorium 6
1 IV 2021 r.
Temat: „Światło i materiały”
Liczba kątów:5

Robert Laszczak
Informatyka I stopień
Stacjonarne, 4 semestr
Grupa 2B

1. Polecenie

Celem jest stworzenie piramidy z użyciem różnych materiałów i umieszczenie jej na „podstawie”. Użytkownik może obracać podstawę wokół osi Y, przeciągając mysz w poziomie. Scena wykorzystuje oświetlenie. Początkowo włączone jest tylko podstawowe oświetlenie. W ramach laboratorium będziesz musiał poprawić oświetlenie.

2. Wprowadzone dane:

Liczba kątów n = 5

3. Wykorzystane komendy:

Kod źródłowy:

```
package gk;
import java.awt.*;
import javax.swing.*;
import java.awt.event.*;
import java.nio.FloatBuffer;

import com.jogamp.opengl.*;
import com.jogamp.opengl.awt.*;
import com.jogamp.opengl.fixedfunc.GLLightingFunc;
import com.jogamp.opengl.glu.GLU;
import com.jogamp.opengl.util.gl2.GLUT;
/**
 * CPSC 424, Fall 2015, Lab 6:  Light and Material in OpenGL 1.1.
 * This program shows a square "stage" with a variety of objects
 * arranged on it.  The objects use several shapes and materials
 * and include a wireframe object that is drawn with lighting
 * turned off. The user can rotate the stage about the y-axis
 * by dragging the mouse horizontally.
 */
public class SubroutineHierarchy extends GLJPanel implements GLEventListener  {
   private double rotateY = 0;  // rotation of view about the y-axis, controlled by mouse.
   /**
    * A main routine to create and show a window that contains a
    * panel of type Lab4.  The program ends when the user closes the
    * window.
    */
   public static void main(String[] args) {
      JFrame window = new JFrame("Stage");
      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 Lab6, sets the preferred size and configures event listeners
    */
   public SubroutineHierarchy() {
      super( new GLCapabilities(null) ); // Makes a panel with default OpenGL "capabilities".
      setPreferredSize( new Dimension(1000,500) );
      addGLEventListener(this); // This panel will respond to OpenGL events.
      MouseHandler mouser = new MouseHandler();
      addMouseListener(mouser);  // The MouseHandler will respond to mouse events.
   }
   // -------------------------- Data for some materials ------------------------------------
   /**
    * One of the rows of this array corresponds to a set of material properties.  Items 0 to 4 in
a row
    * specify an ambient color; items 4 through 7, a diffuse color; items 8 through 11, a specular
color;
    * and item 12, a specular exponent (shininess value).  The data is adapted from the table on
the page
    * http://devernay.free.fr/cours/opengl/materials.html
    */
   private final static float[][] materials = {
```

```
        { /* "emerald" */   0.0215f, 0.1745f, 0.0215f, 1.0f, 0.07568f, 0.61424f, 0.07568f, 1.0f,
0.633f, 0.727811f, 0.633f, 1.0f, 0.6f*128 },
        { /* "jade" */   0.135f, 0.2225f, 0.1575f, 1.0f, 0.54f, 0.89f, 0.63f, 1.0f, 0.316228f,
0.316228f, 0.316228f, 1.0f, 0.1f*128 },
        { /* "obsidian" */   0.05375f, 0.05f, 0.06625f, 1.0f, 0.18275f, 0.17f, 0.22525f, 1.0f,
0.332741f, 0.328634f, 0.346435f, 1.0f, 0.3f*128 },
