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

Laboratorium 7
15 IV 2021 r.
Temat: „Tekstury w OpenGL”
Wariant:
Liczba kątów:5

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

1. 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). Należy opracować metody textureFromPainting() oraz textureFromResource() klasy Lab7

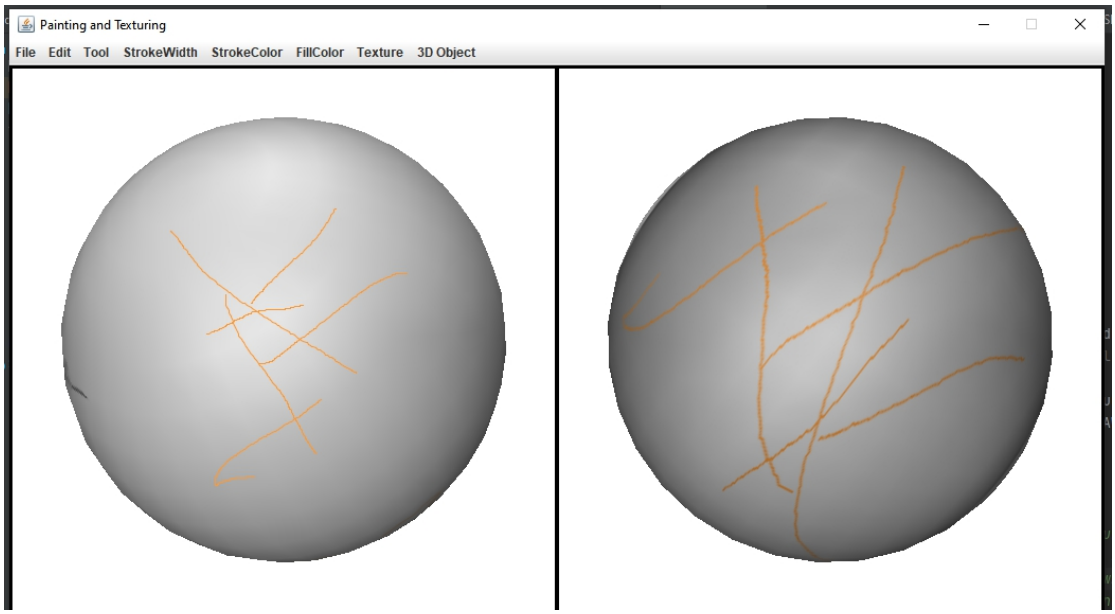
2. Wprowadzam dane:

Liczba kątów n = 5

3. Wykorzystane komendy:

Link do zdalnego repozytorium:
https://github.com/RLaszczak/Lab_GK

4. Wyniki działania:



5. Wnioski

Na podstawie otrzymanego wyniku można stwierdzić, że:

- Dzięki OpenGL możemy łatwo aplikować tekstury na bryły i powierzchnie płaskie.

