Magenta

10

Good job.

```
/* ----------------------------------------------------------------------
 *
 * Author -------------- Cecilia Y. Sui
 * Assignment ---------- Lighted Hexagonal Bipyramid
 * Course -------------- Computer Graphics
 * Instructor ---------- Dr. Crawley
 * Date of Submission -- October 18, 2019
 * Language Used ------- Java & OpenGL
 * Class Imported ------ Camera from Camera.java (written by Dr. Eck)
 * Description --------- Use JOGL to draw a lighted hexagon bipyramid with
 * --------------------- an indexed face set (IFS) and 1+ light sources
 *
--------------------------------------------------------------------------- */

// ------------------------------------------------------------------------
// Imports
// ------------------------------------------------------------------------
import java.awt.*;
import java.awt.event.*;
import javax.swing.*;
import com.jogamp.opengl.*;
import com.jogamp.opengl.awt.*;

// ------------------------------------------------------------------------
// LightedBipyramid class
// ------------------------------------------------------------------------
public class LightedBipyramid extends GLJPanel implements GLEventListener {
    public static void main(String[] args) {
        JFrame window = new JFrame("Lighted Hexagonal Bipyramid");
        LightedBipyramid panel = new LightedBipyramid();
        window.setContentPane(panel);
        window.pack();
        window.setLocation(50,50);
        window.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        window.setVisible(true);
        panel.requestFocusInWindow();
    }

    // --------------------------------------------------------------------
    // Constructor
    // --------------------------------------------------------------------
    public LightedBipyramid() {
        setPreferredSize( new Dimension(800,800) );
        addGLEventListener(this);
    }

    // --------------------------------------------------------------------
    // Declare private variables
    // --------------------------------------------------------------------
    private Camera camera;
    private JRadioButtonMenuItem coloredFaces;

    // --------------------------------------------------------------------
// display method
// ---------------------------------------------------------------------------
public void display(GLAutoDrawable drawable) {
    GL2 gl2 = drawable.getGL().getGL2();
    gl2.glClearColor(0,0,0,0);
    gl2.glClear( GL2.GL_COLOR_BUFFER_BIT | GL2.GL_DEPTH_BUFFER_BIT );

    // ---------------------------------------------------------------------------
    // Set vertices, faces, colors for hexagonal bipyramid
    // ---------------------------------------------------------------------------
    double[][] vertexList = {{0,0,4},{0,0,-4},{-2,0,0},{-1,1.73,0},{1,1.73,0},{2,0,0},{1
    ,-1.73,0},{-1,-1.73,0}};
    int[][] faceList = {{0,3,2},{0,4,3},{0,5,4},{0,6,5},{0,7,6},{0,2,7},{1,2,3},{1,3,4
    },{1,4,5},{1,5,6},{1,6,7},{1,7,2}};
    camera.apply(gl2);

    // ---------------------------------------------------------------------------
    // Specify Material Chosen
    // ---------------------------------------------------------------------------

    float amb[] = {0.2f,0,0,1.0f};
    float dif[] = {0,0,0.4f,1.0f};
    float spe[] = {0,0.8f,0,1.0f};