        { /* "pearl" */   0.25f, 0.20725f, 0.20725f, 1.0f, 1.0f, 0.829f, 0.829f, 1.0f, 0.296648f,
0.296648f, 0.296648f, 1.0f, 0.088f*128 },
        { /* "ruby" */   0.1745f, 0.01175f, 0.01175f, 1.0f, 0.61424f, 0.04136f, 0.04136f, 1.0f,
0.727811f, 0.626959f, 0.626959f, 1.0f, 0.6f*128 },
        { /* "turquoise" */   0.1f, 0.18725f, 0.1745f, 1.0f, 0.396f, 0.74151f, 0.69102f, 1.0f,
0.297254f, 0.30829f, 0.306678f, 1.0f, 0.1f*128 },
        { /* "brass" */   0.329412f, 0.223529f, 0.027451f, 1.0f, 0.780392f, 0.568627f, 0.113725f,
1.0f, 0.992157f, 0.941176f, 0.807843f, 1.0f, 0.21794872f*128 },
        { /* "bronze" */   0.2125f, 0.1275f, 0.054f, 1.0f, 0.714f, 0.4284f, 0.18144f, 1.0f,
0.393548f, 0.271906f, 0.166721f, 1.0f, 0.2f*128 },
        { /* "chrome" */   0.25f, 0.25f, 0.25f, 1.0f, 0.4f, 0.4f, 0.4f, 1.0f, 0.774597f,
0.774597f, 0.774597f, 1.0f, 0.6f*128 },
        { /* "copper" */   0.19125f, 0.0735f, 0.0225f, 1.0f, 0.7038f, 0.27048f, 0.0828f, 1.0f,
0.256777f, 0.137622f, 0.086014f, 1.0f, 0.1f*128 },
        { /* "gold" */   0.24725f, 0.1995f, 0.0745f, 1.0f, 0.75164f, 0.60648f, 0.22648f, 1.0f,
0.628281f, 0.555802f, 0.366065f, 1.0f, 0.4f*128 },
        { /* "silver" */   0.19225f, 0.19225f, 0.19225f, 1.0f, 0.50754f, 0.50754f, 0.50754f, 1.0f,
0.508273f, 0.508273f, 0.508273f, 1.0f, 0.4f*128 },
        { /* "cyan plastic" */   0.0f, 0.1f, 0.06f, 1.0f, 0.0f, 0.50980392f, 0.50980392f, 1.0f,
0.50196078f, 0.50196078f, 0.50196078f, 1.0f, .25f*128 },
        { /* "green plastic" */   0.0f, 0.0f, 0.0f, 1.0f, 0.1f, 0.35f, 0.1f, 1.0f, 0.45f, 0.55f,
0.45f, 1.0f, .25f*128 },
        { /* "red plastic" */   0.0f, 0.0f, 0.0f, 1.0f, 0.5f, 0.0f, 0.0f, 1.0f, 0.7f, 0.6f, 0.6f,
1.0f, .25f*128 },
        { /* "cyan rubber" */   0.0f, 0.05f, 0.05f, 1.0f, 0.4f, 0.5f, 0.5f, 1.0f, 0.04f, 0.7f,
0.7f, 1.0f, .078125f*128 },
        { /* "green rubber" */   0.0f, 0.05f, 0.0f, 1.0f, 0.4f, 0.5f, 0.4f, 1.0f, 0.04f, 0.7f,
0.04f, 1.0f, .078125f*128 },
        { /* "red rubber" */   0.05f, 0.0f, 0.0f, 1.0f, 0.5f, 0.4f, 0.4f, 1.0f, 0.7f, 0.04f,
0.04f, 1.0f, .078125f*128 },
    };
    //------------------------- OpenGL methods from GLEventListener ---------------
```

```
    private GLUT glut = new GLUT(); // An object for drawing GLUT shapes.
    private GLU glu = new GLU(); // An object for calling GLU functions.
    //---Lab5
    private void PiramidaSciany(float n, GL2 gl2)
    {
        float deg = 360/n;
        float x,y;
        gl2.glBegin(GL.GL_TRIANGLE_FAN);
        gl2.glVertex3d(0,0,0);
        gl2.glNormal3f(0,0,2);
        for(float i=1;i<=n+1;i++)
        {
            x= (float) Math.cos(Math.toRadians(deg*i));
            y= (float) Math.sin(Math.toRadians(deg*i));
            gl2.glVertex3d(x, y, 2);
            gl2.glNormal3f(x,y,2);
        }
        gl2.glEnd();
    }
```

```
    private void PiramidaPodstawa(float n, GL2 gl2)
    {
        float deg = 360/n;
        double x,y;
        gl2.glBegin(GL.GL_TRIANGLE_FAN);
        gl2.glShadeModel(GLLightingFunc.GL_SMOOTH);
        gl2.glColor3f(0,0,1);
        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.glColor3f(1,1,1);
            gl2.glVertex3d(x, y, 2);
        }
        gl2.glEnd();
    }
```

