KINGDOM OF SAUDI ARABIA
Ministry of Higher Education
Taibah University
College of Computer Science and Engineering (Girls Section)

المملكة العربية السعودية
وزارة التعليم العالي
جامعة طيبة
كلية علوم و هندسة الحاسبات
(قسم الطالبات)

Project Computer Graphics

Sec :C9A

Student Name / ID

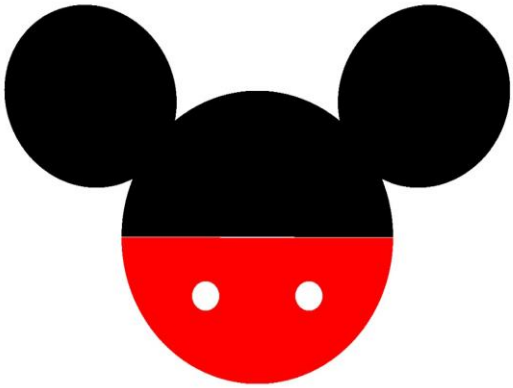
Sana Muhammad Bassam Qarah / ID: 3752533

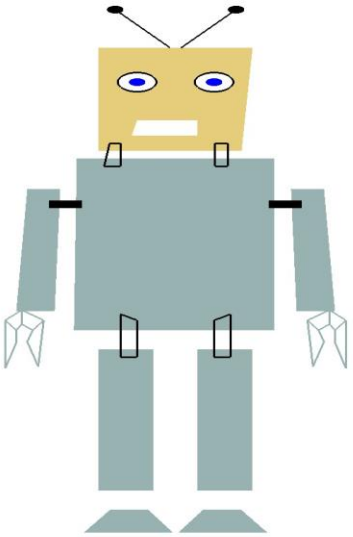
Maryam Hatem Alrdaddi / ID: 3752699

Amera Abdullah Almuhammadi / ID: 3752744

Salwa Eid Alanzi / ID: 3754142

Screen shot





I. Code

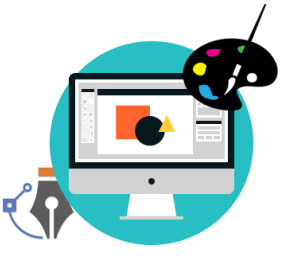
```
1  package cgproject;
2
3  import java.awt.BorderLayout;
4  import java.awt.event.ActionEvent;
5  import java.awt.event.ActionListener;
6  import java.awt.event.KeyListener;
7  import java.awt.event.KeyEvent;
8  import java.awt.event.MouseListener;
9  import java.awt.event.MouseEvent;
10 import com.jogamp.opengl.glu.GLU;
11 import com.jogamp.opengl.GL2;
12 import com.jogamp.opengl.GLAutoDrawable;
13 import com.jogamp.opengl.GLCapabilities;
14 import com.jogamp.opengl.GLEventListener;
15 import com.jogamp.opengl.GLProfile;
16 import com.jogamp.opengl.awt.GLCanvas;
17 import com.jogamp.opengl.util.gl2.GLUT;
18 import com.jogamp.opengl.util.FPSAnimator;
19 import javax.swing.JButton;
20 import javax.swing.JFrame;
21 import javax.swing.JPanel;
22
23 public class CGProject {
24
25     static boolean robotOn = false;
26     static boolean mickeyOn = false;
27
28     public static void main(String[] args) {
29
30         // ------- Robot initialization -------
31         // getting the capabilities object of GL2 profile
32         GLProfile profile = GLProfile.get(GLProfile.GL2);
33         GLCapabilities capabilities = new GLCapabilities(profile);
34         GLCanvas robot_glcanvas = new GLCanvas(capabilities);
35         FPSAnimator robot_animator = new FPSAnimator(robot_glcanvas, 300, true);
36
37         // ------- Mickey initialization -------
38         GLCanvas mickey_glcanvas = new GLCanvas();
39         FPSAnimator mickey_animator = new FPSAnimator(mickey_glcanvas, 300, true);
40
41         // initialize frame
42         JFrame frame = new JFrame("CGProject");
43         frame.setResizable(false);
44         frame.setSize(900, 820);
45         frame.setLocationRelativeTo(null);
46         frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
47
48         // initialize panel
49         JPanel panel = new JPanel();
50         panel.setLayout(new BorderLayout());
51
52         // add button to buttonPanel
53         JPanel buttonPanel = new JPanel();
54         JButton buttonRobot = new JButton();
55         buttonRobot.setText("Robot □");
56         buttonPanel.add(buttonRobot);
57
58         JButton buttonMickey = new JButton();
59         buttonMickey.setText("Mickey □");
60         buttonPanel.add(buttonMickey);
61
62         // add buttonPanel to panel
63         panel.add(buttonPanel, BorderLayout.NORTH);
64         // add panel to frame
65         frame.add(panel);
66
67         // handle robot button click
68         buttonRobot.addActionListener(new ActionListener() {
69
```

```
70                  @Override
                    public void actionPerformed(ActionEvent arg0) {
72
73                      if (mickeyOn) {
74                          // if mickey_glcanvas is on frame remove it
75                          frame.getContentPane().remove(mickey_glcanvas);
76                      }
77
78                      // setup robot canvas
79                      Robot prog = new Robot();
80                      robot_glcanvas.addGLEventListener(prog);
81                      robot_glcanvas.addKeyListener(prog);