    gl2.glMaterialfv(GL2.GL_FRONT_AND_BACK, GL2.GL_AMBIENT, amb,0);
    gl2.glMaterialfv(GL2.GL_FRONT_AND_BACK, GL2.GL_DIFFUSE, dif,0);
    gl2.glMaterialfv(GL2.GL_FRONT_AND_BACK, GL2.GL_SPECULAR, spe,0);
    gl2.glMaterialf(GL2.GL_FRONT_AND_BACK, GL2.GL_SHININESS, 128.0f);  // controls size
    of specular point

    gl2.glPushMatrix();
    int i,j;
    gl2.glColor3d(1,1,1);
    double Vx, Vy, Vz, Wx, Wy, Wz, Nx, Ny, Nz, x[], y[], z[];
    for (i = 0; i < faceList.length; i++) {
        gl2.glBegin(GL2.GL_TRIANGLE_FAN);
        // Compute Normal Vectors
        ------------------------------------------------------------
        // get coordinates of 3 vertices
        x = vertexList[faceList[i][0]];
        y = vertexList[faceList[i][1]];
        z = vertexList[faceList[i][2]];
        // Convert bond vectors to free vectors (centered at origin)
        Vx = y[0] - x[0];
        Vy = y[1] - x[1];
        Vz = y[2] - x[2];
        Wx = z[0] - x[0];
        Wy = z[1] - x[1];
        Wz = z[2] - x[2];
        // Compute cross product of V & W
        Nx = Vy * Wz - Vz * Wy;
        Ny = Vz * Wx - Vx * Wz;
        Nz = Vx * Wy - Vy * Wx;
```

```
            // Declare Normal Vector for each face
            gl2.glNormal3d(Nx, Ny, Nz);
            for (j = 0; j < faceList[i].length; j++) {
                int vertexNum = faceList[i][j];
                gl2.glVertex3dv(vertexList[vertexNum], 0);
            }

            gl2.glEnd();
        }
        gl2.glPopMatrix();
    }

    // ---------------------------------------------------------------------------
    // init method
    // ---------------------------------------------------------------------------
    public void init(GLAutoDrawable graphics) {
        GL2 gl2 = graphics.getGL().getGL2();
        gl2.glClearColor(0, 0, 0, 1);
        gl2.glEnable(GL2.GL_DEPTH_TEST);
        gl2.glEnable(GL2.GL_LIGHTING);
        gl2.glEnable(GL2.GL_NORMALIZE);
        gl2.glEnable(GL2.GL_RESCALE_NORMAL);
        gl2.glEnable(GL2.GL_COLOR_MATERIAL);
        // Disable smoothing to get flat surfaces (sharp edeges)
        gl2.glShadeModel(GL2.GL_FLAT);
        gl2.glPolygonOffset(1,2);
        camera = new Camera();
        camera.lookAt(4,15,6, 0,0,0, 0,1,0);
        camera.installTrackball(this);

        // Light source: LIGHT0
        gl2.glEnable(GL2.GL_LIGHT0);
        float diffuse[] = {0f, 0f, 0.6f, 1.0f};
        float specular[] = {0f, 0.8f, 0f, 1.0f};
        float ambient[] = {0.4f, 0f, 0f, 1.0f};
        gl2.glLightfv(GL2.GL_LIGHT0, GL2.GL_DIFFUSE, diffuse, 0);
        gl2.glLightfv(GL2.GL_LIGHT0, GL2.GL_SPECULAR, specular, 0);
        gl2.glLightfv(GL2.GL_LIGHT0, GL2.GL_AMBIENT, ambient, 0);

    }


    public void dispose(GLAutoDrawable graphics) {
    }
    public void reshape(GLAutoDrawable graphics, int x, int y, int width, int height) {
    }

}
```

```
#-------------------------------------------------------------------------------------
# Author -------------- Cecilia Y. Sui
# Course -------------- Computer Graphics
# Instructor ---------- Dr. Crawley
# Submission Date ----- September 11, 2019
# Language Used ------- Python 3
# Program Description - This program implements the 3 basic transformations:
# -------------------- translation, scaling, rotation.
#-------------------------------------------------------------------------------------
```

