Fouriersynthese mit Java

Ein Programmierbeispiel für Java ab Version 1.1

Martin Schönhacker

Ein Programmierbeispiel macht noch keinen Programmierkurs, und ein kurzes Beispiel ist umso schlimmer, aber für Interessenten sollte es sollte hoffentlich möglich sein, am folgenden kommentierten Quellcode zu erkennen, wie einfach auch etwas komplexer aussehende Aufgaben mit Hilfe der "Internet-Programmiersprache" Java gelöst werden können.

Verwendet wurde das Java Development Kit (JDK) in der Version 1.2, aber tatsächlich kann jede Version ab 1.1 zum Einsatz kommen. Das Entwicklungssystem ist kostenlos verfügbar, entweder vom Hersteller Sun (*http://java.sun.com/*) oder in Österreich zum Beispiel in der "Goodie Domain" an der TU Wien (*http://gd.tuwien.ac.at/languages/java/distrib/jdk1.2/* für Version 1.2).

Es wird in der Folge bewusst nicht auf Details der Programmierung eingegangen. Wer sich für die genauere Funktionsweise des Programms interessiert, dem sei die umfassende Online-Dokumentation des JDK ans Herz gelegt. Mit dem Entwicklungssystem werden auch zahlreiche weitere Beispiele geliefert, die durchzuarbeiten sich durchaus lohnen kann.

Das nachfolgende Programm soll die Überlagerung mehrerer Sinuswellen erlauben. Die Frequenzen dieser Wellen sind vorgegeben, die Amplituden können vom Benutzer eingegeben werden. Außerdem kann jeder einzelne Summand über eine Checkbox dazu- oder weggeschaltet werden. Wegen der Platzbeschränkung ist das Applet nicht auf alle Sonderfälle eingerichtet, die auftreten können, wie zum Beispiel auf eine Änderung der Anzeigefläche. Das sei interessierten Leser/inne/n zur Weiterentwicklung überlassen, denn es handelt sich potentiell um ein Fass ohne Boden.

Um dieses Programm möglichst flexibel zu halten, wird die Anzahl der zu überlagernden Sinusschwingungen in einer Konstanten `NUM` gehalten. Das bedeutet, dass auch die Anzahl der Textfelder und Checkboxen von `NUM` abhängt. Daher enthalten zwei globale Arrays die entsprechenden Textfelder `s[]` und Checkboxen `c[]`.

```
import java.applet.Applet;
import java.awt.*;
import java.awt.event.*;

public class Fourier extends Applet
  implements ActionListener, ItemListener
{
     static final int NUM = 10;
     TextField s[];
     Checkbox c[];
```

In der Methode `init()` werden diese Felder nun initialisiert. Wir wollen alle Eingaben am oberen Rand des Applets stehen haben, damit die untere Hälfte für die Graphikausgabe genutzt werden kann. Der Layoutmanager versucht aber stets, die Komponenten so groß wie möglich zu machen. Um dies zu verhindern, müssen wir alle Textfelder und Checkboxen zuerst in einen Bereich schreiben, der nur einem eigenen Layout versehen werden kann, und dieser Bereich soll dann dem Applet hinzugefügt werden (der Standard Layout Manager wird ihn dann zentriert oben erscheinen lassen).

```
     public void init()
     {
          Panel pan;     // Bereich für Textfelder und Checkboxen

          int i;
          s = new TextField[NUM];
          c = new Checkbox[NUM];
          pan = new Panel();  // Initialisierung des Bereichs

          pan.setLayout(new GridLayout(5,2,5,10));
```

Zuweisen der Initialisierungswerte an Checkboxen und Textfelder, die als Array abgespeichert werden:

```
          for (i = 0; i < NUM; i++)
          {
               // Checkboxen werden voreinstellungsmäßig
               // nicht angewählt
               c[i] = new Checkbox("Sinus("+(i + 1)+"x):",null,false);
               pan.add(c[i]);
               c[i].addItemListener(this);

               // Textfelder erhalten den Wert 1.0
               s[i] = new TextField("1.0");
               pan.add(s[i]);
               s[i].addActionListener(this);
          }
          // Der Bereich muss dem Applet hinzugefügt werden.
          add(pan);
     }
```

Einen solchen Bereich, wie wir ihn verwendet haben, nennt man **Panel**. Zunächst erzeugen wir ein neues Panel `pan` und initialisieren es. Dann wird diesem Panel ein `GridLayout` zugeteilt. In einer `for`-Schleife werden nun beschriftete Checkboxen und Textfelder zur Eingabe der Amplitude dem Panel hinzugefügt. Am Ende wird dieses Panel noch dem Layout des Applets hinzugefügt.

