Course 2DJava: 2D-Computer Graphics with Java
Chapter C4: The Animation Project

Ein leeres Fenster

Beschreibung für: Java Version 6, Update 6 und die Eclipse Platform Version 3.3.2.
Starten Sie Eclipse und wählen im Menu:
`File → New → Java Project → Project name: anim1 → Finish`
`File → New → Class → Name: anim1 → Finish`
Ersetzen Sie den Code von `anim1.java` durch:

```
import java.awt.*;
import java.awt.event.*;
import java.awt.geom.*;
import java.util.*;
import javax.swing.*;

class anim1 extends JFrame {
 public static void main(String args[]) { new anim1(); }
 Point p0, p1;
 Vector polygon = new Vector();
 public anim1() {
  super("anim1: Draw an endless Animation");
  setSize(800, 600);
  show();
 }
}
```

`Run → Run`. Das Ergebnis ist ein leeres weißes Fenster mit Überschrift. Schließen Sie dieses Fenster.

Malprogramm mit dynamischem Array

Beenden Sie Ihr Programm `anim1`.
Erweitern Sie innerhalb von `class anim1` den Konstruktor `public anim1()`, schreiben Sie zusätzlich eine `processWindowEvent`-Methode und eine `DrawArea`-Klasse, welche die Maus-Ereignisse behandelt:

```
 public anim1() {
  super("anim1: Draw an endless Animation");
  setSize(800, 600);
  Container p = getContentPane();
  p.setLayout(new BorderLayout());
  p.add("Center", new DrawArea());
  show();
 }
 protected void processWindowEvent(WindowEvent e) {
  super.processWindowEvent(e);
  if (e.getID() == WindowEvent.WINDOW_CLOSING) System.exit(0);
 }

 static class DrawArea extends JComponent
                       implements MouseListener,
                       MouseMotionListener,
                       ActionListener {
  static Graphics g;
  static float zoom       = 1.01f;
  static float cosinus    = (float) Math.cos(Math.PI/180.0);
  static float sinus      = (float) Math.sin(Math.PI/180.0);
  static Color lineColor  = Color.black;
  static Font arial10     = new Font("Arial", 0, 10+2);
  static float myWidth, myHeight;
  static Point2D.Float[] pf;
  Vector polygon = new Vector();
  Point p0, p1;
```

```
  public DrawArea() {
   addMouseListener(this);
   addMouseMotionListener(this);
   setFont(arial10);
   g = getGraphics();
   Dimension size = getSize();
   myWidth  = size.width;
   myHeight = size.height;
   addComponentListener(new ComponentAdapter() {
    public void componentResized(ComponentEvent e) {
     g = getGraphics();
     Dimension size = getSize();
     myWidth  = size.width;
     myHeight = size.height;
     g.clearRect(0, 0, (int) myWidth, (int) myHeight);
    }   //end of componentResized(..)
   });  //end of addComponentListener(..)
  }     //end of DrawArea()

  public void paint(Graphics g) {
   Dimension size = getSize();
   int fontHeight = g.getFont().getSize();
   g.setColor(Color.red);
   g.drawString("Press the left mouse button and move!", size.width/2-50, fontHeight);
  }

  public void mouseClicked(MouseEvent e) {}
  public void mouseEntered(MouseEvent e) {}
  public void mouseExited(MouseEvent e) {}
  public void mouseMoved(MouseEvent e) {}

  public void mousePressed(MouseEvent e) {
   if ((e.getModifiers() & e.BUTTON3_MASK) != 0) return;
   polygon.clear();
   repaint();
   polygon.add(p0 = new Point(e.getX(), e.getY()));
  }

  public void mouseReleased(MouseEvent e) {
   if (polygon.size() < 2) return;
   pf = new Point2D.Float[polygon.size()];
   for (int i=0; i < polygon.size(); i++) {
    Point p = (Point) polygon.get(i);
    pf[i] = new Point2D.Float(p.x, p.y);
   }
   p0 = null;
  }

  public void mouseDragged(MouseEvent e) {
   if (((e.getModifiers() & e.BUTTON3_MASK) != 0) || (p0 == null)) return;
   p1 = new Point(e.getX(), e.getY());
   if (p1.distance(p0) < 10) return;
   Graphics g = getGraphics();
   g.setColor(lineColor);
   g.drawLine(p0.x, p0.y, p1.x, p1.y);
   polygon.add(p1);
   p0 = p1;
  }

  public void actionPerformed(ActionEvent e) {
  }
 }
```

Zeichnen Sie und erproben Sie den Verlust der Zeichnung nach verkleinern.