82                      robot_glcanvas.addMouseListener(prog);
83                      robot_glcanvas.setSize(800, 700);
84                      robot_glcanvas.setLocation(40, 50);
85                      // add canvas to frame
86                      frame.getContentPane().add(robot_glcanvas, BorderLayout.CENTER);
87                      robotOn = true;
88
89                      // start the animation
90                      robot_animator.start();
91                  }
92              });
93
94               // handle mickey button click
                buttonMickey.addActionListener(new ActionListener() {
96
97                  @Override
                    public void actionPerformed(ActionEvent arg0) {
99
100                     if (robotOn) {
101                         // if robot_glcanvas is on frame remove it
102                         frame.getContentPane().remove(robot_glcanvas);
103                     }
104
105                     // setup mickey canvas
106                     Mickey prog2 = new Mickey();
107                     mickey_glcanvas.addGLEventListener(prog2);
108                     mickey_glcanvas.addKeyListener(prog2);
109                     mickey_glcanvas.addMouseListener(prog2);
110                     mickey_glcanvas.setSize(800, 700);
111                     mickey_glcanvas.setLocation(40, 50);
112                     // add canvas to frame
113                     frame.getContentPane().add(mickey_glcanvas, BorderLayout.CENTER);
114                     mickeyOn = true;
115
116                     // start the animation
117                     mickey_animator.start();
118                 }
119             });
120
121             // display the frame
122             frame.setVisible(true);
123
124     } //end of main
125
126 }
127
128 class Mickey implements GLEventListener, KeyListener, MouseListener {
129
130     GLUT glut = new GLUT();
131     GLU glu;
132     float r = 1;
133     float g = 1;
134     float b = 1;
135     float d = 1;    //defult background is white (1,1,1,1)
136     float x = 0;    //defult
137
138     public Mickey() {
139
140     }
141
142
```

```
143     @Override
144     public void init(GLAutoDrawable g0) {
145     }
146
147     @Override
148     public void dispose(GLAutoDrawable g0) {
149     }
150     float i = -1, theta = 0;
151
152     @Override
153     public void display(GLAutoDrawable drawable) {
154
155         final GL2 gl = drawable.getGL().getGL2();
156
157         glu = GLU.createGLU(gl);
158         gl.glClear(GL2.GL_COLOR_BUFFER_BIT | GL2.GL_DEPTH_BUFFER_BIT);
159         //
160         gl.glMatrixMode(GL2.GL_PROJECTION);
161         gl.glLoadIdentity();
162         gl.glFrustum(-1.0f, 1.0f, -1f, 1.0f, 1.5f, 20.0f);
163
164         gl.glMatrixMode(GL2.GL_MODELVIEW);
165         gl.glLoadIdentity();
166
167         //camera viewing and x for mouse affect
168         glu.gluLookAt(x, 0, 2.0, 0, 0, 0, 0.0f, 1, 0.7f);
169
170         //background colour
171         gl.glClearColor(r, b, g, d);
172         gl.glFlush();
173
174         int numVertices = 600;
175         double radius = 0.5;
176
177         // clear the window
178         gl.glClear(GL2.GL_COLOR_BUFFER_BIT);
179
180         //RED half circle
181         // gl.glColor3f(r,g,b);
182         gl.glColor3f(1, 0, 0);
183         gl.glPushMatrix();
184         gl.glRotatef(180, 0, 0, 1);
185         gl.glBegin(GL2.GL_TRIANGLE_FAN);
186         {
187             double angle = 0;
188             double angleIncrement = Math.PI / numVertices;
189             for (int i = 0; i < numVertices; i++) {
190                 angle = i * angleIncrement;