9

1 ✓

2 — Input restrictions

Column/row major

```
#-------------------------------------------------------------------------------------
# imports
#-------------------------------------------------------------------------------------
import math

#-------------------------------------------------------------------------------------
# Function Definitions
#-------------------------------------------------------------------------------------

# Matrix Multiplication ---------------------------------------------------------------
# A is 3 by 3
# B is 3 by 1
def MatrixMult(A, B):
    if len(A) != 3:
        print("Wrong dimension for transformation matrix")
    elif len(B) != 3:
        print("Wrong dimension for coordinate matrix")
    else:
        result = [0 for i in range(3)]
        for i in range(3):
            for j in range(3):
                result[i] += A[i][j]*B[j]
        return result


# Translation -------------------------------------------------------------------------
def translate(B):
    # B is the original coordinates
    B.append(1)

    # populate identity matrix
    I = [[0 for i in range(3)] for j in range(3)]
    I[0][0], I[1][1], I[2][2] = 1,1,1

    # ask user for translate parameters (e,f)
    I[0][2] = int(input("Translation factor for x: "))
    I[1][2] = int(input("Translation factor for y: "))
```

float

This function should do no I/O. Likewise for the other transforms.

```
    # matrix multiplicaiton
    result = MatrixMult(I, B)
    return round(result[0],2), round(result[1],2)


# Scaling ----------------------------------------------------------------------------
def scale(B):
    # B is the original coordinates
    B.append(1)

    # populate scaling matrix & ask user for scaling factors (a,b)
    I = [[0 for i in range(3)] for j in range(3)]
    I[2][2] = 1
    I[0][0] = int(input("Scaling factor for x: "))
    I[1][1] = int(input("Scaling factor for y: "))

    # matrix multiplication
    result = MatrixMult(I, B)
    return round(result[0],2), round(result[1],2)


# Rotation ----------------------------------------------------------------------
def rotate(B):
    # B is the original coordinates
    B.append(1)

    # ask user for rotation angle r
    R = input("Rotation angle in degrees (r): ").split()
```

Why are you splitting?

```
    R = [int(i) for i in R]
    r = R[0] * math.pi / 180

    # populate scaling matrix
    I = [[0 for i in range(3)] for j in range(3)]
    I[0][0], I[1][1] = math.cos(r), math.cos(r)
    I[0][1], I[1][0] = -math.sin(r), math.sin(r)
```

Don't compute the same trig function more than once.

```
    # matrix multiplication
    result = MatrixMult(I, B)
    return round(result[0],2), round(result[1],2)

#-------------------------------------------------------------------------------
# Main Function
#-------------------------------------------------------------------------------
```

Why restrict the coordinates to ints?

```
def main():
    x = int(input("x coordinate: "))
    y = int(input("y coordinate: "))
    Coor = [x,y]
    user = input("Choose transformation: \nEnter T for translation, S for scaling, R for
    rotation: ").strip().upper()[0]
    #
    if user == "T":
        print("New point at:", translate(Coor))
    elif user == "S":
        print("New point at:", scale(Coor))
    elif user == "R":
        print("New point at:", rotate(Coor))
    else:
        print("Invalid Input.")

if __name__ == "__main__":
    main()
```

Keep I/O & computation separate.

10

```
//--------------------------------------------------------------------
// Author -------------- Cecilia Y. Sui
// Course -------------- Computer Graphics
// Instructor ---------- Dr. Crawley
// Date of Submission -- September 20, 2019
// Assignment ---------- Use Java Graphics2D to draw a house with
// --------------------- roof, door, windows, trees or shrub, sun or
// --------------------- moon.
//--------------------------------------------------------------------

//--------------------------------------------------------------------
// Import
//--------------------------------------------------------------------
import java.awt.*;
import java.awt.geom.*;
import javax.swing.*;
import java.awt.image.BufferedImage;
import java.io.File;
import javax.imageio.ImageIO;
import java.io.IOException;



//--------------------------------------------------------------------
// House Class extends JPanel
//--------------------------------------------------------------------
public class House extends JPanel{
    public static void main (String[] args) throws IOException{
        JFrame window;
        window = new JFrame("The Happy House");
        window.setContentPane(new House());
        window.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        window.pack();
        window.setResizable(false);
        Dimension screen = Toolkit.getDefaultToolkit().getScreenSize(
);
        window.setLocation(
            (screen.width - window.getWidth())/2,
            (screen.height - window.getHeight())/2);
```

