Java

Iterativ berechnete Fraktale

Alfred Nussbaumer

Mit Java können vielgestaltige Fraktale rekursiv erzeugt werden (vgl. **PCNEWS**-76). Zwei äußerst bekannte Fraktale, das Apfelmännchen und die Julia-Menge werden allerdings mit einfachen Algorithmen iterativ berechnet, indem komplexe Zahlen quadriert und addiert werden. Zusätzlich soll im zweiten Beispiel eine einfache Benutzerschnittstelle vorgestellt werden.

1. Das Apfelmännchen (mandel.java)

... ist eigentlich die so genannte Mandelbrot-Menge (genannt nach dem Mathematiker B.B. Mandelbrot, der sie in der 2. Hälfte des 20. Jahrhunderts untersuchte). Den Namen "Apfelmännchen" hat diese Menge wegen ihrer charakteristischen Form erhalten. Das Gebilde ist ein Fraktal, das durch eine rekursive definierte Folge von komplexen Zahlen entsteht:

$z(n+1) = z(n)^2+c \qquad z(0) = (0 , 0)$

Stellen wir die komplexe Zahl $z = (x, y)$ mit Realteil x und Imaginärteil y dar, und sind a und b Real- und Imaginärteil der komplexen Zahl $c = (a, b)$ so erhalten wir die obige Definition in der Form:

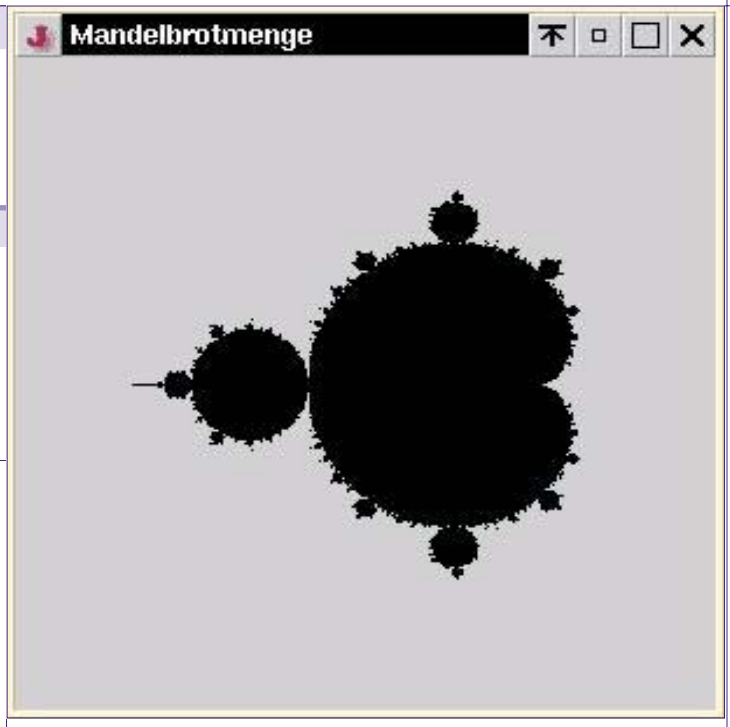
$(x , y) \rightarrow (x^2 - y^2 , 2*x*y) + (a , b)$

Wir untersuchen das Verhalten des Koordinatenursprungs $z = (0,0)$ unter der angegebenen Iteration. Konvergiert er für einen Parameter (a,b) der komplexen Zahlenebene bei der angegebenen Rekursion, so ist dieser ein Element der Mandelbrot-Menge. Der "konvergente" Punkt wird beispielsweise schwarz eingefärbt. Strebt die Rekursion für den gewählten Parameter gegen Unendlich, so wird er "verworfen".

Anschließend wird der nächste Punkt untersucht.

