Ameisen und Turmiten

Java

Alfred Nussbaumer

Die Programmiersprache JAVA wird in zahlreichen ausgezeichneten Büchern beschrieben. In ihnen finden sich viele Beispiele, die die Sprachkonstrukte und Details von JAVA erläutern. Grundthemen der Informatik kommen dabei naturgemäß immer zu kurz. Dieser Artikel soll dazu ergänzend ein faszinierendes Objekt in einigen Varianten behandeln: Turing-Maschinen werden als Grundlage für einige Applikationen und Applets besprochen.

1. Grundlagen

Alan M. Turing, einer der Begründer der modernen Rechnertheorie, entdeckte die nach ihm benannte Turing-Maschine. Dabei handelt es sich um eine universelle Rechenmaschine: Im Prinzip wird dabei ein Lesekopf über eine Datei gesteuert. Steht der Lesekopf still, so liest er aus der Datei den nächsten Steuerbefehl, den er sogleich ausführt. Man kann zeigen, dass alle Rechenprozesse prinzipiell durch eine Turingmaschine realisiert werden können. Die Beispiele in diesem Artikel sind gewissermaßen zweidimensionale Turingmaschinen.

2. Die Langton-Ameisen

Natürlich handelt es sich bei diesen Tierchen um virtuelle; in der Literatur tragen sie auch den Namen "vants" (*virtual ants*). Sie wurden von Chris Langton angegeben und Mitte der Neunzigerjahre von Jan Stewart ***[1]*** publiziert. Eine solche Ameise sitzt anfänglich auf einem unendlich großen Zeichenfeld, das in lauter weiße Quadrate eingeteilt ist. Diese Quadrate werden nun von der Ameise umgefärbt, sobald sie sie betritt. Dabei bewegt sich die Ameise nach dem folgenden Schema: Kommt sie auf ein weißes Feld zu stehen, so färbt sie das Quadrat ein, wendet sich um 90° nach rechts und wandert in der neuen Richtung um ein Quadrat weiter. Ist dieses bereits eingefärbt, so färbt sie es wieder weiß, dreht sich um 90° nach links und wandert dann ein Feld weiter. Mit einem Blatt kariertem Papier, einem Radiergummi und einem nicht zu harten Bleistift lassen sich die ersten Schritte der Langton-Ameise "per Hand" nachvollziehen.

Der Algorithmus bestimmt die Bahn der Ameise im Sinn einer Turingmaschine vollständig. Während die Ameise zunächst in einem heillosen Durcheinander über das Zeichenfeld zu sausen scheint, wird die Bewegung nach einer endlichen Zahl von Schritten regelmäßig:

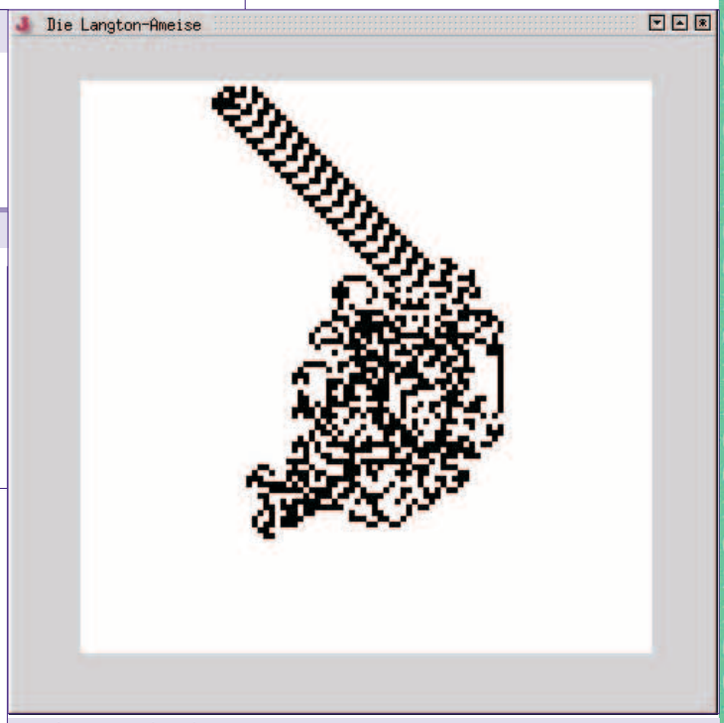
Die Bahn der Langton-Ameise wird durch zwei von der jeweiligen Feldfarbe abhängige Regeln festgelegt, die wir zur Regelkette **"RL"** zusammenfassen. Dabei steht **"R"** für die Drehung nach rechts und **"L"** für die Drehung nach links. Es ist günstig, diese Regeln binär auszudrücken: Statt **"RL"** wählen wir die Bitfolge **"10"**.