Cecilia Sui

```
        window.setVisible(true);
    }

    private float pixelSize;

    //--------------------------------------------------------------
    // Constructor
    //--------------------------------------------------------------
    public House(){
        setPreferredSize( new Dimension(1200,700));
    }

    //--------------------------------------------------------------
    // paintComponent Function
    //--------------------------------------------------------------
    protected void paintComponent(Graphics g) {

        Graphics2D g2 = (Graphics2D)g.create();
        g2.setRenderingHint(RenderingHints.KEY_ANTIALIASING, Renderin
gHints.VALUE_ANTIALIAS_ON);
        g2.setPaint(Color.WHITE);
        g2.fillRect(0,0,getWidth(),getHeight());
        applyapplyWindowToViewportTransformation(g2, -100, 100, -100,
 100, true);

        //----------------------------------------------------------
        // Draw the chimney
        //----------------------------------------------------------
        Rectangle2D chim = new Rectangle2D.Double(-120,25,12,30);
        g2.setPaint(new Color(128,64,0,255));
        g2.fill(chim);
        g2.setStroke(new BasicStroke(4*pixelSize));
        g2.setPaint(new Color(190,188,193));
        g2.draw(chim);

        //----------------------------------------------------------
        // Draw the smoke from chimney
        //----------------------------------------------------------
```

Cecilia Sui

```
AffineTransform savedChim = g2.getTransform();
Ellipse2D smoke = new Ellipse2D.Double(-120,58,10,8);
g2.setPaint(new Color(190,188,193));
g2.fill(smoke);
g2.scale(1.1,1.1);
g2.translate(-1,1);
g2.fill(smoke);
g2.setTransform(savedChim);
g2.scale(1.5,1.5);
g2.translate(19,-9);
g2.fill(smoke);
g2.setTransform(savedChim);

//-------------------------------------------------------------
// Draw Triangle House Roof
//-------------------------------------------------------------
Path2D p = new Path2D.Double();
p.moveTo(-150,0);
p.lineTo(-60,80);
p.lineTo(30,0);
p.closePath();
g2.setPaint(new Color(190,188,193));
g2.fill(p);

//-------------------------------------------------------------
// Print the image "Happy" on house roof
//-------------------------------------------------------------
BufferedImage img = null;
try {
    img = ImageIO.read(new File("Happy.png"));
}
catch (IOException e) {
}
g2.drawImage(img, -80,10,48,44,null);

//-------------------------------------------------------------
// Draw the House body rectangle
//-------------------------------------------------------------
```

Cecilia Sui

```
g2.setPaint(new Color(172,229,238));
g2.fill( new Rectangle2D.Double(-150,-80,180,80) );

//-----------------------------------------------------------
// Draw the door in the middle
//-----------------------------------------------------------
g2.setPaint(new Color(33,46,83,200));
g2.fill(new Rectangle2D.Double(-80,-80,40,35));
// Door knobs   Details!
g2.setPaint(Color.pink);
g2.fill(new Ellipse2D.Double(-58,-65,2,2));
g2.fill(new Ellipse2D.Double(-64,-65,2,2));
// Door Line
g2.setStroke(new BasicStroke(2*pixelSize));
g2.draw(new Line2D.Double(-60,-80,-60,-45));
// Filled arc on top
Path2D p2 = new Path2D.Double();
p2.moveTo(-80,-45);
p2.quadTo(-60,-20,-40,-45);
p2.closePath();
g2.setPaint(Color.pink);
g2.fill(p2);

//-----------------------------------------------------------
// Draw the concrete floor
//-----------------------------------------------------------
g2.setPaint(new Color(190,188,193));
g2.setStroke(new BasicStroke(20*pixelSize));
g2.draw(new Line2D.Double(-155,-82,35,-82));

//-----------------------------------------------------------
// Draw the windows
//-----------------------------------------------------------
Rectangle2D wind = new Rectangle2D.Double(-135,-45,30,30);
g2.setPaint(new Color(33,46,83,200));
g2.fill(wind);
AffineTransform savedWind = g2.getTransform();
g2.translate(120,0);
```

