KAPITEL 3

Grafikeinführung

3.1 Voraussetzungen festlegen

Der Begriff »Computergrafik« beschwört eine Vielzahl von Möglichkeiten herauf. Wir können einen computergenerierten Hollywood-Film diskutieren, ein hoch entwickeltes Videospiel, eine Umgebung mit virtueller Realität, ein statisches Bild in Fotoqualität auf einem Bildschirm oder ein eher einfaches Bild, das aus Linien besteht. Der erfahrene Java-Programmierer kann Animationen anzeigen lassen, aber wir wollen uns auf die Anzeige von Bildern mit einfachen Formen beschränken.

3.2 Ein erstes Bild

Im Folgenden beschäftigen wir uns mit dem grafischen Äquivalent des `Gruesse`-Programms aus Kapitel 2. Es zeigt eine diagonale Linie auf dem Bildschirm an. Das Programm-Listing sieht folgendermaßen aus:

```
import java.awt.*;
import java.applet.Applet;
public class ErsteLinie extends Applet {
    public void paint(Graphics g) {
            g.drawLine(0, 0, 100, 100);
    }
}
```

Abbildung 3.1 zeigt das Ergebnis des Programms, wenn es über den Applet-Viewer gestartet wird.

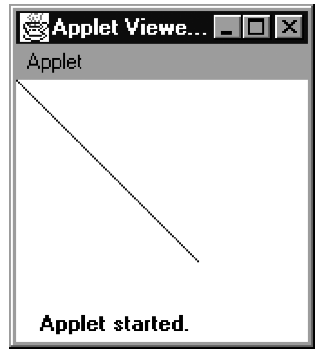
Denken Sie daran: Um ein Applet zu starten, müssen wir eine HTML-Datei erstellen. Auf einigen Java-Programmierumgebungen wird die HTML-Datei automatisch erzeugt. Wenn Ihr System das nicht macht, sollten Sie eine HTML-Datei, die in Kapitel 2 benutzt wurde, kopieren, sie in `ErsteLinie.html` umbenennen und so ändern, dass `ErsteLinie.class` benutzt wird.

```
<title> Webseite mit Applet </title>
<applet code="ErsteLinie.class"
        width=300 height=200> </applet>
```

Die meisten Java-Programmierumgebungen werden diese Datei für Sie generieren.

Um das oben angeführte Programm zu starten, erstellen Sie eine neue Datei mit dem Namen `ErsteLinie.java` und versuchen Sie dann, sie zu kompilieren. Wenn korrekt kompiliert wurde, dann benutzen Sie den Applet-Viewer mit der Datei `ErsteLinie.html`.

Abbildung 3.1 Ausgabe des Applets `Erste Linie`

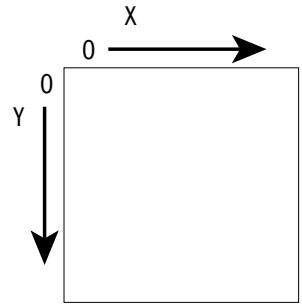


3.3 Der Grafik-Bildschirm

Java-Grafiken basieren auf der Einheit *Pixel*. Ein Pixel ist ein kleiner Punkt auf dem Bildschirm, der eine bestimmte Farbe haben kann. Leider haben verschiedene Bildschirme unterschiedliche Mengen Pixel, sodass wir an dieser Stelle der Einführung einen Grafikbereich festlegen, der klein genug für jeden Computer ist: 300 Pixel in der Breite und 200 Pixel in der Höhe. Dies legen wir im HTML-Code fest, wie der Befehl zeigt:

```
width=300 height=200
```

Abbildung 3.2 Das Pixel-Koordinatensystem in Java



Jedes Pixel wird durch ein Zahlenpaar (seine Koordinaten) angegeben, beginnend bei Null:

- die horizontale Position, häufig (wie auch bei der Java-Dokumentation) in Anlehnung zur Mathematik mit *x* bezeichnet – dieser Wert steigt von links nach rechts an

- die vertikale Position, häufig mit *y* bezeichnet – dieser Wert steigt von oben nach unten

Wir benutzen dieses System, wenn wir mit Java einfache Formen zeichnen wollen. Abbildung 3.2 zeigt den ersten Versuch.

