```
/* ------------------------------------------------------------------------
 *
 * Author -------------- Cecilia Y. Sui
 * Assignment ---------- 3D House with Windows (Exam#2)
 * Course -------------- Computer Graphics
 * Instructor ---------- Dr. Crawley
 * Date of Submission -- November 13, 2019
 * Language Used ------- Java & OpenGL
 * Class Imported ------ Camera from Camera.java (written by Dr. Eck)
 * -------------------- ObjectsToDraw (written by Cecilia S.)
 * Description --------- Use JOGL to draw a house with windows
 * Warning ------------- It takes a while to render the graphics.
------------------------------------------------------------------------ */

//-------------------------------------------------------------------------
// Imports
//-------------------------------------------------------------------------
import java.awt.*;
import java.awt.event.*;
import javax.swing.*;
import com.jogamp.opengl.*;
import com.jogamp.opengl.awt.*;
import com.jogamp.opengl.util.gl2.GLUT;
import javax.imageio.ImageIO;
import com.jogamp.opengl.util.texture.Texture;
import com.jogamp.opengl.util.texture.awt.AWTTextureIO;
import com.jogamp.opengl.util.awt.ImageUtil;
import java.awt.image.BufferedImage;
import java.net.URL;

// -------------------------------------------------------------------------
// HouseWindow class
// -------------------------------------------------------------------------
public class HouseWindow extends GLJPanel implements GLEventListener {
    public static void main(String[] args) {
        JFrame window = new JFrame("House with windows");
        HouseWindow panel = new HouseWindow();
        window.setContentPane(panel);
        window.pack();
        window.setLocation(100, 100);
        window.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        window.setVisible(true);
        panel.requestFocusInWindow();
    }

    // ---------------------------------------------------------------------
    // Constructor
    // ---------------------------------------------------------------------
    public HouseWindow() {
        setPreferredSize(new Dimension(800, 800));
        addGLEventListener(this);
```

```
}

// ------------------------------------------------------------------
// Declare private variables
// ------------------------------------------------------------------
private Camera camera;
private ObjectsToDraw currObject;

private int currentTexture;

private String[] textureFileNames = { "marble.jpg", // 0
        "grass.jpg", // 1
        "bluewall.jpg", // 2
        "brick001.jpg", // 3
        "teapot.jpg", // 4
        "road.jpg", // 5
        "bluewall.jpg", // 6
        "pinkwall.jpg" // 7
};

private Texture[] textures = new Texture[textureFileNames.length];

// ------------------------------------------------------------------
// 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);
    camera.apply(gl2);
    gl2.glBindTexture(GL2.GL_TEXTURE_2D, 0);

    float amb[] = { 0.1f, 0.1f, 0.1f, 1.0f };
    float dif[] = { 0, 0, 0.1f, 1.0f };
    float spe[] = { 0, 0.8f, 0, 1.0f };

    gl2.glColor3d(1, 1, 1);
    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);

    // ------------------------------------------------------------------
    // ----- EDIT HERE ------
    // Scale the scene to see details and overview
    // ------------------------------------------------------------------
    gl2.glScaled(0.9, 0.9, 0.9);
    gl2.glTranslated(0, -3, 0);

    // ------------------------------------------------------------------
    // draw objects
```

```
// ------------------------------------------------------------
currObject = ObjectsToDraw.plane;
textures[1].bind(gl2);
drawobject(gl2, currObject);
drawfloors(gl2);
drawtable(gl2);
drawteapot(gl2);

// draw walls
drawwalls(gl2);
gl2.glPushMatrix();
gl2.glTranslated(0, 0, -5.6);
drawwalls(gl2);
gl2.glPopMatrix();
gl2.glPushMatrix();
gl2.glTranslated(0, 3.1, -5.6);
drawwalls2(gl2);
gl2.glTranslated(0, 0, 5.6);
drawwalls2(gl2);
gl2.glPopMatrix();

// draw side walls
currObject = ObjectsToDraw.sidewall;
drawobject(gl2, currObject);
gl2.glPushMatrix();
gl2.glTranslated(5.6, 0, 0);
drawobject(gl2, currObject);
gl2.glTranslated(0, 3.1, 0);
drawobject(gl2, currObject);
gl2.glTranslated(-5.6, 0, 0);
drawobject(gl2, currObject);
gl2.glPopMatrix();

// draw side door
gl2.glPushMatrix();
gl2.glColor3d(1, 0.49, 0);
currObject = ObjectsToDraw.sidedoor;
drawobject(gl2, currObject);
gl2.glPopMatrix();

// draw lightbar
gl2.glColor3d(1, 0.49, 0);
currObject = ObjectsToDraw.lightbar;
drawobject(gl2, currObject);

// bipyramid
gl2.glPushMatrix();
gl2.glColor3d(0, 0, 0);
gl2.glTranslated(0, 5.2, 0);
gl2.glScaled(0.1, 0.1, 0.1);
gl2.glRotated(90, 1, 0, 0);
```

```
    currObject = ObjectsToDraw.bipyramid;
    gl2.glPushMatrix();
    // internal emission light source
    float emit[] = { 1, 0.86f, 0, 1 };
    gl2.glMaterialfv(GL2.GL_FRONT_AND_BACK, GL2.GL_EMISSION, emit, 0);
    drawobject(gl2, currObject);
    gl2.glPopMatrix();
    gl2.glPopMatrix();
}

// ----------------------------------------------------------------------
// draw functions for separate objects
// ----------------------------------------------------------------------
public void drawfloors(GL2 gl2) {
    textures[0].bind(gl2);
    currObject = ObjectsToDraw.floor1;