Kod:

```
import java.awt.*;
import javax.imageio.ImageIO;
import javax.swing.*;
import java.awt.event.*;
import java.awt.image.BufferedImage;
import java.io.IOException;
import java.net.URL;
import java.util.Objects;

import com.jogamp.opengl.*;
import com.jogamp.opengl.awt.*;
import com.jogamp.opengl.util.awt.AWTGLReadBufferUtil;
import com.jogamp.opengl.util.awt.ImageUtil;
import com.jogamp.opengl.util.gl2.GLUT;
import com.jogamp.opengl.util.texture.Texture;
import com.jogamp.opengl.util.texture.awt.AWTTextureIO;

import static com.jogamp.opengl.GL.*;

/**
 * CPSC 424, Fall 2015, Lab 7:  Image Textures in OpenGL/JOGL.
 * When run as an application, the program shows a window with
 * two panels:  on the left, a PaintPanel where the user can
 * draw 2D images and on the right a GLJPanel that shows a single
 * object to which a texture can be applied.  There is a menu
 * for selecting the object and one for selecting the texture.
 * It is possible to apply the image from the paint panel as
 * a texture on the 3D object, and it possible to copy the
 * image from the OpenGL panel to the paint panel.
 * This program depends on PaintPanel.java, TexturedShapes.java,
 * Camera.java, and three resource files (brick.jpg, earth.jpg,
 * and clouds.jpg) that contain images used as textures.
 */
public class Lab7 extends JPanel implements GLEventListener {

    /**
     * Main routine allows this class to be run as an application.
     * It creates a JFrame, sets its content pane to be a Lab7 panel,
     * and sets its JMenuBar to be the menu bar for the panel.
     */
    public static void main(String[] args) {
        JFrame window = new JFrame("Painting and Texturing");
        Lab7 panel = new Lab7();
        window.setContentPane(panel);
        window.setJMenuBar(panel.getMenuBar());
        window.pack();
        window.setResizable(false);
        window.setLocation(50, 50);
        window.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        window.setVisible(true);
    }
```

```
    private PaintPanel paintPanel;  // A panel where the user can paint images.

    private GLJPanel displayGL;  // A GLJPanel where a shape is displayed using OpenGL.
    private GLUT glut = new GLUT();   // For drawing the teapot.

    private Camera camera;  // A Camera for setting the view/projection, which user can rotate
by mouse.

    private int currentObjectNum = 0; // tells which object to draw.

    private int textureRepeatHorizontal = 1;  // Horizontal scale factor for texture transform
    private int textureRepeatVertical = 1;    // Vertical scale factor for texture transform

    private Texture currentTexture = null;  // the texture that will be applied when the
object is drawn.

    /**
     * Constructor for class Lab7, sets the preferred size and configures event listeners
     */
    public Lab7() {
        paintPanel = new PaintPanel();
        displayGL = new GLJPanel(new GLCapabilities(null));
        paintPanel.setPreferredSize(new Dimension(512, 512));
        displayGL.setPreferredSize(new Dimension(512, 512));
        this.setLayout(new GridLayout(1, 2, 4, 4));
        this.add(paintPanel);
        this.add(displayGL);
        this.setBackground(Color.BLACK);
        this.setBorder(BorderFactory.createLineBorder(Color.BLACK, 3));
        displayGL.addGLEventListener(this); // This panel will respond to OpenGL events.
        camera = new Camera();
        camera.lookAt(1, 1, 4, 0, 0, 0, 0, 1, 0);
        camera.setLimits(-0.8, 0.8, -0.8, 0.8, -2, 2);
        camera.installTrackball(displayGL);
    }

    //------------------------- Draw a shape -----------------------------------

    /**
     * Draws the currently selected 3D object.
     */
    private void drawCurrentShape(GL2 gl2) {
        switch (currentObjectNum) {
            case 0:
                gl2.glScaled(0.9, 0.9, 0.9);
                TexturedShapes.cube(gl2);
                break;
            case 1:
                gl2.glRotated(-90, 1, 0, 0);
                gl2.glTranslated(0, 0, -0.5);
                TexturedShapes.uvCylinder(gl2);
                break;
            case 2:
                gl2.glRotated(-90, 1, 0, 0);
                gl2.glTranslated(0, 0, -0.4);
                TexturedShapes.uvCone(gl2);
                break;
            case 3:
                gl2.glScaled(1.3, 1.3, 1.3);
                TexturedShapes.uvSphere(gl2);
                break;
