Prachet Agrawal

Assignment 2

CMSC 405

Shows a scene (a teapot on a short cylindrical base) that is illuminated by up to four lights plus global ambient light. The user can turn the lights on and off. The global ambient light is a dim white. There is a white "viewpoint" light that points from the direction of the viewer into the scene. There is a red light, a blue light, and a green light that rotate in circles above the teapot. (The user can turn the animation on and off.) The locations of the colored lights are marked by spheres, which are gray when the light is off and are colored by some emission color when the light is on. The teapot is gray with weak specular highlights. The base is colored with a spectrum. (The user can turn the display ofthe base on and off.) The mouse can be used to rotate the scene.

public static void main(String[] args) {  
 JFrame window = new JFrame("A Lighting Demo");  
 FourLights panel = new FourLights();  
 window.setContentPane(panel);  
 window.pack();  
 window.setLocation(50,50);  
 window.setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);  
 window.setVisible(true);  
}  
  
private JCheckBox animating; // Checked if animation is running.  
  
private JCheckBox viewpointLight; // Checked if the white viewpoint light is on.  
private JCheckBox redLight; // Checked if the red light is on.  
private JCheckBox greenLight; // Checked if the green light is on.  
private JCheckBox blueLight; // Checked if the blue light is on.  
private JCheckBox ambientLight; // Checked if the global ambient light is on.  
  
private JCheckBox drawBase; // Checked if the base should be drawn.  
  
private GLJPanel display;  
private Timer animationTimer;  
  
private int frameNumber = 0; // The current frame number for an animation.  
  
private Camera camera;  
  
private GLUT glut = new GLUT();

The constructor adds seven checkboxes under the display, to control the options.

{  
 GLCapabilities caps = new GLCapabilities(null);  
 display = new GLJPanel(caps);  
 display.setPreferredSize( new Dimension(600,600) );  
 display.addGLEventListener(this);  
 setLayout(new BorderLayout());  
 add(display,BorderLayout.CENTER);  
 camera = new Camera();  
 camera.lookAt(5,10,30, 0,0,0, 0,1,0);  
 camera.setScale(15);  
 camera.installTrackball(display);  
 animationTimer = new Timer(30, new ActionListener() {  
 public void actionPerformed(ActionEvent evt) {  
 frameNumber++;  
 display.repaint();  
 }  
 });  
 ActionListener boxHandler = new ActionListener() {  
 public void actionPerformed(ActionEvent evt) {  
 if (evt.getSource() == animating) {  
 if (animating.isSelected()) {  
 animationTimer.start();  
 }  
 else {  
 animationTimer.stop();  
 }  
 }  
 else {  
 display.repaint();  
 }  
 }  
 };  
 viewpointLight = new JCheckBox("Viewpoint Light", true);  
 redLight = new JCheckBox("Red Light", true);  
 blueLight = new JCheckBox("Blue Light", true);  
 greenLight = new JCheckBox("Green Light", true);  
 ambientLight = new JCheckBox("Global Ambient Light", true);  
 animating = new JCheckBox("Animate", true);  
 drawBase = new JCheckBox("Draw Base", true);  
 viewpointLight.addActionListener(boxHandler);  
 ambientLight.addActionListener(boxHandler);  
 redLight.addActionListener(boxHandler);  
 greenLight.addActionListener(boxHandler);  
 blueLight.addActionListener(boxHandler);  
 animating.addActionListener(boxHandler);  
 drawBase.addActionListener(boxHandler);  
 JPanel bottom = new JPanel();  
 bottom.setLayout(new GridLayout(2,1));  
 JPanel row1 = new JPanel();  
 row1.add(animating);  
 row1.add(drawBase);  
 row1.add(ambientLight);  
 bottom.add(row1);  
 JPanel row2 = new JPanel();  
 row2.add(viewpointLight);  
 row2.add(redLight);  
 row2.add(greenLight);  
 row2.add(blueLight);  
 bottom.add(row2);  
 add(bottom,BorderLayout.SOUTH);  
 animationTimer.setInitialDelay(500);  
 animationTimer.start();  
}

Methods for drawing

Sets the positions of the colored lights and turns them on and off, depending on the state of the redLight, greenLight, and blueLight options. Draws a small sphere at the location of each light.