    drawobject(gl2, currObject);
    gl2.glPushMatrix();
    gl2.glTranslated(0, 3.1, 0);
    drawobject(gl2, currObject);
    gl2.glTranslated(0, 3.1, 0);
    drawobject(gl2, currObject);
    gl2.glPopMatrix();
}

public void drawwalls(GL2 gl2) {
    gl2.glPushMatrix();
    textures[7].bind(gl2);
    currObject = ObjectsToDraw.wall1;
    drawobject(gl2, currObject);
    currObject = ObjectsToDraw.wall2;
    drawobject(gl2, currObject);
    currObject = ObjectsToDraw.wall3;
    drawobject(gl2, currObject);
    currObject = ObjectsToDraw.wall4;
    drawobject(gl2, currObject);
    currObject = ObjectsToDraw.wall5;
    drawobject(gl2, currObject);
    gl2.glPopMatrix();
}

public void drawwalls2(GL2 gl2) {
    gl2.glPushMatrix();
    currObject = ObjectsToDraw.wall1;
    drawobject(gl2, currObject);
    currObject = ObjectsToDraw.wall2;
    drawobject(gl2, currObject);
    currObject = ObjectsToDraw.wall3;
    drawobject(gl2, currObject);
    currObject = ObjectsToDraw.wall4;
    drawobject(gl2, currObject);
```

```
    gl2.glPopMatrix();
}

public void drawtable(GL2 gl2) {
    textures[6].bind(gl2);
    currObject = ObjectsToDraw.tabletop;
    drawobject(gl2, currObject);
    currObject = ObjectsToDraw.leg1;
    drawobject(gl2, currObject);
    currObject = ObjectsToDraw.leg2;
    drawobject(gl2, currObject);
    currObject = ObjectsToDraw.leg3;
    drawobject(gl2, currObject);
    currObject = ObjectsToDraw.leg4;
    drawobject(gl2, currObject);

    gl2.glPushMatrix();
    gl2.glTranslated(0, 3.1, 0);
    currObject = ObjectsToDraw.tabletop;
    drawobject(gl2, currObject);
    currObject = ObjectsToDraw.leg1;
    drawobject(gl2, currObject);
    currObject = ObjectsToDraw.leg2;
    drawobject(gl2, currObject);
    currObject = ObjectsToDraw.leg3;
    drawobject(gl2, currObject);
    currObject = ObjectsToDraw.leg4;
    drawobject(gl2, currObject);
    gl2.glPopMatrix();
}

public void drawteapot(GL2 gl2) {
    gl2.glPushMatrix();
    GLUT glut = new GLUT();
    gl2.glTranslated(0, 1.22, 0);
    textures[4].bind(gl2);
    gl2.glFrontFace(GL2.GL_CW);
    glut.glutSolidTeapot(0.3);
    gl2.glFrontFace(GL2.GL_CCW);
    gl2.glPopMatrix();
}

// ---------------------------------------------------------------
// drawobject method (with normal vectors)
// ---------------------------------------------------------------
public void drawobject(GL2 gl2, ObjectsToDraw currObject) {
    gl2.glPushMatrix();
    double Vx, Vy, Vz, Wx, Wy, Wz, Nx, Ny, Nz, x[], y[], z[];
    int i, j, vertexNum;
    int[][] faces = currObject.faces;
    double[][] vertices = currObject.vertices;
```

```
    for (i = 0; i < faces.length; i++) {
        gl2.glBegin(GL2.GL_TRIANGLE_FAN);
        // get coordinates of 3 vertices
        x = vertices[faces[i][0]];
        y = vertices[faces[i][1]];
        z = vertices[faces[i][2]];
        // convert to free vectors
        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;
        // get normal vector
        gl2.glNormal3d(Nx, Ny, Nz);
        for (j = 0; j < faces[i].length; j++) {
            vertexNum = faces[i][j];
            if (j == 0) {
                gl2.glTexCoord2d(0, 0);
            } else if (j == 1) {
                gl2.glTexCoord2d(1, 0);
            } else if (j == 2) {
                gl2.glTexCoord2d(0.5, 0.5);
            }
            gl2.glVertex3dv(vertices[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();
    // ----------------------------------------------------------------
```

```
        // ----- EDIT HERE ------
        // Change Camera setting
        // ---------------------------------------------------------------
        camera.lookAt(1, 8, -50, 0, 0, 1, 0, 1, 0);
        camera.installTrackball(this);

        // ---------------------------------------------------------------
        // Light source: LIGHT0 (directional)
        // ---------------------------------------------------------------
        gl2.glEnable(GL2.GL_LIGHT0);
        float diffuse[] = { 0.94f, 0.87f, 0.8f, 1.0f };
        float specular[] = { 0.1f, 0f, 0.5f, 1.0f };
        float ambient[] = { 0.54f, 0.47f, 0.4f, 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);

        // ---------------------------------------------------------------
        // Textures
        // ---------------------------------------------------------------
        for (int i = 0; i < textureFileNames.length; i++) {
            try {
                URL textureURL;
                textureURL =
getClass().getClassLoader().getResource("textures/" +
textureFileNames[i]);
                if (textureURL != null) {
                    BufferedImage img = ImageIO.read(textureURL);
                    ImageUtil.flipImageVertically(img);
                    textures[i] =
AWTTextureIO.newTexture(GLProfile.getDefault(), img, true);
                    textures[i].setTexParameteri(gl2,
GL2.GL_TEXTURE_WRAP_S, GL2.GL_REPEAT);
                    textures[i].setTexParameteri(gl2,
GL2.GL_TEXTURE_WRAP_T, GL2.GL_REPEAT);
                }
            } catch (Exception e) {
                e.printStackTrace();
            }
        }
        textures[0].enable(gl2);
    }

    public void dispose(GLAutoDrawable graphics) {
    }

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

}
```