Die Gleichung lösen wir also koordinatenweise auf und kodieren eine entsprechende Iteration über alle Punkte eines gewählten Quadrates der komplexen Zahlenebene. Das Quadrat wird durch den "Eckpunkt" (`amin`, `bmin`) und durch seine Kantenlänge `k` festgelegt. Zusätzlich müssen die Punkte der komplexen Zahlenebene auf ein Bildschirmquadrat (im untenstehenden Programm 200x200 Pixel) "gezoomt" werden. Dazu dividieren wir die Kantenlänge des Quadrats durch 200 (Zahl der Pixel,Schrittweite `ds=k/200`) und erhöhen bei jeder Iteration den Realteil bzw. den Imaginärteil der komplexen Zahl um diese Schrittweite `ds`:

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

public class mandel extends Frame {
  public static void main(String arguments[]) {
    mandel proggi = new mandel();
    WindowListener wl = new WindowAdapter() {
      public void windowClosing(WindowEvent e) {
          System.exit(0);
      }
    };
    proggi.addWindowListener(wl);
    proggi.setLocation(100,100);
    proggi.setSize(300,300);
    proggi.show();
  }

  public mandel() {
    super("Mandelbrotmenge");
  }
  public void paint (Graphics bs) {
    double amin;
    double bmin;
    double kante;
    double ds;
    double a;
    double b;
    double x;
    double y;
    double xx;
    double yy;
    int s;
    int r;
    int zaehler;
// Startwerte
    amin=-1.5;
    bmin=-1;
    kante=2;
    ds=kante/200;
    a=amin;