Cecilia Sui

```
g2.fill(wind);
g2.setTransform(savedWind);
g2.setPaint(Color.pink);
g2.setStroke(new BasicStroke(4*pixelSize));
g2.draw(new Line2D.Double(-120,-45,-120,-15));
g2.draw(new Line2D.Double(-135,-30,-105,-30));
g2.draw(new Line2D.Double(0,-45,0,-15));
g2.draw(new Line2D.Double(-15,-30,15,-30));
g2.setStroke(new BasicStroke(10*pixelSize));
g2.draw(new Line2D.Double(-135,-45,-105,-45));
g2.draw(new Line2D.Double(-15,-45,15,-45));

//-----------------------------------------------------------
// Draw the sun
//-----------------------------------------------------------
Ellipse2D sun = new Ellipse2D.Double(100,40,32,32);
g2.setPaint(new Color(255,247,0,220));
g2.fill(sun);
Rectangle2D light = new Rectangle2D.Double(116,56,11,3.5);
g2.setStroke( new BasicStroke(2*pixelSize) );
for (int i = 0; i < 10; i++) {
    AffineTransform savedTransform = g2.getTransform();
    double angle = (2*Math.PI/10) * i;
    g2.rotate(angle, 116,56);
    g2.translate(20,0);
    g2.setPaint( new Color(255,255,51) );
    g2.fill(light);
    g2.setPaint(Color.yellow);
    g2.draw(light);
    g2.setTransform(savedTransform);
}

//-----------------------------------------------------------
// Draw the Trees
//-----------------------------------------------------------
Rectangle2D trunk = new Rectangle2D.Double(57,-80,6,68);
g2.setPaint(new Color(101,67,33));
g2.fill(trunk);
```

Cecilia Sui

```
        Ellipse2D tree = new Ellipse2D.Double(40,-20,20,15);
        g2.setPaint(new Color(144,151,0));
        for (int i = 0; i < 8; i++){
            AffineTransform savedTree = g2.getTransform();
            double angle2 = (2*Math.PI/8) * i;
            g2.rotate(angle2, 60,-15);
            g2.fill(tree);
            g2.setTransform(savedTree);
        }
        Rectangle2D trunk2 = new Rectangle2D.Double(103,-80,4,51);
        g2.setPaint(new Color(101,67,33));
        g2.fill(trunk2);
        Ellipse2D tree2 = new Ellipse2D.Double(90,-40,15,10.5);
        g2.setPaint(new Color(68,75,9));
        for (int i = 0; i < 8; i++){
            AffineTransform savedTree = g2.getTransform();
            double angle2 = (2*Math.PI/8) * i;
            g2.rotate(angle2, 105,-35);
            g2.translate(0,0);
            g2.fill(tree2);
            g2.setTransform(savedTree);
        }

        //----------------------------------------------------------
        // Print Welcome Note
        //----------------------------------------------------------
        g.setColor(new Color(140,190,214));
        g.setFont(new Font("Courier", Font.PLAIN, 25));
        g.drawString("Welcome to The Happy House!", 195, 50);

    }

    //--------------------------------------------------------------
    // applyapplyWindowToViewportTransformation
    //--------------------------------------------------------------
    private void applyapplyWindowToViewportTransformation(Graphics2D
g2,
            double left, double right, double bottom, double top,
```

Cecilia Sui

```
            boolean preserveAspect) {
        int width = getWidth();
        int height = getHeight();
        if (preserveAspect) {
            double displayAspect = Math.abs((double)height / width);
            double requestedAspect = Math.abs(( bottom-
top ) / ( right-left ));
            if (displayAspect > requestedAspect) {
                double excess = (bottom-top) * (displayAspect/
requestedAspect - 1);
                bottom += excess/2;
                top -= excess/2;
            }
            else if (displayAspect < requestedAspect) {
                double excess = (right-left) * (requestedAspect/
displayAspect - 1);
                right += excess/2;
                left -= excess/2;
            }
        }
        g2.scale( width / (right-left), height / (bottom-top) );
        g2.translate( -left, -top );
        double pixelWidth = Math.abs(( right - left ) / width);
        double pixelHeight = Math.abs(( bottom - top ) / height);
        pixelSize = (float)Math.max(pixelWidth,pixelHeight);
    }
}
```

Name: Cecilia Y. Sui

Examination #1

CS 3233 — October 2, 2019

Cecilia Sui

Part I (50 points)

Total Score: 100

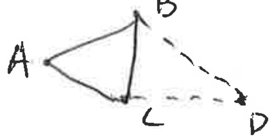
1. (8 points)
Give two reasons for using triangles as the fundamental shape units for building more complex shapes. polygon.

vertices .

a. Triangles are the simplest planar object, i.e. it requires at least 3 points to define a plane, The vertices are always in the same plane. smallest polygon that defines a plane .