            case 4:
                gl2.glScaled(1.4, 1.4, 1.4);
                TexturedShapes.uvTorus(gl2);
                break;
            case 5:
                glut.glutSolidTeapot(0.47);
                break;
            case 6:
                triangularPrism(gl2);
                break;
            case 7:
                pyramid(gl2, 14);
                break;
```

```
    }
}

/**
 * Draws a triangular prism by drawing its five faces.
 */
private void triangularPrism(GL2 gl2) {
    // TODO: add texture coordinates
    double t = Math.sqrt(3) / 4;

    gl2.glBegin(GL2.GL_TRIANGLES);  // top triangular face
    gl2.glNormal3d(0, 1, 0);
    gl2.glVertex3d(-t, 0.5, -0.25);
    gl2.glVertex3d(t, 0.5, -0.25);
    gl2.glVertex3d(0, 0.5, 0.5);
    gl2.glEnd();

    gl2.glBegin(GL2.GL_TRIANGLES);  // bottom triangular
    gl2.glNormal3d(0, -1, 0);
    gl2.glVertex3d(t, -0.5, -0.25);
    gl2.glVertex3d(-t, -0.5, -0.25);
    gl2.glVertex3d(0, -0.5, 0.5);
    gl2.glEnd();

    gl2.glBegin(GL2.GL_TRIANGLE_FAN); // back face (side facing towards negative z-axis)
    gl2.glNormal3d(0, 0, -1);
    gl2.glVertex3d(-t, -0.5, -0.25);
    gl2.glVertex3d(-t, 0.5, -0.25);
    gl2.glVertex3d(t, 0.5, -0.25);
    gl2.glVertex3d(t, -0.5, -0.25);
    gl2.glEnd();

    gl2.glBegin(GL2.GL_TRIANGLE_FAN); // front left face
    gl2.glNormal3d(-0.75, 0, t);
    gl2.glVertex3d(-t, 0.5, -0.25);
    gl2.glVertex3d(-t, -0.5, -0.25);
    gl2.glVertex3d(0, -0.5, 0.5);
    gl2.glVertex3d(0, 0.5, 0.5);
    gl2.glEnd();

    gl2.glBegin(GL2.GL_TRIANGLE_FAN); // front right face
    gl2.glNormal3d(0.75, 0, t);
    gl2.glVertex3d(0, 0.5, 0.5);
    gl2.glVertex3d(0, -0.5, 0.5);
    gl2.glVertex3d(t, -0.5, -0.25);
    gl2.glVertex3d(t, 0.5, -0.25);
    gl2.glEnd();

}

//---------------------------- Implement Texture Menu Commands ---------------

/**
 * Create a JOGL Texture from an image that is given as a
 * resource file in the program.
 *
 * @param resourceName path to the resource file
 * @return the newly created texture.
 */

public void Triangle(GL2 gl2, float x1, float y1, float x2, float y2, float H) {
    gl2.glBegin(GL_TRIANGLE_FAN);

    gl2.glNormal3f(x1, 1, y1);
    gl2.glTexCoord2f(x1 + 0.5f, y1 + 0.5f);
    gl2.glVertex3f(x1, 0, y1);

    gl2.glNormal3f(x2, 1, y2);
    gl2.glTexCoord2f(x2 + 0.5f, y2 + 0.5f);
    gl2.glVertex3f(x2, 0, y2);

    gl2.glTexCoord2f(0.5f, 0.5f);
    gl2.glVertex3f(0, H, 0);
    gl2.glEnd();
```

```
}


public void pyramid(GL2 gl2, int nrOfVer) {
    float R = 0.5f;
    int n = nrOfVer;
    float[] xPoints = new float[n];
    float[] yPoints = new float[n];
    xPoints[0] = (float) (R * Math.cos((2.0 * Math.PI * 0) / n));
    yPoints[0] = (float) (R * Math.sin((2.0 * Math.PI * 0) / n));


    for (int i = 1; i < n; i++) {

        xPoints[i] = (float) (R * Math.cos((2.0 * Math.PI * i) / n));
        yPoints[i] = (float) (R * Math.sin((2.0 * Math.PI * i) / n));

        Triangle(gl2, xPoints[i - 1], yPoints[i - 1], xPoints[i], yPoints[i], 0);

        Triangle(gl2, xPoints[i - 1], yPoints[i - 1], xPoints[i], yPoints[i], 0.5f);

    }


    Triangle(gl2, xPoints[n - 1], yPoints[n - 1], xPoints[0], yPoints[0], 0);

    Triangle(gl2, xPoints[n - 1], yPoints[n - 1], xPoints[0], yPoints[0], 0.5f);


}

private Texture createContext(BufferedImage img) {
    Texture texture;

    //tworzenie kontekstu
    GLContext context = displayGL.getContext();
    boolean needsRelease = false;
    if (!context.isCurrent()) {
        context.makeCurrent();
        needsRelease = true;
    }

    GL2 gl2 = context.getGL().getGL2(); //przypisanie kontekstu

    //tworzenie tekstury
    texture = AWTTextureIO.newTexture(displayGL.getGLProfile(), img, true);

    //konfiguracja powtarzania się textury
    texture.setTexParameteri(gl2, GL2.GL_TEXTURE_WRAP_S, GL2.GL_REPEAT);
    texture.setTexParameteri(gl2, GL2.GL_TEXTURE_WRAP_T, GL2.GL_REPEAT);

    if (needsRelease) {
        context.release();
    }

    return texture;
}

private Texture textureFromResource(String resourceName) throws IOException {
    // TODO: write this method
    URL textureURL;
    textureURL = this.getClass().getClassLoader().getResource(resourceName);
    return createContext(ImageIO.read(Objects.requireNonNull(textureURL)));

}

/**
 * Create a JOGL Texture from the image in the PaintPanel.
 *
 * @return the newly created texture object.
 */