3. Der Programmcode für die 10-Ameise

Die aktuelle Position der Ameise wird durch das Koordinatenpaar (x/y) beschrieben. Der Farbstatus wird in einem entsprechend großen, 2-dimensionalen Array gespeichert. Die Bewegung der Ameise nach den vier "Himmelsrichtungen" wird durch folgende Zahlen beschrieben:

1	Bewegung nach Osten (inkrementiere x)
2	Bewegung nach Süden (inkrementiere y)
3	Bewegung nach Westen (dekrementiere x)
4≡0 (mod 4)	Bewegung nach Norden (dekrementiere y)

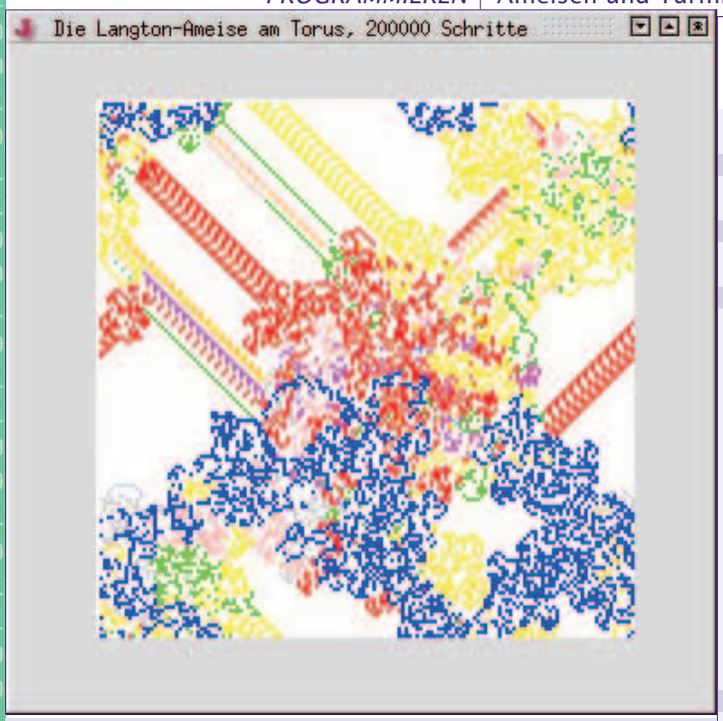
Der gesamte Algorithmus ist innerhalb der `paint()` - Methode enthalten. Die einzelnen Quadrate werden als ausgefüllte Rechtecke mit der Seitenlänge von 4 Pixeln ausgegeben. Die Berechnung bricht ab, wenn eine Koordinate den Gültigkeitsbereich für die Array-Indizes verlässt.