// horizontaler Wert gewählt
    for (s=0; s<=200; s++) {
      b=bmin;
// vertikaler Wert gewählt
      for (r=0; r<=200; r++) {
      x=0;
      y=0;
      zaehler=0;
// Iteration am gewählten Punkt
      while ((zaehler < 100) &&
              (Math.sqrt(x*x+y*y)<2)) {
        zaehler++;
        xx=x*x-y*y+a;
        y=2*x*y+b;
        x=xx;
      }
// Wie verhält sich der Punkt beim Iterieren?
      if (zaehler == 100) {
        bs.setColor(Color.black);
        bs.drawLine(s+50,r+50,s+50,r+50);
      }
// nächsten Punkt in der Spalte wählen
      b=b+ds;
      }
// nächste Spalte wählen
      a=a+ds;
    }
  }
}
```



Die "eigentlichen Programmzeilen" stehen also in der Methode `paint()`, die beim Öffnen des Programmes ausgeführt wird.

Interessanterweise sind die "Fortsätze" des "Apfelmännchens" selbst wieder von der Gestalt eines "Apfelmännchens", wenngleich sie deutlich kleiner sind. Diese "Selbstähnlichkeit" legt nahe, dass das erhaltene Gebilde ein Fraktal darstellt.

Um kleinere Ausschnitte des Fraktales darstellen zu können, ist es notwendig, die Eckpunktskoordinaten (`amin`, `bmin`) des Quadrates und seine Kantenlänge `kante` neu zu wählen. Das Inkrement ds für die Iteration über alle Spalten bzw. über alle Punkte der Spalten wird so berechnet, dass der gewählte Ausschnitt auf einem 200 Pixel x 200 Pixel großen Bereich des Bildschirmes dargestellt wird.

2. Die Julia-Menge (julia.java)

Der französische Mathematiker Gaston Julia hat zu Beginn des 20. Jahrhunderts eine bestimmte Menge von komplexen Zahlen untersucht. Diese komplexen Zahlen haben die Eigenschaft, dass sie bei der Iteration

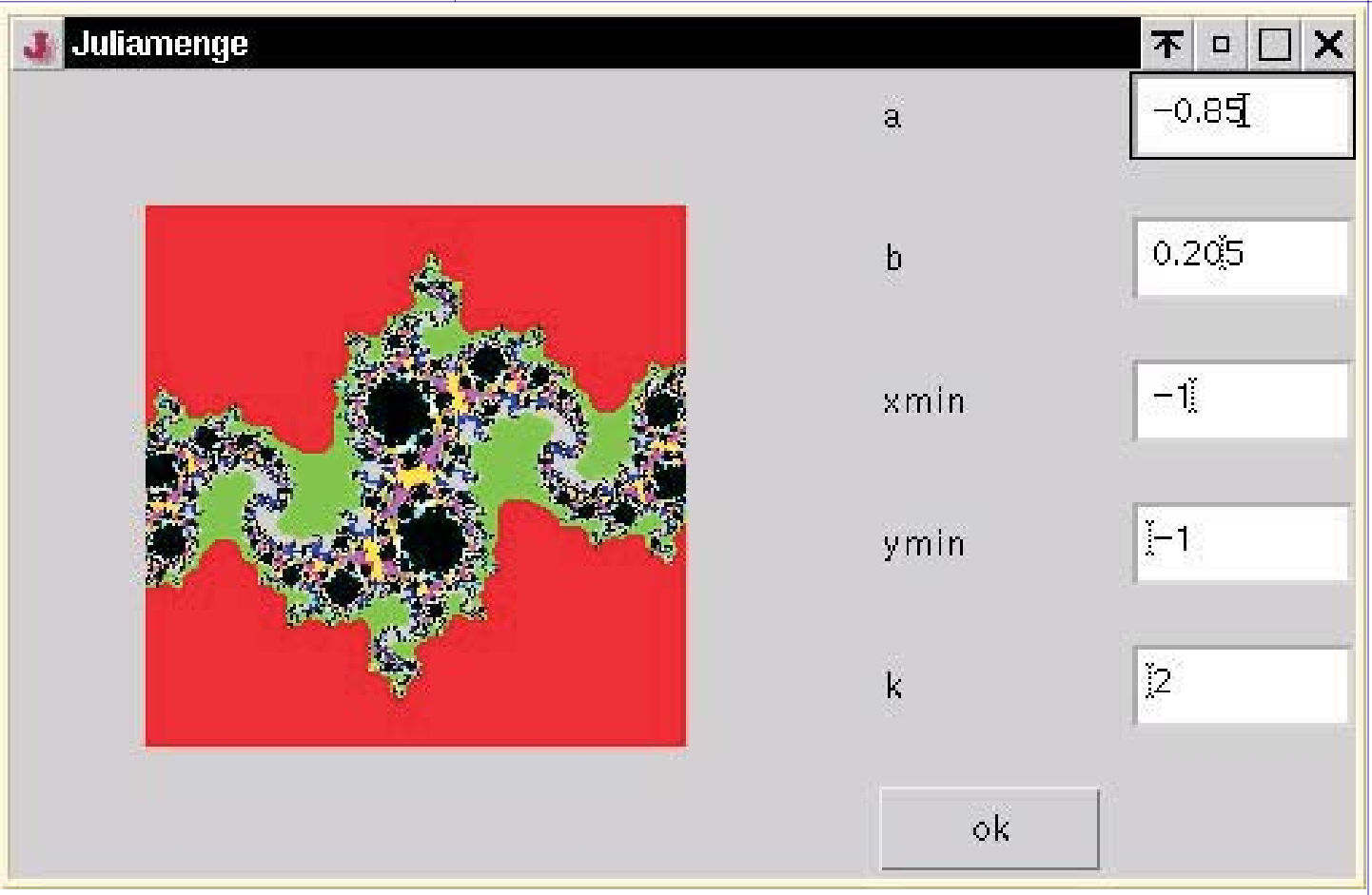
$z(n+1) = z(n)^2 + c$

konvergieren. Stellt man die komplexen Zahlen, die diese Eigenschaft aufweisen, farblich dar, so erhält man komplizierte Flächenmuster, deren Gestalt von der Wahl der Konstanten c abhängt. Man sich leicht davon über-zeugen, dass eine geringfügig anders gewählte Konstante c zu einer deutlich anderen Julia-Menge führt.

Ähnlich wie bei der Mandelbrot-Menge ***(siehe vorige Seite)*** wählen wir ein Quadrat aus der komplexen Zahlenebene aus (Eckpunkt (`xmin`, `ymin`), Kantenlänge `k`) und geben eine Konstante `c = (a, b)` an. Das (im Allgemeinen relativ kleine) Quadrat der komplexen Zahlenebene wird auf einen größeren Bildschirmbereich "gezoomt" (im Beispiel auf 200x200 Pixel). Daraus bestimmen wir die Schrittweite `ds=k/200` für die Iterationen.

Ist ein Punkt Element der Julia-Menge (verhält er sich bei der Iteration konvergent), wird er schwarz eingefärbt. Um die "divergenten" Punkte klassifizieren zu können, untersucht man, wie "rasch" sie gegen Unendlich streben. Weist die komplexe Zahl schon nach weniger als 10 Iterationen einen Betrag größer als 2 auf, so wird der zugehörige Punkt rot eingefärbt. Verlässt der Punkt den Kreis mit Radius 2 erst nach mehr als 10 aber mit weniger als 20 Iterationen, so zeichnen wir ihn grün, usf. Dies ist im Programmcode durch eine Reihe von if-Abfragen realisiert.

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

public class julia extends Frame
              implements ActionListener {

  TextField aein;
  TextField bein;
  TextField xminein;
  TextField yminein;
  TextField kein;
  Button ok;

  double a;
  double b;
  double xmin;
  double ymin;
  double k;

  public static void main(String arguments[]) {
    julia proggi = new julia();
    WindowListener wl = new WindowAdapter() {
      public void windowClosing(WindowEvent e) {
         System.exit(0);
      }
    };
    proggi.addWindowListener(wl);
    proggi.setLocation(100,100);
    proggi.setSize(500,300);
    proggi.show();
  }

  public julia() {
    super("Juliamenge");
    setLayout(new BorderLayout());
    Panel panel = new Panel();
    panel.setLayout(new GridLayout(6,2,10,20));
    Label l1 = new Label("a");
    panel.add(l1);
    aein = new TextField("-1",8);
    panel.add(aein);
    Label l2 = new Label("b");
    panel.add(l2);
    bein = new TextField("0.28",8);
    panel.add(bein);
    Label l3 = new Label("xmin");
    panel.add(l3);
    xminein = new TextField("-1",8);
    panel.add(xminein);
    Label l4 = new Label("ymin");
    panel.add(l4);
    yminein = new TextField("-1",8);
    panel.add(yminein);
    Label l5 = new Label("k");
    panel.add(l5);
    kein = new TextField("2",8);
    panel.add(kein);
    ok = new Button ("ok");
    panel.add(ok);
    ok.addActionListener(this);
    add("East",panel);
  }

  public void actionPerformed(ActionEvent e) {
    if (e.getSource() == ok) {
      werteuebernehmen();
      repaint();
    }
  }

  public void werteuebernehmen() {
      a = Double.valueOf(aein.getText()).doubleValue();
    b = Double.valueOf(bein.getText()).doubleValue();
      xmin = Double.valueOf(xminein.getText()).doubleValue();
    ymin = Double.valueOf(yminein.getText()).doubleValue();
    k = Double.valueOf(kein.getText()).doubleValue();
  }

  public void paint (Graphics bs) {
    double ds;
    double x;
    double y;
    double x1;
    double y1;
    double xx;
    int s;
    int r;
    int zaehler;

    werteuebernehmen();

    ds = k/200;
    x = xmin;
    for (s=0; s<=200; s++) {
      y=ymin;
      for (r=0; r<=200; r++) {
      x1=x;
      y1=y;
      zaehler=0;
      while ((zaehler < 100) &&
          (Math.sqrt(x1*x1+y1*y1)<2)) {
        zaehler++;
        xx=x1*x1-y1*y1+a;
        y1=2*x1*y1+b;
        x1=xx;
      }
     if (zaehler >= 100) {
        bs.setColor(Color.black);
        bs.drawLine(s+50,r+50,s+50,r+50);
     }
     if (zaehler < 10) {
        bs.setColor(Color.red);
        bs.drawLine(s+50,r+50,s+50,r+50);
     }
     else if (zaehler < 20) {
        bs.setColor(Color.green);
        bs.drawLine(s+50,r+50,s+50,r+50);
     }
     else if (zaehler < 30) {
        bs.setColor(Color.lightGray);
        bs.drawLine(s+50,r+50,s+50,r+50);
     }
     else if (zaehler < 40) {
        bs.setColor(Color.blue);
        bs.drawLine(s+50,r+50,s+50,r+50);
     }
     else if (zaehler < 50) {
        bs.setColor(Color.yellow);
        bs.drawLine(s+50,r+50,s+50,r+50);
     }
     else if (zaehler < 60) {
        bs.setColor(Color.magenta);
        bs.drawLine(s+50,r+50,s+50,r+50);
     }
     else if (zaehler < 70) {
        bs.setColor(Color.gray);
        bs.drawLine(s+50,r+50,s+50,r+50);
     }
     else if (zaehler < 80) {
        bs.setColor(Color.orange);
        bs.drawLine(s+50,r+50,s+50,r+50);
     }
     else if (zaehler < 90) {
        bs.setColor(Color.white);
        bs.drawLine(s+50,r+50,s+50,r+50);
     }
     else if (zaehler <100) {
```