191                 double x = radius * Math.cos(angle);
192                 double y = radius * Math.sin(angle);
193                 gl.glVertex2d(x, y);
194             }
195         }
196         gl.glEnd();
197         gl.glPopMatrix();
198
199         //BLACK half circle
200         gl.glColor3f(0, 0, 0);
201         gl.glBegin(GL2.GL_TRIANGLE_FAN);
202         {
203             double angle = 0;
204             double angleIncrement = Math.PI / numVertices;
205             for (int i = 0; i < numVertices; i++) {
206                 angle = i * angleIncrement;
207                 double x = radius * Math.cos(angle);
208                 double y = radius * Math.sin(angle);
209                 gl.glVertex2d(x, y);
210             }
211         }
212         gl.glEnd();
213
214         //medium black circle
215         gl.glPushMatrix();
216         gl.glTranslatef(0.6f, i, 0);
```

```
217             gl.glRotatef(theta, 1, 1, 0); // rotate around x and y
218
219             //translation
220             i += 0.005f;
221             if (i >= 0.47f) {
222                 i = 0.47f;
223             }
224
225             //rotation
226             theta += 0.5;
227             if (theta >= 360) {
228                 theta = 0;
229             }
230
231             gl.glColor3f(0, 0, 0);
232
233             gl.glBegin(GL2.GL_TRIANGLE_FAN);
234             glut.glutSolidSphere(0.3, 100, 100);//3D
235             gl.glPopMatrix();
236
237             //medium black circle
238             gl.glPushMatrix();
239             gl.glTranslatef(-0.6f, i, 0);
240             gl.glRotatef(theta, 1, -1, 0); // rotate around x and y
241
242             //translation
243             i += 0.005f;
244             if (i >= 0.47f) {
245                 i = 0.47f;
246             }
247
248             //rotation
249             theta += 0.5;
250             if (theta >= 360) {
251                 theta = 0;
252             }
253
254             gl.glColor3f(0, 0, 0);
255             glut.glutSolidSphere(0.3, 100, 100); //3D
256             gl.glPopMatrix();
257
258             //right small white circle
259             gl.glPushMatrix();
260             gl.glTranslatef(0.19f, -0.2f, 0);
261             gl.glColor3f(1, 1, 1);
262
263             gl.glBegin(GL2.GL_TRIANGLE_FAN);
264             {
266                 double angle = 0;
266                 double angleIncrement = 2 * Math.PI / numVertices;
267                 for (int i = 0; i < numVertices; i++) {
268                     angle = i * angleIncrement;
269                     double x = 0.05 * Math.cos(angle);
270                     double y = 0.05 * Math.sin(angle);
271                     gl.glVertex2d(x, y);
272                 }
273             }
274             gl.glEnd();
275             gl.glPopMatrix();
276
277             //left small white circle
278             gl.glPushMatrix();
279             gl.glTranslatef(-0.19f, -0.2f, 0);
280             gl.glColor3f(1, 1, 1);
281
282             gl.glBegin(GL2.GL_TRIANGLE_FAN);
283             {
285                 double angle = 0;
285                 double angleIncrement = 2 * Math.PI / numVertices;
287                 for (int i = 0; i < numVertices; i++) {
287                     angle = i * angleIncrement;
288                     double x = 0.05 * Math.cos(angle);
289                     double y = 0.05 * Math.sin(angle);
```

```
290                 gl.glVertex2d(x, y);
291             }
292         }
293         gl.glEnd();
294         gl.glPopMatrix();
295     }
296
297     @Override
        public void reshape(GLAutoDrawable glad, int i, int i1, int i2, int i3) {
299     }
300
301     @Override
        public void keyTyped(KeyEvent ke) {