Langton-Ameisen

```
import java.awt.*;
import java.awt.event.*;
public class Langton extends Frame {
  Langton() {
    super("Die Langton-Ameise");
  }
  public static void main (String[] arguments) {
    Langton proggi = new Langton();
    WindowListener wl = new WindowAdapter() {
      public void windowClosing(WindowEvent e) {
        System.exit(0);
      }
    };
    proggi.addWindowListener(wl);
    proggi.setSize(500,500);
    proggi.show();
  }
  public void paint(Graphics bs) {
    int x;
    int y;
    int i;
    int j;
    int richtung;
    int[][] welt = new int [100][100];
    for (i=0;i<100;i++)
      for (j=0;j<100;j++) welt[i][j]=0;
    bs.setColor(Color.white);
    bs.fillRect(50,50,400,400);
    richtung = 1;
    x = 50;
    y = 50;
    x++;
    while ((x>=1) && (x<=100) && (y>=1) && (y<=100)) {
      if (welt[x][y]==1) {
        richtung = ((richtung - 1) % 4);
        welt[x][y]=0;
        bs.setColor(Color.white);
        bs.fillRect(4*x+50,4*y+50,4,4);
      }
      else {
        richtung = ((richtung + 1) % 4);
        welt[x][y]=1;
        bs.setColor(Color.black);
        bs.fillRect(4*x+50,4*y+50,4,4);
      }
      if (richtung == 0) {
        richtung=4;
        y--;
      }
      else if (richtung == 1) x++;
      else if (richtung == 2) y++;
      else if (richtung == 3) x--;
    }
  }
}
```

alfred.nussbaumer@schule.at



Langton-Ameise am Torus

4. Die Langton-Ameise am Torus

Anstelle der begrenzten Zeichenfläche kann man die Bewegung der Ameise auf einem Torus untersuchen. Dazu müssen wir die Koordinaten ersetzen, wenn diese den Randbereich der Fläche erreicht haben:

```
...
        if (x==150) x=1;
        if (y==150) y=1;
        if (x==0) x=149;
        if (y==0) y=149;
...
```

Der Benutzer kann die Anzahl der Rechenschritte mit einem Befehlszeilenparameter festlegen:

```
alfred@duron:~/java> java LangtonTorus 200000
```

Die Langton-Ameise läuft "kreuz und quer" über den Torus. Um die Bahnen besser auseinanderhalten zu können färben wir die Punkte jeweils nach einem Zehntel der gewählten Rechenschritte in einer anderen Farbe. Wie aus der obenstehenden Abbildung ersichtlich ist, bilden sich immer wieder "Autobahnen".

5. Die verallgemeinerte Langton-Ameise

Die Regelkette **"RL"** lässt sich beispielsweise zur Regelkette **"RRLL"** verallgemeinern, wobei man nun vier verschiedene Farben wählt. Ordnet man die Farben der Reihe nach mit den Indizes i an, lassen sich die Regeln folgendermaßen formulieren:

Die Ameise ersetzt die i-te Farbe eines Feldes durch die i+1-te Farbe. Statt der 4. Farbe wählt sie die erste.

Die Drehrichtung wird durch die Regelkette **"RRLL"** angegeben: Gelangt die Ameise auf ein Feld, das die 1. oder 2. Farbe trägt, wendet sie sich nach rechts, und sonst nach links.