Timer

Beenden Sie Ihr Programm anim1.
Schreiben Sie in den Kopf von `class DrawArea` unterhalb der Zeile `Point p0, p1;` folgende weitere Zeile:

```
javax.swing.Timer timer = null;
```

Schreiben Sie als **zweiten** Befehl der Funktion `public void mousePressed(MouseEvent e)`:

```
if (timer != null) timer.stop();
```

Schreiben Sie zwei **letzte** Befehle in die Funktion `public void mouseReleased(MouseEvent e)`:

```
if (timer == null) timer = new javax.swing.Timer(1, this);
timer.start();
```

Schreiben Sie folgenden Code in die leere Funktion `public void actionPerformed(ActionEvent e)`:

```
if (e.getSource() != timer) return;
float x, y, xmin, ymin, xmax, ymax, xmid, ymid;
xmin = xmax = pf[0].x;
ymin = ymax = pf[0].y;
for (int i=0; i < pf.length; i++) {
 x = pf[i].x;
 y = pf[i].y;
 if (x < xmin) xmin = x;
 if (x > xmax) xmax = x;
 if (y < ymin) ymin = y;
 if (y > ymax) ymax = y;
}
xmid = (xmin+xmax) / 2;
ymid = (ymin+ymax) / 2;
if ((xmin < 0) || (ymin < 0) || (xmax > myWidth) || (ymax > myHeight)) {
 g.clearRect(0, 0, (int) myWidth, (int) myHeight);
 int fontHeight = g.getFont().getSize();
 g.setColor(Color.red);
 g.drawString("Press the left mouse button and move!", 320, fontHeight);
 zoom = 0.99f;
}
if ((xmax - xmin < 50) || (ymax - ymin < 50)) {
 g.clearRect(0, 0, (int) myWidth, (int) myHeight);
 zoom = 1.01f;
}
for (int i=0; i < pf.length; i++) {
 x = pf[i].x - xmid;
 y = pf[i].y - ymid;
 x *= zoom;
 y *= zoom;
 float xx = x*cosinus - y*sinus;
 float yy = x*sinus + y*cosinus;
 pf[i].x = xx + xmid;
 pf[i].y = yy + ymid;
}
g.setColor(lineColor);
for (int i=0; i < pf.length-1; i++)
 g.drawLine((int) pf[i].x, (int) pf[i].y, (int) pf[i+1].x, (int) pf[i+1].y);
```

Erproben Sie die Animation. Sie können nacheinander beliebig viele Figuren malen.

Weitere Aufgaben

Malen Sie eine Figur in die rechte untere Fensterecke. Ziehen Sie zur Laufzeit am Fensterrand und beobachten Sie das adaptive Verhalten der Animation auf die wechselnde Fenstergröße.
Bremsen Sie die Geschwindigkeit der Animation in
die Funktion `public void mouseReleased(MouseEvent e)` durch hochsetzen des
ersten Parameters beim Aufruf von `new javax.swing.Timer(1, this)` auf 100, 1000.
Erhöhen Sie die Geschwindigkeit der Animation durch Erhöhen der Schrittweite von 1 auf 2, 4, 6 Grad.
Vergrößern Sie die Abbruchbedingungen durch verkleinern (auch negativ !) von `xmin` und `xmax` und vergrößern von `ymin` und `ymax`.
Suchen Sie folgende Begriffe: `Timer, addComponentListener, Math.cos, Math.sin, Math.PI, clearRect` ein.
Mit diesem Wissen wandeln Sie den Code ab und erzeugen Varianten.
Erfinden und erproben Sie neue Varianten des Programms (in Form von neuen Projekten `anim2, anim3` usw. nach obigem Muster).