303         char key = ke.getKeyChar();
304
305         System.out.printf("Key typed: %c\n", key);
306
307         switch (key) {
308
309             case 'r':
310                 System.out.println("Changing the background Color to RED....");
311                 r = 0.7f;
312                 g = 0.2f;
313                 b = 0.12f;
314                 d = 0;
315                 break;
316             case 'g':
317                 System.out.println("Changing the background Color to GREEN....");
318                 r = 0.2f;
319                 g = 0.5f;
320                 b = 0.7f;
321                 d = 0;
322                 break;
323             case 'b':
324                 System.out.println("Changing the background Color to BLUE....");
325                 r = 0.2f;
326                 g = 0.8f;
327                 b = 0.5f;
328                 d = 0.3f;
329                 break;
330             case 'p':
331                 System.out.println("Changing the background Color to PINK....");
332                 r = 1f;
333                 g = 0.6f;
334                 b = 0.5f;
335                 d = 0f;
336                 break;
337
338             case 'y':
339                 System.out.println("Changing the background Color to YELLOW....");
340                 r = 0.9f;
341                 g = 0.6f;
342                 b = 1f;
343                 d = 0f;
344                 break;
345
346             case 27:
347                 System.out.println("Exit!");
348                 System.exit(0);
349         }
350     }
351
352     @Override
        public void keyPressed(KeyEvent ke) {
354     }
355
356     @Override
        public void keyReleased(KeyEvent ke) {
358     }
359
360     @Override
        public void mouseClicked(MouseEvent me) {
362         if (me.getButton() == MouseEvent.BUTTON1) {
```



```
363             x += 0.1;
364         }
365         if (me.getButton() == MouseEvent.BUTTON3) {
366             x -= 0.1;
367         }
368 
369         System.out.println("(" + me.getX() + "," + me.getY() + ")");
370     }
371 
372     @Override
373     public void mousePressed(MouseEvent me) {
374     }
375 
376     @Override
377     public void mouseReleased(MouseEvent me) {
378     }
379 
380     @Override
381     public void mouseEntered(MouseEvent me) {
382     }
383 
384     @Override
385     public void mouseExited(MouseEvent me) {
386     }
387 
388 }
389 
390 class Robot implements GLEventListener, KeyListener, MouseListener {
391 
392     static boolean robotOn = false;
393     static boolean mickeyOn = false;
394     GLUT glut = new GLUT();
395     GLU glu;
396     float PI = 3.14154f;
397     double theta, x, y;
398     int r = 0, g = 0, b = 0;
399     int XX = 1, YY = 1, ZZ = 1;
400 
401     float xx = 0;
402 
403     void drawCircle(GL2 gl, float xPos, float yPos, float rx, float ry) {
404 
405         gl.glBegin(GL2.GL_POLYGON);
406         for (theta = 0; theta <= (2 * PI); theta += 2 * PI / 1000) {
407             x = xPos + rx * Math.cos(theta);
408             y = yPos + ry * Math.sin(theta);
409             gl.glVertex2d(x, y);
410         }
411         gl.glEnd();
412 
413     }
414 
415     void drawSolidCircle(GL2 gl, float xPos, float yPos, float rx, float ry) {
416         gl.glBegin(GL2.GL_LINE_LOOP);
417         for (theta = 0; theta <= (2 * PI); theta += 2 * PI / 1000) {
418             x = xPos + rx * Math.cos(theta);
419             y = yPos + ry * Math.sin(theta);
420             gl.glVertex2d(x, y);
421         }
422         gl.glEnd();
423     }
424 
425     @Override
426     public void display(GLAutoDrawable drawable) {
427 