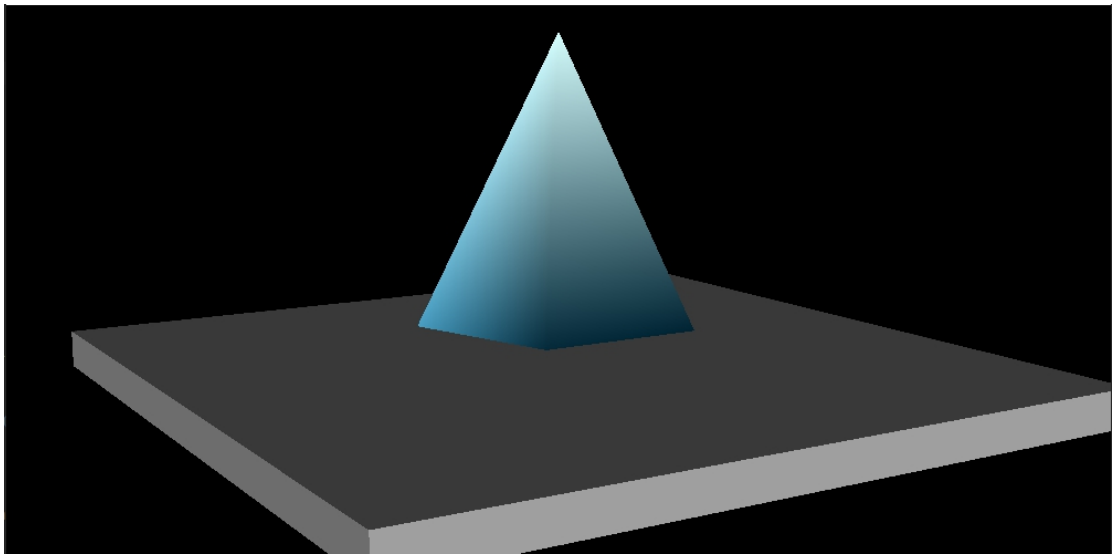
```
    private void piramida(float n, float scale, GL2 gl2)
    {
        gl2.glColor3f(1,1,1);
        gl2.glScalef(scale,scale,scale);
        gl2.glRotatef(90,1,0,0);
        gl2.glTranslatef(0,0,-2);
        PiramidaSciany(n,gl2);
        PiramidaPodstawa(n,gl2);
    }
```