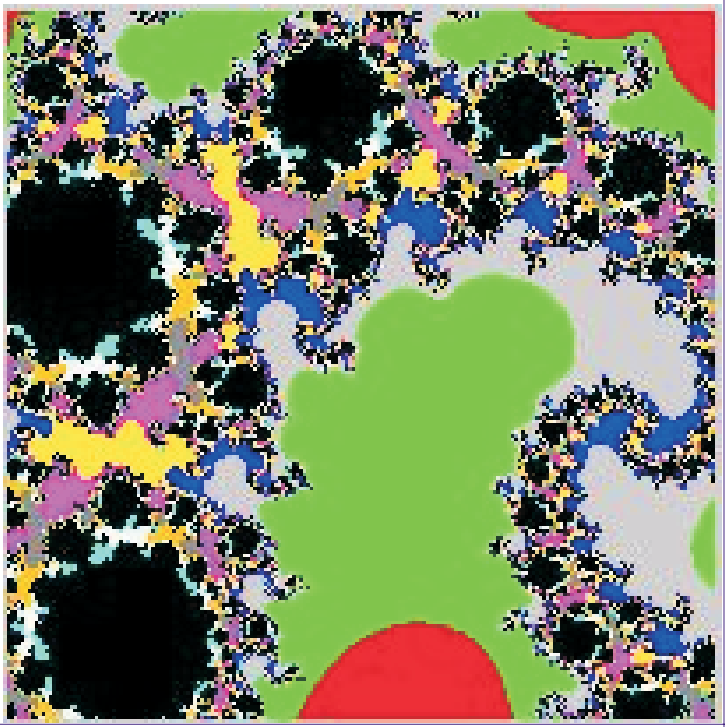



```
            bs.setColor(Color.cyan);
            bs.drawLine(s+50,r+50,s+50,r+50);
        }
       y=y+ds;
        }
        x=x+ds;
      }
    }
}
```