428         final GL2 gl = drawable.getGL().getGL2();
429         glu = GLU.createGLU(gl);
430         gl.glClearColor(1, 1, 1, 1);
431         gl.glClear(GL2.GL_COLOR_BUFFER_BIT | GL2.GL_DEPTH_BUFFER_BIT);
432         gl.glMatrixMode(GL2.GL_PROJECTION);
433         gl.glLoadIdentity();
434 
435         gl.glPushMatrix();
```

```
436             gl.glTranslatef(xx, 0, 0);
437             // Draw the hesd
438             gl.glColor3f(0.9f, 0.8f, 0.5f);
439             gl.glBegin(GL2.GL_POLYGON);
440             gl.glVertex2f(-0.14f, 0.47f);
441             gl.glVertex2f(-0.14f, 0.74f);
442             gl.glVertex2f(0.14f, 0.74f);
443             gl.glVertex2f(0.12f, 0.47f);
444             gl.glEnd();
445             // Draw the eyes
446             // External eyes
447             gl.glColor3f(1, 1, 1);
448             drawCircle(gl, -0.07f, 0.65f, .035f, .025f);
449             drawCircle(gl, 0.07f, 0.65f, .035f, .025f);
450             gl.glColor3f(0, 0, 0);
451             drawSolidCircle(gl, -0.07f, 0.65f, .035f, .025f);
452             drawSolidCircle(gl, 0.07f, 0.65f, .035f, .025f);
453             //Internal eyes
454             gl.glColor3f(0, 0, 1);
455             drawCircle(gl, -0.07f, 0.65f, .015f, .010f);
456             drawCircle(gl, 0.07f, 0.65f, .015f, .010f);
457             //Draw the Mouth
458             // Mouth
459             gl.glColor3f(1, 1, 1);
460             gl.glBegin(GL2.GL_QUADS);
461             gl.glVertex2f(-0.08f, 0.51f);
462             gl.glVertex2f(-0.07f, 0.55f);
463             gl.glVertex2f(0.04f, 0.55f);
464             gl.glVertex2f(0.04f, 0.51f);
465             gl.glEnd();
466             // Teeth
467             gl.glColor3f(XX, YY, ZZ);
468             gl.glBegin(GL2.GL_QUADS);
469             gl.glVertex2f(-0.07f, 0.523f);
470             gl.glVertex2f(-0.07f, 0.533f);
471             gl.glVertex2f(0.04f, 0.533f);
472             gl.glVertex2f(0.04f, 0.523f);
473             gl.glEnd();
474             // Anternnae
475             gl.glColor3f(0, 0, 0);
476             gl.glLineWidth(3.0f);
477             gl.glBegin(GL2.GL_LINES);
478             gl.glVertex2f(-0.01f, 0.74f);
479             gl.glVertex2f(-0.11f, 0.84f);
480             gl.glVertex2f(0.01f, 0.74f);
481             gl.glVertex2f(0.11f, 0.84f);
482             gl.glEnd();
483             gl.glColor3f(r, g, b);
484             drawCircle(gl, -0.11f, 0.84f, .015f, .010f);
485             drawCircle(gl, 0.11f, 0.84f, .015f, .010f);
486         //Body
487             gl.glColor3f(0.6f, 0.7f, 0.7f);
488             gl.glBegin(GL2.GL_POLYGON);
489
490             gl.glVertex2f(-0.185f, 0f);
491             gl.glVertex2f(-0.18f, 0.45f);
492             gl.glVertex2f(0.18f, 0.45f);
493             gl.glVertex2f(0.18f, 0.0f);
494             gl.glEnd();
495         //Left Arm
496
497             gl.glBegin(GL2.GL_POLYGON);
498             gl.glVertex2f(-0.29f, 0.05f);
499             gl.glVertex2f(-0.27f, 0.37f);
500             gl.glVertex2f(-0.21f, 0.37f);
501             gl.glVertex2f(-0.22f, 0.05f);
502             gl.glEnd();
503         // Right Arm
504
505             gl.glBegin(GL2.GL_POLYGON);
506             gl.glVertex2f(0.29f, 0.05f);
507             gl.glVertex2f(0.27f, 0.37f);
508             gl.glVertex2f(0.21f, 0.37f);
509             gl.glVertex2f(0.22f, 0.05f);
```

```
            gl.glEnd();
    // Hand

            //Left Hand
            gl.glBegin(GL2.GL_LINE_LOOP);
            gl.glVertex2f(-0.31f, 0.026f); //2
            gl.glVertex2f(-0.31f, -0.1f); //1
            gl.glVertex2f(-0.29f, -0.0325f);//10
            gl.glVertex2f(-0.29f, 0f);//9
            gl.glEnd();