private Texture textureFromPainting() {
    // TODO: write this method
```

```
        BufferedImage img = paintPanel.copyOSC();
        return createContext(img);
    }

    /**
     * Copy the image from the OpenGL display panel to the PaintPanel.
     * (Note that this method is called from the event-handling thread.
     * To work with the OpenGL context, that context must be made current
     * on the event-handling thread (if it is not already -- and it
     * presumably won't be).  If the context has to be made current,
     * it is important to release it.)
     */
    private void paintingFromOpenGL() {
        GLContext context = displayGL.getContext(); // OpenGL context for the display panel.
        boolean needsRelease = false;  // Will be set to true if context needs to be made
current.
        if (!context.isCurrent()) {
            // Make the context current on the current thread.
            context.makeCurrent();
            needsRelease = true;
        }
        GL2 gl2 = context.getGL().getGL2();
        AWTGLReadBufferUtil readBuf = new AWTGLReadBufferUtil(displayGL.getGLProfile(), false);
        BufferedImage img = readBuf.readPixelsToBufferedImage(gl2, true); // Get display
content as image.
        if (needsRelease) {
            context.release();
        }
        paintPanel.installImage(img); // copy the image into the PaintPanel.
    }

    //------------------------- OpenGL methods from GLEventListener ---------------

    /**
     * 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);

        camera.apply(gl2); // Sets projection and view transformations.

        // TODO: apply currentTexture (or turn off texturing if it is null)
        Texture curTexture = currentTexture;
        //Włączenie wybranej textury jeżeli nie jest pusta
        if (curTexture != null) {
            curTexture.enable(gl2);
            curTexture.bind(gl2);
            drawCurrentShape(gl2);
            curTexture.disable(gl2);
        } else
            drawCurrentShape(gl2);


    } // end display()


    /**
     * The init method is called once, before the window is opened, to initialize
     * OpenGL.  Here is mostly initializes lighting and material, which will remain
     * constant for the rest of the program.  It also sets the background color
     * to white.
     */
    public void init(GLAutoDrawable drawable) {
        GL2 gl2 = drawable.getGL().getGL2();
        gl2.glClearColor(1, 1, 1, 1);
        gl2.glEnable(GL2.GL_DEPTH_TEST);
        gl2.glEnable(GL2.GL_NORMALIZE);
        gl2.glEnable(GL2.GL_LIGHTING);
        gl2.glEnable(GL2.GL_LIGHT0);
```

```
        gl2.glEnable(GL2.GL_LIGHT1);
        gl2.glEnable(GL2.GL_LIGHT2);
        gl2.glLightfv(GL2.GL_LIGHT0, GL2.GL_POSITION, new float[]{1, 1, 10, 0}, 0);
        gl2.glLightfv(GL2.GL_LIGHT1, GL2.GL_POSITION, new float[]{0, 5, 0, 0}, 0);
        gl2.glLightfv(GL2.GL_LIGHT2, GL2.GL_POSITION, new float[]{-5, -1, 10, 0}, 0);
        float[] dimmer = {0.3f, 0.3f, 0.3f, 1};
        for (int i = 0; i <= 2; i++) {
            gl2.glLightfv(GL2.GL_LIGHT0 + i, GL2.GL_DIFFUSE, dimmer, 0);
            gl2.glLightfv(GL2.GL_LIGHT0 + i, GL2.GL_SPECULAR, dimmer, 0);
        }
        gl2.glLightModeli(GL2.GL_LIGHT_MODEL_COLOR_CONTROL, GL2.GL_SEPARATE_SPECULAR_COLOR);
// Not in OpenGL 1.1
        float[] diffuse = {1, 1, 1, 1};
        float[] specular = {0.3f, 0.3f, 0.3f, 1};
        gl2.glMaterialfv(GL2.GL_FRONT_AND_BACK, GL2.GL_AMBIENT_AND_DIFFUSE, diffuse, 0);
        gl2.glMaterialfv(GL2.GL_FRONT_AND_BACK, GL2.GL_SPECULAR, specular, 0);
        gl2.glMateriali(GL2.GL_FRONT_AND_BACK, GL2.GL_SHININESS, 32);
    }


    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
    }

    // -------------------- Define a menu bar for use with this panel ----------------------

    /**
     * Creates a menu bar for use in this program.  The menu bar consists of
     * a menu bar from the PaintPanel, with two additional menus.  The menu
     * bar from PaintPanel contains commands relevant to drawing on that
     * panel.  The two additional menus let the user select the displayed
     * 3D object and texture.
     */
    public JMenuBar getMenuBar() {
        JMenuBar menuBar = paintPanel.getMenuBar();

        JMenu textureMenu = new JMenu("Texture");
        ActionListener textureListener = new ActionListener() {
            public void actionPerformed(ActionEvent e) {