private void square(GL3 gl3) {  
 gl3.glColor3d(1, 1, 1);  
gl3.glBegin(GL3.GL\_TRIANGLE\_FAN);  
gl3.glNormal3d(0, 0, 1);  
gl3.glVertex3d(-0.5, -0.5, 0.5);  
gl3.glVertex3d(0.5, -0.5, 0.5);  
gl3.glVertex3d(0.5, 0.5, 0.5);  
gl3.glVertex3d(-0.5, 0.5, 0.5);  
gl3.glEnd();  
 }  
   
 private void lights(GL3 gl) {  
  
 gl.glColor3d(0.5,0.5,0.5);  
 float zero[] = { 0, 0, 0, 1 };  
 gl.glMaterialfv(GL3.GL\_FRONT\_AND\_BACK, GL3.GL\_SPECULAR, zero, 0);  
   
 if (viewpointLight.isSelected())  
 gl.glEnable(GL3.GL\_LIGHT0);  
 else  
 gl.glDisable(GL3.GL\_LIGHT0);  
   
 if (redLight.isSelected()) {  
 float red[] = { 0.5F, 0, 0, 1 };  
 gl.glMaterialfv(GL3.GL\_FRONT\_AND\_BACK, GL3.GL\_EMISSION, red, 0);   
 gl.glEnable(GL3.GL\_LIGHT1);  
 }  
 else {  
 gl.glMaterialfv(GL3.GL\_FRONT\_AND\_BACK, GL3.GL\_EMISSION, zero, 0);   
 gl.glDisable(GL3.GL\_LIGHT1);  
 }  
 gl.glPushMatrix();  
 gl.glRotated(-frameNumber, 0, 1, 0);  
 gl.glTranslated(10, 7, 0);  
 gl.glLightfv(GL3.GL\_LIGHT1, GL3.GL\_POSITION, zero, 0);  
 glut.glutSolidSphere(0.5, 16, 8);  
 gl.glPopMatrix();  
   
 if (greenLight.isSelected()) {  
 float green[] = {0, 0.5F, 0, 1 };  
 gl.glMaterialfv(GL3.GL\_FRONT\_AND\_BACK, GL3.GL\_EMISSION, green, 0);   
 gl.glEnable(GL3.GL\_LIGHT2);  
 }  
 else {  
 gl.glMaterialfv(GL3.GL\_FRONT\_AND\_BACK, GL3.GL\_EMISSION, zero, 0);   
 gl.glDisable(GL3.GL\_LIGHT2);  
 }  
 gl.glPushMatrix();  
 gl.glRotated((frameNumber+100)\*0.8743, 0, 1, 0);  
 gl.glTranslated(9, 8, 0);  
 gl.glLightfv(GL3.GL\_LIGHT2, GL3.GL\_POSITION, zero, 0);  
 glut.glutSolidSphere(0.5, 16, 8);  
 gl.glPopMatrix();  
   
 if (blueLight.isSelected()) {  
 float blue[] = { 0, 0, 0.5F, 1 };  
 gl.glMaterialfv(GL3.GL\_FRONT\_AND\_BACK, GL3.GL\_EMISSION, blue, 0);   
 gl.glEnable(GL3.GL\_LIGHT3);  
 }  
 else {  
 gl.glMaterialfv(GL3.GL\_FRONT\_AND\_BACK, GL3.GL\_EMISSION, zero, 0);  
 gl.glDisable(GL3.GL\_LIGHT3);  
 }  
 gl.glPushMatrix();  
 gl.glRotated((frameNumber-100)\*1.3057, 0, 1, 0);  
 gl.glTranslated(9.5, 7.5, 0);  
 gl.glLightfv(GL3.GL\_LIGHT3, GL3.GL\_POSITION, zero, 0);  
 glut.glutSolidSphere(0.5, 16, 8);  
 gl.glPopMatrix();  
  
 gl.glMaterialfv(GL3.GL\_FRONT\_AND\_BACK, GL3.GL\_EMISSION, zero, 0); // Turn off emission color!  
 } // end lights()

Test Cases

The following test cases have been run to cover the program and show the edge cases are passing.

|  |  |  |  |
| --- | --- | --- | --- |
| Sr No. | Test Case | Scenario | Output |
| 1 | 3D shape is considered lighted | In this test case, the 3D shape is run through the JOGL function and the arguments are passed as 1 & 0 | Pass |
| 2 | 2D shape is considered lighted | In this test case, the 2D shape is run through the JOGL function and the arguments are passed as 0 & 1 | Pass |
| 3 | 2D shape is not considered lighted | In this test case, the 2D shape is run through the JOGL function and the arguments are passed as 1 & 0 | Fail |
| 4 | 3D shape is not considered lighted | In this test case, the 3D shape is run through the JOGL function and the arguments are passed as 0 & 1 | Fail |

Lessons Learned

Through this assignment, I have been able to learn about the effects of lighted & unlighted 2D & 3D shapes and what happens when they are passed in the JOGL function with different combination of 1& 0 passed as arguments