            gl.glBegin(GL2.GL_LINE_LOOP);
            gl.glVertex2f(-0.28f, 0.053f);//3
            gl.glVertex2f(-0.31f, 0.026f); //2
            gl.glVertex2f(-0.29f, 0f);//9
            gl.glVertex2f(-0.28f, 0.016f);//8
            gl.glEnd();
            gl.glBegin(GL2.GL_LINE_LOOP);
            gl.glVertex2f(-0.24f, 0.026f);//4
            gl.glVertex2f(-0.28f, 0.053f);//3
            gl.glVertex2f(-0.28f, 0.016f);//8
            gl.glVertex2f(-0.26f, 0f);//7
            gl.glEnd();

            gl.glBegin(GL2.GL_LINE_LOOP);
            gl.glVertex2f(-0.24f, 0.026f);//4
            gl.glVertex2f(-0.26f, 0f);//7
            gl.glVertex2f(-0.27f, -0.03f);//6
            gl.glVertex2f(-0.25f, -0.095f);//5
            gl.glEnd();
            //Right Hand
            gl.glBegin(GL2.GL_LINE_LOOP);
            gl.glVertex2f(0.31f, 0.026f); //2
            gl.glVertex2f(0.31f, -0.1f); //1
            gl.glVertex2f(0.29f, -0.0325f);//10
            gl.glVertex2f(0.29f, 0f);//9
            gl.glEnd();

            gl.glBegin(GL2.GL_LINE_LOOP);
            gl.glVertex2f(0.28f, 0.053f);//3
            gl.glVertex2f(0.31f, 0.026f); //2
            gl.glVertex2f(0.29f, 0f);//9
            gl.glVertex2f(0.28f, 0.016f);//8
            gl.glEnd();
            gl.glBegin(GL2.GL_LINE_LOOP);
            gl.glVertex2f(0.24f, 0.026f);//4
            gl.glVertex2f(0.28f, 0.053f);//3
            gl.glVertex2f(0.28f, 0.016f);//8
            gl.glVertex2f(0.26f, 0f);//7
            gl.glEnd();

            gl.glBegin(GL2.GL_LINE_LOOP);
            gl.glVertex2f(0.24f, 0.026f);//4
            gl.glVertex2f(0.26f, 0f);//7
            gl.glVertex2f(0.27f, -0.03f);//6
            gl.glVertex2f(0.25f, -0.095f);//5
            gl.glEnd();
    //Left Leg

            gl.glBegin(GL2.GL_POLYGON);
            gl.glVertex2f(-0.04f, -0.42f);
            gl.glVertex2f(-0.04f, -0.05f);
            gl.glVertex2f(-0.14f, -0.05f);
            gl.glVertex2f(-0.14f, -0.42f);
            gl.glEnd();
    // Right Lag

            gl.glBegin(GL2.GL_POLYGON);
            gl.glVertex2f(0.04f, -0.42f);
            gl.glVertex2f(0.04f, -0.05f);
            gl.glVertex2f(0.14f, -0.05f);
            gl.glVertex2f(0.14f, -0.42f);
            gl.glEnd();
```

```
584     //left Foot
585             gl.glBegin(GL2.GL_POLYGON);
586             gl.glVertex2f(-0.167f, -0.53f);
587             gl.glVertex2f(-0.119f, -0.47f);
588             gl.glVertex2f(-0.05f, -0.47f);
589             gl.glVertex2f(-0.005f, -0.53f);
590             gl.glEnd();
591
592     //Right Foot
593             gl.glBegin(GL2.GL_POLYGON);
594             gl.glVertex2f(0.167f, -0.53f);
595             gl.glVertex2f(0.119f, -0.47f);
596             gl.glVertex2f(0.05f, -0.47f);
597             gl.glVertex2f(0.005f, -0.53f);
598             gl.glEnd();
599     //Breaks
600             //Breaks Body
601             // Brearks Left Body
602             gl.glColor3f(0, 0, 0);
603             gl.glBegin(GL2.GL_LINE_LOOP);
604             gl.glVertex2f(-0.12f, 0.49f);
605             gl.glVertex2f(-0.13f, 0.43f);
606             gl.glVertex2f(-0.1f, 0.43f);
607             gl.glVertex2f(-0.1f, 0.49f);
608             gl.glEnd();
609             //Brearks Right Body
610             gl.glColor3f(0, 0, 0);
611             gl.glBegin(GL2.GL_LINE_LOOP);
612             gl.glVertex2f(0.072f, 0.49f);
613             gl.glVertex2f(0.072f, 0.43f);
614             gl.glVertex2f(0.095f, 0.43f);
615             gl.glVertex2f(0.095f, 0.49f);
616             gl.glEnd();
617             //Breaks Arm
618             // Brearks Left Arm
619             gl.glColor3f(0, 0, 0);
620             gl.glBegin(GL2.GL_POLYGON);