Diesmal überrascht uns die Langton-Ameise mit symmetrischen Bahnen, die aus weißen, roten, blauen und grünen Quadraten zusammengesetzt sind. Beobachtet man die Ameise bei ihrer Bewegung, so merkt man, dass sie immer wieder zu ihrer Ausgangsposition zurückläuft.

```
import java.awt.*;
import java.awt.event.*;
public class RRLL extends Frame {
  static long maxzahl;
  static long zaehler;
  RRLL() {
    super("Turmite 1100 - "+maxzahl+" Schritte");
  }
  public static void main (String[] args) {
    if (args.length<1) {
      System.out.println("Aufruf: % java RRLL schritte (16464)");
      System.exit(0);
    }
    maxzahl = Long.parseLong(args[0]);

    RRLL proggi = new RRLL();
    WindowListener wl = new WindowAdapter() {
      public void windowClosing(WindowEvent e) {
        System.exit(0);
      }
    };
    proggi.addWindowListener(wl);
    proggi.setLocation(100,100);
    proggi.setSize(500,500);
    proggi.show();
  }
  public void paint(Graphics bs) {
    int x;
    int y;
    int i;
    int j;
    int richtung;
    int[][] Welt = new int [100][100];
    for (i=0;i<100;i++)
      for (j=0;j<100;j++) welt[i][j]=0;
    bs.setColor(Color.white);
    bs.fillRect(0,0,500,500);
    zaehler = 0;
    richtung = 1;
    x = 50;
    y = 50;
    x++;
    while  ((!((x==50)&&(y==50)) || (zaehler < maxzahl)) &&
           ((x>0) && (x<100) && (y>0) && (y<100))) {
      if (welt[x][y]==0) {
        richtung = ((richtung + 1) % 4);
        welt[x][y]=1;
        bs.setColor(Color.red);
        bs.fillRect(5*x,5*y,5,5);
      }
      else if (welt[x][y]==1) {
        richtung = ((richtung + 1) % 4);
        welt[x][y]=2;
        bs.setColor(Color.blue);
        bs.fillRect(5*x,5*y,5,5);
      }
      else if (welt[x][y]==2) {
        richtung = ((richtung - 1) % 4);
        welt[x][y]=3;
        bs.setColor(Color.green);
        bs.fillRect(5*x,5*y,5,5);
      }
      else if (welt[x][y]==3) {
        richtung = ((richtung - 1) % 4);
        welt[x][y]=0;
        bs.setColor(Color.white);
        bs.fillRect(5*x,5*y,5,5);
      }
      if (richtung == 0) {
        richtung=4;
        y--;
      }
      else if (richtung == 1) x++;
```

RRLL

RLLRRLLRRLLRRLLR

RLRLLRLLLRLLLLRR

RLRLLRRRLRLRRLRR

```
        else if (richtung == 2) y++;
        else if (richtung == 3) x--;
        zaehler++;
      }
      System.out.println(zaehler + " Schritte");
    }
}
```

Diese verallgemeinerten Langton-Ameisen haben in der Literatur den Namen "Turmiten" erhalten. In den abschließenden Beispielen sollen Turmiten mit beliebigen, 16 Zeichen langen Regelketten behandelt werden:

6. Verschiedene Regelketten für Turmiten verwenden

Erweitert man das letzte Programm auf bis zu 16 Farben, so lassen sich zahlreiche verschiedene Regelketten und damit verschiedenste Turmitenarten wählen. Da die Regelkette als Zeichenkette in ein Textfeld eingegeben wird, müssen die Werte "0" und "1" zunächst mit der `charAt()` - Methode der String-Klasse bestimmt werden.

```
import java.awt.*;
import java.awt.event.*;
public class Turmiten extends Frame
     implements ActionListener {
 int regel[] = new int [16];
 Button ok;
 TextField eingabe;
 static long maxzahl;
 static long zaehler;
 Turmiten() {
   super("Turmiten - wenigstens " + maxzahl + " Schritte");
   setLayout(new FlowLayout());
   Label was = new Label("Regelkette, 16 Ziffern 0-1: ");
   add(was);
   eingabe = new TextField("1001100110011001",16);
   add(eingabe);
   ok = new Button("ok");
   add(ok);
   ok.addActionListener(this);
   werteuebernehmen();
 }
 public static void main (String[] arguments) {
   if (arguments.length<1) {
     System.out.println("Aufruf: % java Turmiten schritte");
     System.exit(0);
   }
   maxzahl = Long.parseLong(arguments[0]);
   Turmiten proggi = new Turmiten();
   WindowListener wl = new WindowAdapter() {
     public void windowClosing(WindowEvent e) {
       System.exit(0);
     }
   };
   proggi.addWindowListener(wl);
   proggi.setLocation(100,100);
   proggi.setSize(500,570);
   proggi.show();
 }
 public void actionPerformed(ActionEvent e) {
   if (e.getSource() == ok) {
     werteuebernehmen();
     repaint();
   }
 }
 public void werteuebernehmen() {
   int el;
   String eintrag = eingabe.getText();
   el = eintrag.length();
   if (el > 16) el = 16;
   for (int i=0; i<el;i++) {
     if (eintrag.charAt(i) == '0') regel[i]=-1;
     if (eintrag.charAt(i) == '1') regel[i]=1;
   }
 }
 public void paint(Graphics bs) {
       ...
 }
}
```