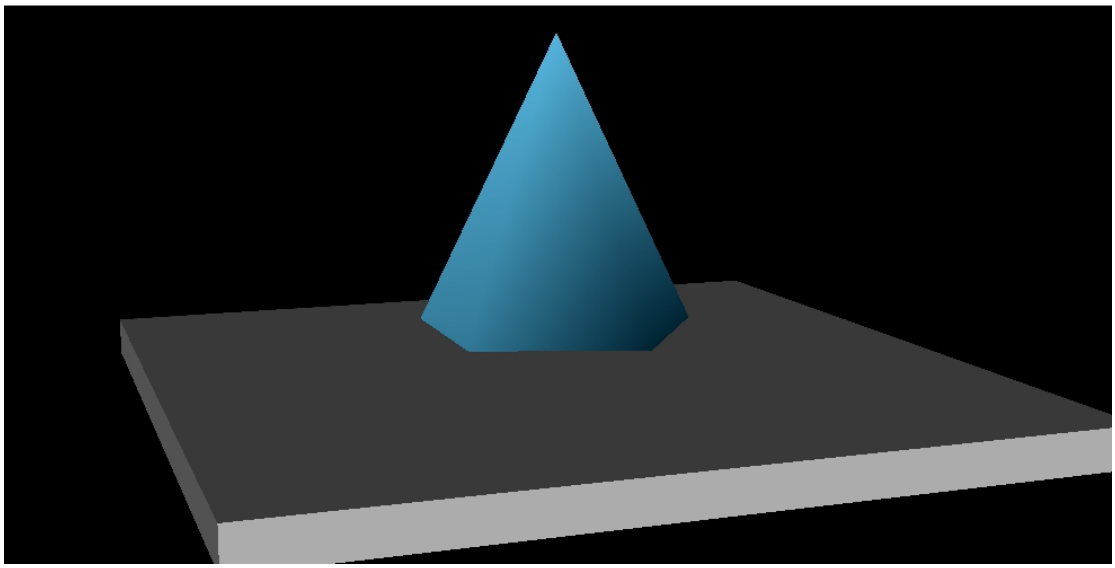
```
    /**
```

```
    * The display method is called when the panel needs to be drawn.
    * Here, it draws a stage and some objects on the stage.
    */


   public void display(GLAutoDrawable drawable) {

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

      gl2.glClear( GL2.GL_COLOR_BUFFER_BIT | GL2.GL_DEPTH_BUFFER_BIT );

      gl2.glLoadIdentity();
      glu.gluLookAt( 0,8,40, 0,1,0, 0,1,0 );  // viewing transform

      gl2.glRotated( rotateY, 0, 1, 0 );  // modeling transform: rotation of the scene about y-
axis

      float[] gray = { 0.6f, 0.6f, 0.6f, 1 };
      float[] zero = { 0, 0, 0, 1 };
      gl2.glMaterialfv(GL2.GL_FRONT_AND_BACK, GL2.GL_AMBIENT_AND_DIFFUSE, gray, 0);
      gl2.glMaterialfv(GL2.GL_FRONT_AND_BACK, GL2.GL_SPECULAR, zero, 0);
      gl2.glPushMatrix();
      gl2.glTranslated(0,-1.5,0); // Move top of stage down to y = 0
      gl2.glScaled(1, 0.05, 1); // Stage will be one unit thick,
      glut.glutSolidCube(20);
      gl2.glPopMatrix();

      // TODO draw some shapes
      float[] ambient = {0.0f,0.7f,1.0f};
      float[] diffuse = {0.0f,0.7f,1.0f};
      float[] specular = {1.0f,1.0f,1.0f};
      float[] shininess = {1.0f};
      ;        gl2.glMaterialfv(gl2.GL_FRONT_AND_BACK,GLLightingFunc.GL_AMBIENT,
FloatBuffer.wrap(ambient));
      gl2.glMaterialfv(gl2.GL_FRONT_AND_BACK,GLLightingFunc.GL_DIFFUSE, FloatBuffer.wrap(diffuse));
      gl2.glMaterialfv(gl2.GL_FRONT_AND_BACK,GLLightingFunc.GL_SPECULAR,
FloatBuffer.wrap(specular));
      gl2.glMaterialfv(gl2.GL_FRONT_AND_BACK, GLLightingFunc.GL_SHININESS,
FloatBuffer.wrap(shininess));
      piramida(5,3.7f,gl2);

   } // end display()

   /** The init method is called once, before the window is opened, to initialize
    *  OpenGL.  Here, it sets up a projection, configures some lighting, and enables
    *  the depth test.
    */
   public void init(GLAutoDrawable drawable) {
      GL2 gl2 = drawable.getGL().getGL2();
      gl2.glMatrixMode(GL2.GL_PROJECTION);
      gl2.glLoadIdentity();
      glu.gluPerspective(20, (double)getWidth()/ getHeight(), 1, 100);
      gl2.glMatrixMode(GL2.GL_MODELVIEW);
      gl2.glEnable(GL2.GL_DEPTH_TEST);
      gl2.glEnable(GL2.GL_NORMALIZE);
      gl2.glEnable(GL2.GL_LIGHTING);
      gl2.glEnable(GL2.GL_LIGHT0);
      // TODO configure better lighting!
      gl2.glLightfv(GL2.GL_LIGHT2, GL2.GL_AMBIENT, FloatBuffer.wrap(new float[] {0f,0f,0f,1}));
      gl2.glLightfv(GL2.GL_LIGHT2, GL2.GL_POSITION, FloatBuffer.wrap(new float[] {-25f,-
0.5f,0f,1}));
      gl2.glEnable(GL2.GL_LIGHT2);
   }


   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
   }

   //----------------- A class to support simple mouse interaction; ----------
   //----------------- horizontal mouse motion rotates about y-axis ----------

   private class MouseHandler extends MouseAdapter {

      private int prevX;  // Previous mouse x-coord during a drag gesture.
      private boolean dragging;  // Set to true during dragging.

      public void mouseDragged(MouseEvent evt) {
         if (dragging) {
            int x = evt.getX(); // current x coord of mouse
            double dx = x - prevX; // change in mouse coord
            rotateY += dx/7;
```

```
            repaint(); // redraw the scene
            prevX = x;
         }
      }
      public void mousePressed(MouseEvent evt) {
         if (dragging)
            return;
         prevX = evt.getX();
         dragging = true;
         SubroutineHierarchy.this.addMouseMotionListener(this);
      }
      public void mouseReleased(MouseEvent evt) {
         dragging = false;
         SubroutineHierarchy.this.removeMouseMotionListener(this);
      }

   } // end nested class MouseHandler


}
```

4. Wyniki działania:





5. Wnioski

Na podstawie otrzymanego wyniku można stwierdzić, że:

a) Biblioteka udostępnia metody sterowania oświetleniem.
b) Dzięki temu, że polecenia są realizowane przez sprzęt (procesor graficzny), tworzenie grafiki następuje szybciej niż innymi sposobami.