621             gl.glVertex2f(-0.23f, 0.34f);
622             gl.glVertex2f(-0.23f, 0.32f);
623             gl.glVertex2f(-0.17f, 0.32f);
624             gl.glVertex2f(-0.17f, 0.34f);
625             gl.glEnd();
626             //Brearks Right Arm
627             gl.glColor3f(0, 0, 0);
628             gl.glBegin(GL2.GL_POLYGON);
629             gl.glVertex2f(0.23f, 0.34f);
630             gl.glVertex2f(0.23f, 0.32f);
631             gl.glVertex2f(0.17f, 0.32f);
632             gl.glVertex2f(0.17f, 0.34f);
633             gl.glEnd();
634             //Breaks leg
635             // Brearks Left leg
636             gl.glColor3f(0, 0, 0);
637             gl.glBegin(GL2.GL_LINE_LOOP);
638             gl.glVertex2f(-0.10f, 0.042f);
639             gl.glVertex2f(-0.10f, -0.07f);
640             gl.glVertex2f(-0.07f, -0.07f);
641             gl.glVertex2f(-0.07f, 0.026f);
642             gl.glEnd();
643             //Brearks Right leg
644             gl.glColor3f(0, 0, 0);
645             gl.glBegin(GL2.GL_LINE_LOOP);
646             gl.glVertex2f(0.10f, 0.042f);
647             gl.glVertex2f(0.10f, -0.07f);
648             gl.glVertex2f(0.07f, -0.07f);
649             gl.glVertex2f(0.07f, 0.026f);
650             gl.glEnd();
651
652             gl.glPopMatrix();
653             xx += 0.0005;
654             if (xx >= 1.4) {
655                 xx = -xx;
656             }
657         }
```

```
658
659     @Override
660     public void dispose(GLAutoDrawable arg0) {
661
662     }
663
664     @Override
665     public void init(GLAutoDrawable drawable) {
666
667     }
668
669     @Override
670     public void reshape(GLAutoDrawable drawable, int x, int y, int width, int height) {
671
672     }
673
674     @Override
675     public void keyTyped(KeyEvent ke) {
676         char key = ke.getKeyChar();
677
678         System.out.printf("Key typed: %c\n", key);
679
680         switch (key) {
681
682             case 'r':
683                 System.out.println("Changing Color to RED....");
684                 r = 1;
685                 g = 0;
686                 b = 0;
687                 break;
688             case 'g':
689                 System.out.println("Changing Color to GREEN....");
690                 r = 0;
691                 g = 1;
692                 b = 0;
693
694                 break;
695             case 'b':
696                 System.out.println("Changing Color to BLUE....");
697                 r = 0;
698                 g = 0;
699                 b = 1;
700                 break;
701
702             case 27:
703                 System.out.println("Exit!");
704                 System.exit(0);
705         }
706     }
707
708     @Override
709     public void keyPressed(KeyEvent ke) {
710     }
711
712     @Override
713     public void keyReleased(KeyEvent ke) {
714     }
715
716     @Override
717     public void mouseClicked(MouseEvent me) {
718         if (me.getButton() == MouseEvent.BUTTON1) {
719             XX = 1;
720         }
721         YY = 0;
722         ZZ = 0;
723         if (me.getButton() == MouseEvent.BUTTON3) {
724             XX = 0;
725         }
726         YY = 0;
727         ZZ = 0;
728
729         System.out.println("(" + me.getX() + "," + me.getY() + ")");
730
731     }
```

```
732
733         @Override
            public void mousePressed(MouseEvent me) {
735
736         }
737
738         @Override
            public void mouseReleased(MouseEvent me) {
740         }
741
742         @Override
            public void mouseEntered(MouseEvent me) {
744         }
745
746         @Override
            public void mouseExited(MouseEvent me) {
748         }
749
750     }
751
```