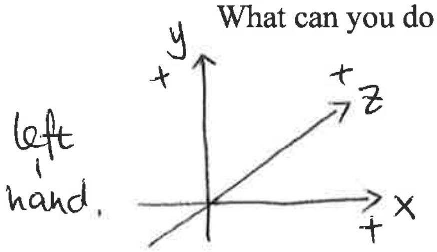
b. After initializing the first triangle, every extra point defines a new triangle, which saves a lot of memory.

memory efficiency (data size).



- any polygons of higher degree can be formed as a collection of triangles .

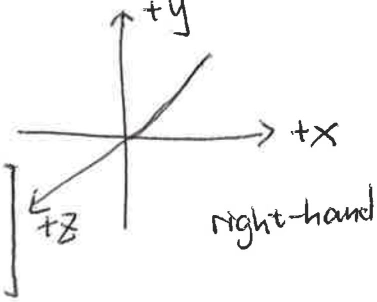
2. (8 points)
Explain what it means to say that by default 3D OpenGL uses a left-hand coordinate system? What can you do to transform it to a right-hand coordinate system?



By default, OpenGL uses a left-hand coordinate system like the drawing on the left, where the ~~z~~ positive z-axis goes into the page.

We can scale the z-axis by -1 eg. scale(1, 1, -1) to transform to a righ-hand coordinate system.

3. (10 points)
Show the transform matrix for each of these two-dimensional transforms:



a. rotate(25°)

$$R = \begin{bmatrix} \cos(\text{Math.toRadians}(25)) & -\sin(\text{Math.toRadians}(25)) & 0 \\ \sin(\text{Math.toRadians}(25)) & \cos(\text{Math.toRadians}(25)) & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

b. translate(10,15)

$$T = \begin{bmatrix} 1 & 0 & 10 \\ 0 & 1 & 15 \\ 0 & 0 & 1 \end{bmatrix}$$

26

Cecilia Sui

4. (8 points)
Show how to use a matrix/vector multiplication to scale a 2D vertex (3,4) by a factor of 5 on both the *x* and *y* axes.

$$\text{Scale}: \begin{bmatrix} 5 & 0 & 0 \\ 0 & 5 & 0 \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} 3 \\ 4 \\ 1 \end{bmatrix} = \cancel{15}\begin{bmatrix} 3\times5 \\ 4\times5 \\ 1\times1 \end{bmatrix} = \begin{bmatrix} 15 \\ 20 \\ 1 \end{bmatrix}$$

Thus, vertex(3,4) is transformed to vertex (15, 20).

alpha values : eg. α=20% → Front 20% + Back · 80% → overlap.

5. (8 points)
Give a brief explanation of RGBA color.

RGBA color uses 32 bits to represent colors for pixels. R represents red; G is green; B is blue; A is alpha, which gives the transparency. The range of each ~~values~~ goes from 0 to 255, since each color ~~section~~ has 8 bits in total. eg. (1,0,0) gives the color red. The parameters can take values from 0 to 1 depending on the method used to specify the colors. Alphas allows the color to be completely opaque with a value of 1, or ~~so~~ transparent to some extent with values below 1.

Mercy here. You've switched from 0-255 to 0-1 without explanation

the floating point numbers don't actually provide more values, since each color still has 8-bits.

6. (8 points)
How do you specify the axis of rotation for a 3D rotation in OpenGL?

The axis of rotation for 3D is specified by giving only one vertex, and the line defined by that vertex and the origin (0,0,0) is used as the axis of rotation. The positive direction of rotation can also be obtained using the right-hand rule.

Part II (50 points)

+50

box comment.