Immer, wenn der Benutzer eine Eingabe tätigt, verändert er die Parameter für die Ergebniskurve; die Graphik muss also jedesmal aktualisiert werden. Daher veranlassen die Methoden `itemStateChanged()` und `actionPerformed()` nur das Neuzeichnen der Graphik (die alte Zeichnung muss in der Methode `paint()` durch ein mit der Hintergrundfarbe gefülltes Rechteck gelöscht werden):

```
     public void itemStateChanged(ItemEvent e)
     {
          repaint();
     }

     public void actionPerformed(ActionEvent e)
     {
          repaint();
     }

     public void update (Graphics g)
     {
          paint(g);
     }
```

In der Methode `paint()` muss nun die Eingabe des Benutzers ausgewertet werden und die entsprechende Funktion ausgerechnet und angezeigt werden. Dazu werden zunächst die Amplituden aus den Textfeldern eingelesen, in `double`-Werte umgewandelt und im Array `a` gespeichert. Danach werden alle Amplituden der eingeschalteten Summanden aufaddiert und das Ergebnis in `d` gespeichert.

Die Variable `d` repräsentiert die maximale Amplitude, die im Ergebnis auftreten kann. Dieser Wert wird zur Skalierung der Graphik benötigt. Danach wird die alte Graphik durch ein weißes Rechteck (unsere Hintergrundfarbe ist weiß) gelöscht. Für jeden x-Wert der Funktion werden nun die erforderlichen Summanden aufaddiert und das Ergebnis durch `d` dividiert. Letztendlich zeichnet der Befehl `drawLine()` die resultierende Funktion.

Martin Schönhacker schoenhacker@apm.tuwien.ac.at

```
        public void paint (Graphics g)
        {
             int x, y, yAlt, i;
             double a[], d;
             a = new double [NUM];

             // Einlesen der Textfelder in a
             for (i = 0; i < NUM; i++)
                  a[i] = Double.valueOf(s[i].getText()).doubleValue();

             // Maximale Amplitude ermitteln (Skalierung)
             d = 0;
             for (i = 0; i < NUM; i++)
                  if (c[i].getState())
                       d = d+Math.abs(a[i]);

             // Hintergrund löschen
             yAlt = 0;
             g.setColor(getBackground());
             g.fillRect(0,200,500,500);
             g.setColor(getForeground());

             // Neue Kurve zeichnen
             for (x = 1; x < 500; x++)
             {
                  y = 0;
```

Der Gesamtwert der Auslenkung wird, abhängig vom Status der Checkboxen, ermittelt:

```
                  for (i = 0; i < NUM; i++)
                       if (c[i].getState())
                            y = y + (int)(150 * a[i] *
                                 Math.sin((double)(x * (i + 1))
                                 / 250 * Math.PI));
```

Anpassung der ermittelten Werte an die Skalierung und Ausgabe der entsprechenden Werte:

```
                  if (d != 0)
                       y = (int) ((double) y / d);

                  g.drawLine(x - 1,
                     350 - yAlt, x, 350 - y);
                  yAlt = y;
             }
        }
}
```

Die Einbindung in eine Internet-Seite erfolgt über den folgenden HTML-Code, wobei die Angaben für `WIDTH` und `HEIGHT` nicht verändert werden sollten:

```
<HTML>
<TITLE>Fourier als Applet</TITLE>

     <APPLET CODE="Fourier.class" WIDTH=500 HEIGHT=500>
     </APPLET>
</HTML>
```

Diese Seite kann jetzt mit Hilfe des Applet Viewers aus dem JDK angezeigt werden. Aktuelle Browser haben mit Java ab Version 1.1 noch ihre kleinen und großen Probleme, aber wenn dieser Artikel erscheint, wird sich das vielleicht auch schon gebessert haben. Ansonsten gibt es ein Plug-In von Sun, das auch dieses Problem behebt.

Damit bleibt nur noch, viel Spaß beim Entwickeln eigener Programme zu wünschen, falls Sie anhand dieses kleinen Beispiels auf den Geschmack gekommen sind. Gutes Gelingen!