Die schwarz codierten Punkte stellen die Julia-Menge dar. Ob bei der Iteration eine zusammenhängende Julia-Menge entsteht oder nicht, hängt davon ab, ob der Parameter c ein Element der Mandelbrotmenge ist. ***(Bild oben)***

Durch die Wahl geeigneter Parameter, kann die Julia-Menge für kleinere Ausschnitte berechnet werden. Dabei erkennt man die Selbstähnlichkeit der Strukturen. Darüber hinaus sind mehr oder weniger stark "verzerrte" Apfelmännchen zu erkennen... ***(Bild rechts)***

Die Julia-Menge mit den Parametern `a=-0.85` und `b=0.205` wurde für den Ausschnitt der komplexen Zahlenebene mit `xmin=-1` und `ymin=-0.2`, Kantenlänge `k=0.5` berechnet.



Anmerkungen zum Programmcode

Das Programmbeispiel zur Julia-Menge zeigt die Verwendung einer ActionListener-Schnittstelle. Diese definiert die Methode actionPerformed, die beim Eintreten eines Ereignisses aufgerufen wird (im Programm das ***"Drücken der OK-Schaltfläche"***): In der Methode `werteuebernehmen()` holt die Methode `getText()` die Eingaben aus den Textfeldern. Methoden der Klasse `Double` liefern schließlich die Werte für die Konstante `c = (a,b)` und für den Ausschnitt der komplexen Zahlenebene ***(vgl. PCNEWS-74)***.

Im Konstruktor sind der Titel des verwendeten Frames und die Anordnung der einzelnen Elemente durch Layouts festgelegt: Die GUI-Elemente Label, TextField und Button werden mit Hilfe eines so genannten "`GridLayout()`" in Reihen und Spalten platziert. Das `GridLayout()` ist selbst an ein Panel gebunden, das in den "East"-Teil eines `BorderLayout()` eingefügt wurde. Für das Erstellen professioneller Benutzerschnittstellen ist jedenfalls die Verwendung einer Java-IDE (z.B. Borlands JBuilder) zu empfehlen ;-)