This part of the exam is a take-home programming assignment. Submit your solution to Canvas by the designated deadline.

Camera Class.

Write a Java program that uses OpenGL to draw a *hexagonal bipyramid.* Use a different color for each face of the figure. Give it a slight rotation so the viewer can see the shape clearly. Use an indexed face set (IFS) as discussed in Section 3.4.1 of the textbook to draw the figure.

50 + 24

50/50

```
/* -------------------------------------------------------------------------
 * Author -------------- Cecilia Y. Sui
 * Assignment ---------- Exam #1 Part II
 * Course -------------- Computer Graphics
 * Instructor ---------- Dr. Crawley
 * Date of Submission -- October 5, 2019
 * Language Used ------- Java & OpenGL
 * Class Imported ------ Camera from Camera.java
 * Description --------- Use JOGL to draw a hexagon bipyramid with
 * --------------------- an indexed face set (IFS)
---------------------------------------------------------------------------- */

// -------------------------------------------------------------------------
// Imports
// -------------------------------------------------------------------------
import java.awt.*;
import java.awt.event.*;
import javax.swing.*;
import com.jogamp.opengl.*;
import com.jogamp.opengl.awt.*;

// -------------------------------------------------------------------------
// Bipyramid class
// -------------------------------------------------------------------------
public class Bipyramid extends GLJPanel implements GLEventListener {
    public static void main(String[] args) {
        JFrame window = new JFrame("Hexagonal Bipyramid -- Rotate With Mouse! ");
        Bipyramid panel = new Bipyramid();
        window.setContentPane(panel);
        window.setJMenuBar(panel.createMenuBar());
        window.pack();
        window.setLocation(50,50);
        window.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        window.setVisible(true);
        panel.requestFocusInWindow();
    }

    // ---------------------------------------------------------------------
    // Constructor
// ---------------------------------------------------------------------------
public Bipyramid() {
    setPreferredSize( new Dimension(800,800) );
    addGLEventListener(this);
}

// -------Private-------------------------------------------------------------
// Declare pricate variables
// ---------------------------------------------------------------------------
private Camera camera;
private JRadioButtonMenuItem orthographic, drawEdges, drawFaces, drawBoth, coloredFaces;


// ---------------------------------------------------------------------------
// display method
// ---------------------------------------------------------------------------
public void display(GLAutoDrawable drawable) {
    GL2 gl2 = drawable.getGL().getGL2();
    gl2.glClear( GL2.GL_COLOR_BUFFER_BIT | GL2.GL_DEPTH_BUFFER_BIT );

    // ---------------------------------------------------------------------------
    // Set vertices, faces, colors for hexagonal bipyramid
    // ---------------------------------------------------------------------------
    double[][] vertexList = {{0,0,4},{0,0,-4},{-2,0,0},{-1,1.73,0},{1,1.73,0},{2,0,0},{1,-1.73,0},{-1,-1.73,0}};
    int[][] faceList = {{0,2,3},{0,3,4},{0,4,5},{0,5,6},{0,6,7},{0,7,2},{1,2,3},{1,3,4},{1,4,5},{1,5,6},{1,6,7},{1,7,2
    }};
    double[][] faceColors = {{0.93,0.44,0.39},{0.36,0.63,0.89},{0.96,0.82,0.25},
                             {0.1,0.74,0.61},{0.63,0.82,0.91},{0.95,0.61,0.07},
                             {0.96,0.82,0.25},{0.1,0.74,0.61},{0.63,0.82,0.91},
                             {0.95,0.61,0.07},{0.93,0.44,0.39},{0.36,0.63,0.89}};

    // ---------------------------------------------------------------------------
    // Control Render Options: faces | edges | both
    // ---------------------------------------------------------------------------
    camera.apply(gl2);
    gl2.glPushMatrix();
    boolean colored = coloredFaces.isSelected();
    int i,j;
    if (drawFaces.isSelected() || drawBoth.isSelected()) {
            if (drawBoth.isSelected()) {
                gl2.glEnable(GL2.GL_POLYGON_OFFSET_FILL);
            }
            for (i = 0; i < faceList.length; i++) {
                if (colored) {
                    gl2.glColor3dv(faceColors[i], 0 );
                }