                int itemNum = Integer.parseInt(e.getActionCommand());
                switch (itemNum) {
                    case 0:
                        currentTexture = textureFromPainting();
                        break;
                    case 1:
                        paintingFromOpenGL();
                        break;
                    case 2:
                        currentTexture = null;
                        break;
                    case 3:
                        try {
                            currentTexture = textureFromResource("earth.jpg");
                        } catch (IOException ex) {
                            ex.printStackTrace();
                        }
                        break;
                    case 4:
                        try {
                            currentTexture = textureFromResource("brick.jpg");
                        } catch (IOException ex) {
                            ex.printStackTrace();
                        }
                        break;
                    case 5:
                        try {
                            currentTexture = textureFromResource("clouds.jpg");
                        } catch (IOException ex) {
                            ex.printStackTrace();
```

```
                        }
                        break;
                    default:
                        if (itemNum < 10)
                            textureRepeatHorizontal = itemNum - 5;
                        else
                            textureRepeatVertical = itemNum - 9;
                }
                if (itemNum != 1) {
                    displayGL.repaint();
                }
            }
        };
        makeMenuItem(textureMenu, ">>> Texture From Painting >>>", textureListener, 0);
        makeMenuItem(textureMenu, "<<< Painting From OpenGL <<<", textureListener, 1);
        textureMenu.addSeparator();
        makeMenuItem(textureMenu, "No Texture", textureListener, 2);
        makeMenuItem(textureMenu, "Earth Texture", textureListener, 3);
        makeMenuItem(textureMenu, "Brick Texture", textureListener, 4);
        makeMenuItem(textureMenu, "Clouds Texture", textureListener, 5);
        textureMenu.addSeparator();
        makeMenuItem(textureMenu, "Horizontal Repeat = 1", textureListener, 6);
        makeMenuItem(textureMenu, "Horizontal Repeat = 2", textureListener, 7);
        makeMenuItem(textureMenu, "Horizontal Repeat = 3", textureListener, 8);
        makeMenuItem(textureMenu, "Horizontal Repeat = 4", textureListener, 9);
        textureMenu.addSeparator();
        makeMenuItem(textureMenu, "Vertical Repeat = 1", textureListener, 10);
        makeMenuItem(textureMenu, "Vertical Repeat = 2", textureListener, 11);
        makeMenuItem(textureMenu, "Vertical Repeat = 3", textureListener, 12);
        makeMenuItem(textureMenu, "Vertical Repeat = 4", textureListener, 13);
        menuBar.add(textureMenu);

        JMenu objectMenu = new JMenu("3D Object");
        ActionListener objectListener = new ActionListener() {
            public void actionPerformed(ActionEvent e) {
                currentObjectNum = Integer.parseInt(e.getActionCommand());
                displayGL.repaint();
            }
        };
        makeMenuItem(objectMenu, "Cube", objectListener, 0);
        makeMenuItem(objectMenu, "Cylinder", objectListener, 1);
        makeMenuItem(objectMenu, "Cone", objectListener, 2);
        makeMenuItem(objectMenu, "Sphere", objectListener, 3);
        makeMenuItem(objectMenu, "Torus", objectListener, 4);
        makeMenuItem(objectMenu, "Teapot", objectListener, 5);
        makeMenuItem(objectMenu, "Triangular Prism", objectListener, 6);
        makeMenuItem(objectMenu, "Pyramid", objectListener, 7);
        menuBar.add(objectMenu);

        return menuBar;
    }

    /**
     * Create a menu item and add it to a menu.
     *
     * @param menu     the menu to which the item will be added
     * @param name     the text for the item
     * @param listener an ActionListener that will handle commands from the item
     * @param i        the action command for the item is set to ("" + i)
     */
    private void makeMenuItem(JMenu menu, String name, ActionListener listener, int i) {
        JMenuItem item = new JMenuItem(name);
        item.addActionListener(listener);
        item.setActionCommand("" + i);
        menu.add(item);
    }

} // end class Lab7
```