Course 2DJava: 2D-Computer Graphics with Java
Chapter C2: The Draw 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: draw1 → Finish`
`File → New → Class → Name: draw1 → Finish`
Ersetzen Sie den Code von `draw1.java` durch:

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

class draw1 extends JFrame {
 public static void main(String args[]) { new draw1(); }
 Graphics g;
 Point p0         = new Point();
 Point p1         = new Point();
 public draw1() {
  super("draw1: A Scribble Program");
  setSize(800,600);
  getContentPane().setBackground(Color.white);
  setVisible(true);
 }
}
```

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

Minimales Malprogramm

Schreiben Sie in den Konstruktor unter `setVisible(true);`; folgende 3 Zeilen:

```
  MyMouseAdapter mma = new MyMouseAdapter();
  addMouseListener(mma);
  addMouseMotionListener(mma);
```

Schreiben Sie hinter den Konstruktor `public draw1()`, aber noch vor die letzte geschweifte Klammer:

```
 public void paint(Graphics g){
  super.paint(g);
  g.setColor(Color.red);
  g.drawString("Press the left mouse button and move!", 10, 40 );
 }
class MyMouseAdapter extends MouseAdapter implements MouseMotionListener{
  public void mousePressed(MouseEvent e){
   p0 = e.getPoint();
   repaint();
  }
  public void mouseDragged(MouseEvent e){
   p1 = e.getPoint();
   g = getGraphics();
   g.drawLine(p0.x, p0.y, p1.x, p1.y);
   p0 = p1;
  }
  public void mouseReleased(MouseEvent e){}
  public void mouseMoved(MouseEvent e){}
 }
```

`Run → Run`. Was Sie zeichnen verschwindet, wenn man das Fenster zuerst verkleinert dann vergrößert.

Anzeige der Koordinaten

Beenden Sie das Programm.
Erweitern Sie die Methode `public void mouseDragged(MouseEvent e)` unterhalb der Zeile:
`g = getGraphics();`

```
g.setColor(Color.white);
g.fillRect(10,45,100, 20);
g.setColor(Color.black);
g.drawString("x=" + p1.x + ", y=" + p1.y, 10, 60);
```

Anzeige der Vertices

Vertices (deutsche Aussprache: Wertizehs, englisch: wörtiziis) ist der Plural des lateinischen Vertex = Eckpunkt.
Version3: Beenden Sie das Programm.
Schreiben Sie eine weitere Zeilen in die Funktion `public void mouseDragged(MouseEvent e)` unterhalb die schon vorhandenen Befehle:

```
g.drawRect( p1.x-3, p1.y-3, 7, 7 );
```

Zeichnen Sie schnell und langsam und beobachten Sie, wie die Dichte der Vertices von der Malgeschwindigkeit abhängt.
Langsame Computer und Graphikkarten erzeugen weniger Vertices als schnelle, weil das Betriebssystem die Windows-Message `WM_MouseMove` seltener erzeugt und an den `MouseMotionListener` schickt,
welcher infolgedessen seltener seine Methode `public void mouseDragged(MouseEvent e)` aufruft.
Wichtige Aussage: Das Programm kann keine konstante Anzahl von Vertices erzeugen,
sondern diese Anzahl hängt von der Malgeschwindigkeit und der Hardware ab. Bei langsamen Mausbewegungen und schnellen Rechnern werden offensichtlich viel zu viele Vertices erzeugt.
Es gibt witzige Effekte, wenn Sie die Quadrate größer machen z.B. mit `p1.x-19, p1.y-19, 40, 40`.

Minimalabstand der Vertices

Beenden Sie das Programm.
Schreiben Sie drei weitere Zeilen in `public void mouseDragged(MouseEvent e)` direkt unterhalb des Statements `p1 = e.getPoint();` :

```
if (p1.distance(p0) < 10.0) return;
```

Vertex-Array

Beenden Sie das Programm.
Version5: Das gezeichnete Polygon war bisher offen und es war verloren, wenn das Fenster verkleinert wurde oder wenn es von einem anderen Fenster zeitweilig bedeckt war.
Diese Mängel werden nun behoben dadurch, dass die Vertices laufend in einem Array fester Länge gespeichert werden, dass der letzte mit dem ersten Vertex automatisch verbunden wird und dass bei jeder Fensterneuerung alles automatisch neu gezeichnet wird.
Schreiben Sie folgende Zeilen in den Kopf von `class draw1 extends JFrame`
unterhalb der Zeile `Graphics g`:

```
int i, n;
 final int nMax = 100;
 Point[] polygon  = new Point[nMax];
```

Ändern Sie die Funktion `public void mousePressed(MouseEvent e)`, dass sie so aussieht:

```
  public void mousePressed(MouseEvent e){
   polygon[0] = p0 = e.getPoint();
   n = 1;
   repaint();
  }
```

Ändern Sie die Funktion `public void mouseDragged(MouseEvent e)`, dass sie so aussieht:

```
  public void mouseDragged(MouseEvent e){
   p1 = e.getPoint();
   if (p1.distance(p0) < 10.0) return;
   if ( n >= nMax-1 ) return;
   Graphics g = getGraphics();
   g.drawLine(p0.x, p0.y, p1.x, p1.y);
   g.drawRect(p1.x-3, p1.y-3, 7, 7 );
   g.setColor(Color.white); g.fillRect(0,60,100,20); g.setColor(Color.black);
   g.drawString(" x=" + p1.x + " y="+ p1.y, 10, 80);
   p0 = polygon[n++] = p1;
  }
```

Ändern Sie die bisher leere Funktion `public void mouseReleased(MouseEvent e)`, dass sie so aussieht:

```
  public void mouseReleased(MouseEvent e){
   if ( n < 2 ) return;
   polygon[n++] = p0 = polygon[0]; //close it
   repaint();
  }
```

Ändern Sie die Funktion `public void paint(Graphics g)`, dass sie so aussieht:

```
 public void paint(Graphics g) {
  super.paint(g);
  g.setColor(Color.red);
  g.drawString("Press the left mouse button and move!", 10, 40 );
  for ( i=0; i < n-1; i++ )
   g.drawLine(polygon[i].x, polygon[i].y, polygon[i+1].x, polygon[i+1].y );
 }
```

Erproben Sie das Programm durch
1.) ziehen am Fensterrand: extrem verkleinern und zurückvergrößern und
2.) durch überdecken und abdecken mit irgend einer anderen Windows-Applikation.
Verkürzen und verlängern Sie den Array durch Änderung von `nMax` auf `10` und `200` und erproben Sie die Auswirkungen.

Umfang, Fläche, Schwerpunkt, umschreibendes Rechteck

Version6: Beenden Sie Ihr Programm `draw1`.
Schreiben Sie eine weitere Deklaration in den Kopf von `class draw1 extends JFrame` unterhalb der Zeile `Point p1 = new Point();`:

```
 Point mid_of_p   = new Point();
 Point mid_of_r   = new Point();
 Rectangle minmax = new Rectangle();
 double perimeter, area;
```

Ändern Sie die Funktion `public void mouseReleased(MouseEvent e)`, dass sie so aussieht:

```
  public void mouseReleased(MouseEvent e){
   if ( n < 2 ) return;
   polygon[n++] = p0 = polygon[0];//close the polygon
   perimeter = area = 0;
   mid_of_p.x = mid_of_p.y = 0;
   int xmin, xmax, ymin, ymax;
   xmin = xmax = p0.x;
   ymin = ymax = p0.y;
   for ( i = 1; i < n; i++ ){
    p1 = polygon[i];
    double dx = p1.x - p0.x;
    double dy = p1.y - p0.y;
    double my = (p0.y + p1.y) / 2.0;
    perimeter  += Math.sqrt( dx*dx + dy*dy );
    area       += dx * my;
    mid_of_p.x += p1.x;
    mid_of_p.y += p1.y;
    if ( p1.x < xmin ) xmin = p1.x;
    if ( p1.x > xmax ) xmax = p1.x;
    if ( p1.y < ymin ) ymin = p1.y;
    if ( p1.y > ymax ) ymax = p1.y;
    p0 = p1;
   }
   mid_of_r.x = ( xmax + xmin ) / 2;
   mid_of_r.y = ( ymax + ymin ) / 2;
   mid_of_p.x /= n-1;
   mid_of_p.y /= n-1;
   minmax.x = xmin-1; minmax.width  = xmax - xmin + 2;
   minmax.y = ymin-1; minmax.height = ymax - ymin + 2;
   repaint();
  }
```

Ändern Sie die Funktion `public void paint(Graphics g)`, dass sie so aussieht:

```
 public void paint(Graphics g){
  super.paint(g);
  g.setColor(Color.red);
  g.drawString("Press the left mouse button and move!", 10, 40 );
  if ( n < 2 ) return;
  String myline = "Perimeter= "+(int)perimeter+", Area= "+(int)area;
  g.setColor(Color.black);
  g.drawString( myline, 10, 80);
  g.setColor(Color.green);
  g.drawRect(minmax.x, minmax.y, minmax.width, minmax.height);
  g.drawLine(mid_of_r.x - 4, mid_of_r.y    , mid_of_r.x + 4, mid_of_r.y     );
  g.drawLine(mid_of_r.x    , mid_of_r.y - 4, mid_of_r.x    , mid_of_r.y + 4 );
  g.setColor(Color.black);
  g.fillOval(mid_of_p.x-5, mid_of_p.y-5, 11, 11 );
  for ( i=0; i < n-1; i++ )
   g.drawLine(polygon[i].x,polygon[i].y, polygon[i+1].x,polygon[i+1].y );
 }
```

Erproben Sie das Programm und erarbeiten Sie sich den Zusammenhang zwischen dem Code in `mouseReleased(..) / paint(..)` und dem was Sie im Ergebnis sehen. Beachten Sie, wie das Vorzeichen von `Area` von der Umlaufrichtung abhängt.