                gl2.glBegin(GL2.GL_TRIANGLE_FAN);
                for (j = 0; j < faceList[i].length; j++) {
                    int vertexNum = faceList[i][j];
                    gl2.glVertex3dv( vertexList[vertexNum], 0 );
                }
                gl2.glEnd();
            }
            gl2.glDisable(GL2.GL_POLYGON_OFFSET_FILL); // when not selected
        }
        if (drawEdges.isSelected() || drawBoth.isSelected()) {
            if (drawBoth.isSelected()) {
                gl2.glColor3f(0,0,0);
            }
            else {
                gl2.glColor3f(1,1,1);
            }
            for (i = 0; i < faceList.length; i++) {
                gl2.glBegin(GL2.GL_LINE_LOOP);
                for (j = 0; j < faceList[i].length; j++) {
                    int vertexNum = faceList[i][j];
                    gl2.glVertex3dv( vertexList[vertexNum], 0 );
                }
                gl2.glEnd();
            }
        }
        gl2.glPopMatrix();
    }

    // ----------------------------------------------------------------------------
    // init method
    // ----------------------------------------------------------------------------
    public void init(GLAutoDrawable graphics) {
        GL2 gl2 = graphics.getGL().getGL2();
        gl2.glEnable(GL2.GL_DEPTH_TEST);
        gl2.glLineWidth(2);
        gl2.glPolygonOffset(1,2);
        camera = new Camera();
        // Camera class is used exactly as the author created with no modification.
        // It is downloaded from the website of the textbook.
        camera.lookAt(4,15,6, 0,0,0, 0,1,0);
        camera.installTrackball(this);
    }

    public void dispose(GLAutoDrawable graphics) {
    }
    public void reshape(GLAutoDrawable graphics, int x, int y, int width, int height) {
    }

    // -----------------------------------------------------------------------------
    // create the Menu Bar for Render Options
    // -----------------------------------------------------------------------------
    private JMenuBar createMenuBar() {
        JMenuBar menuBar = new JMenuBar();
        JMenu render = new JMenu("Render Options");
        menuBar.add(render);

        // -----------------------------------------------------------------------------
        // repaint() method for repainting
        // -----------------------------------------------------------------------------
        ActionListener repainter = new ActionListener() {
            public void actionPerformed(ActionEvent evt) {
                repaint();
            }
        };

        // -----------------------------------------------------------------------------
        // Menu Botton Control:
        // -----------------------------------------------------------------------------
        JRadioButtonMenuItem[] items;
        items = createRadioMenuGroup(new String[] {"Colored Faces", "WhiteFaces"}, render, repainter);
        coloredFaces = items[0];
        coloredFaces.setSelected(true);
        render.addSeparator();
        items = createRadioMenuGroup(new String[] {"Draw Faces Only", "Draw Edges Only", "Draw Both"},
                                                  render, repainter);
        drawFaces = items[0];
        drawEdges = items[1];
        drawBoth = items[2];
        drawBoth.setSelected(true);
        render.addSeparator();
        items = createRadioMenuGroup(new String[] {"Perspective Projetion", "Orthographics Projection"}, render, repainter
        );
        orthographic = items[1];
        items[0].setSelected(true);
        return menuBar;
    }

    // ---------------------------------------------------------------------
    // create Radio Menu Group
    // ---------------------------------------------------------------------
    private JRadioButtonMenuItem[] createRadioMenuGroup(String[] itemNames, JMenu menu, ActionListener listener) {
        JRadioButtonMenuItem[] items = new JRadioButtonMenuItem[itemNames.length];
        ButtonGroup group = new ButtonGroup();
        for (int i = 0; i < itemNames.length; i++) {
            JRadioButtonMenuItem item = new JRadioButtonMenuItem(itemNames[i]);
            group.add(item);
            items[i] = item;
            menu.add(item);
            if (listener != null) {
                item.addActionListener(listener);
            }
        }
        return items;
    }
}
```