Für die Bewegung und für das Umfärben der einzelnen Quadrate ergeben sich innerhalb der `paint()`-Methode nun 16 Verzweigungen. Der genaue Programmcode für ein entsprechendes Applet kann unter ***[10]*** eingesehen und kopiert werden. Dieses Applet erlaubt, das Verhalten der Turmiten unter dem Einfluss verschiedener Regelketten zu studieren.
Einige Ergebnisse sind auf ***Seite 32*** angegeben.

Die Turmite **"RLLRRLLRRLLRRLLR"** erzeugt diese typischen Flächenfüllungen, die streng symmetrisch verlaufen ***(Seite 32 unten, 2. Abb. v.l.)***. Die Regelkette **"RLRLRLRLRLRLRLRL"** reproduziert die schon bekannte Bahn der Langton-Ameise ***(S. 33 unten)***.

Seltsamerweise ergeben auch Turmiten mit Regelketten, die scheinbar keiner Gesetzmäßigkeit folgen, periodisch wiederkehrende Bewegungen, sodass die auffälligen "Autobahnen" entstehen, etwa die Turmite **"RLRLLRLLLRLLLLRR"** ***(S. 32 unten, 3. Abb. v.l.)***.

Auch völlig irreguläre Turmiten fallen dadurch auf, dass ihre Bahn eine zusammenhängende Fläche erzeugt ***(S. 32 unten, 4. Abb. v.l.)***.

Rechnet man den 16-stelligen Binärcode der Regelketten in Dezimalzahlen um, kann jede Turmite einfach benannt werden. Die Benutzerschnittstelle des letzten Kapitels macht Appetit auf mehr "Performance" - so sollte beispielsweise die Ausgabe der Turmitenbahnen mit Hilfe der Maus gesteuert werden können. Dazu sind so genannte "Threads" notwendig, die in einem späteren Artikel dargestellt werden sollen.

7. Literatur, Weblinks

[1] Jan Stewart, Spektrum der Wissenschaften Aug. 1995, S. 10

[2] Bernd Rümmler, Spektrum der Wissenschaften, Sept. 1995, S. 12

[3] Bernd Rümmler, Spektrum der Wissenschaften, Okt. 1995, S. 10

[4] Wofram Stephen, "A New Kind of Science", ISBN 1579550088

[5] http://www.math.ubc.ca/~cass/www/ant.html (ausgezeichnetes Applet zu Langtons Ameise)

[6] http://www.complex.iastate.edu/information/download/Trend/examples/langton_ant.html (Programmierbeispiel für Langtons Ameise, weitere ausgezeichnete Themen)

[7] http://www.math.sunysb.edu/~scott/ants/ (Informationen zu Ameisen)

[8] http://www.gymmelk.ac.at/~nus/informatik/wpf/JAVA/index.php?kat=appt&teil=lang (Applet zur Langton-Ameise)

[7] http://www-rocq.inria.fr/~fleuret/turmite.en.html (Beispielapplet und Programmcode zu Turmite)

[9] http://cs17.eou.edu/~yosuke1/java/Turmite/apturmite.html (Applet für eine Turmite, die sich über eine Fläche aus lauter gleichseitigen Dreiecken bewegt)

[10] http://www.gymmelk.ac.at/~nus/informatik/wpf/JAVA/index.php?kat=appt&teil=turm (Einfaches Applet zu Turmiten, Programmcodes können durch "copy & paste" übernommen werden ;-))

RLRLRLRLRLRLRLRL