3.4 Erklärung des Programms

Wir erklären nur den folgenden Abschnitt (den wir später verändern wollen). Das restliche Programm sollten Sie einfach nehmen wie es ist. Es sagt lediglich aus, dass wir ein Applet schreiben. Erinnern Sie sich daran, dass wir Java als eine Sprache mit Industriestandard bezeichnet haben, und Zehn-Zeilen-Programme sind nicht seine Stärke!

```
1.    public void paint(Graphics g) {
2.        g.drawLine(0, 0, 100, 100);
3.    }
```

Zeile 1 stellt den Teil Ihres Programms vor, der für die Anzeige von Formen am Bildschirm zuständig ist. Die { } in den Zeilen 1 und 3 umschließen die grafischen Anweisungen. Ein Programmabschnitt, wie oben aufgeführt, wird in Java mit dem Begriff *Methode* bezeichnet.

Die eigentliche Funktion des Programms steht in der zweiten Zeile:

```
g.drawLine(0, 0, 100, 100);
```

Es ließe sich jetzt vermuten, dass diese Zeile sich mit dem Zeichnen einer Linie beschäftigt, aber was ist mit dem »g.«? Es gibt den Bereich an, in dem die Zeichnung stehen soll. In kurzen Programmen hat man möglicherweise keine Wahl – alles, was uns zur Verfügung steht, ist das Applet-Fenster – aber später werden wir sehen, dass der Bildschirm in separate Zeichenbereiche aufgeteilt werden kann.

Die Methode `drawLine` ist eine der vielen Methoden, die vom Java-System in einer Bibliothek zur Verfügung gestellt werden. Die dargestellte Anweisung ist ein Aufruf der Methode mit der Aufforderung, eine Linie anzuzeigen. Die Methode heißt deshalb so, weil es sich um eine Methode (oder einen Weg) handelt, etwas zu tun.

Wenn man die Methode `drawLine` benutzt, muss man sie mit einigen Werten für die Anfangs- und Endpunkte der Linie versorgen, und das in der richtigen Reihenfolge:

1. der horizontale Wert (*x*) des Linienanfangs
2. der vertikale Wert (*y*) des Linienanfangs
3. der horizontale Wert des Linienendes
4. der vertikale Wert des Linienendes

Diese Punkte heißen in Java *Parameter* – sie stellen Eingaben für die Methode `drawLine` dar. Parameter müssen zwischen runden Klammern stehen, getrennt durch Kommata. (Vielleicht sind Sie schon auf den Begriff »Argument« gestoßen, eine alternative Bezeichnung für Parameter.) Diese besondere Methode verlangt vier Parameter, die jeweils Ganzzahlen sein müssen. Wenn wir versuchen, eine falsche Anzahl Parameter oder den falschen Typ einzugeben, erhalten wir eine Fehlermeldung des Compilers. Wir müssen sicher sein, dass

- wir die korrekte Anzahl Parameter eingeben,

- wir den korrekten Typ eingeben,
- sie in der richtigen Reihenfolge stehen.

Einige Methoden verlangen keinerlei Parameter. In diesem Fall müssen wir aber immer noch die Klammern benutzen, wie in

```
machWas();
```

Schließlich beachten Sie bitte noch das Semikolon »;« am Ende der `drawLine`-Parameterliste. In Java muss jede »Anweisung« mit einem Semikolon abgeschlossen werden. Aber was ist eine Anweisung? Die Antwort ist nicht einfach! Wie Sie dem oben angeführten Programm entnehmen können, steht nicht an *jedem* Zeilenende ein Semikolon. Statt komplizierte formale Regeln zu geben, raten wir Ihnen, sich bei Ihren ersten Programmen an unsere Beispiele zu halten. Allerdings ist der Gebrauch einer Methode, gefolgt von ihren Parametern, tatsächlich eine Anweisung, sodass ein Semikolon erforderlich ist.

3.5 Die Methode `paint`

In unserem Beispiel gibt es zwei Arten von Methoden:

- die vom Programmierer geschriebenen Methoden, die ein Teil des Applet-Programms sind
- diejenigen Methoden, die in den Bibliotheken stehen

Alle Methoden haben einen Namen und die meisten haben Parameter. Die Methode `paint` muss vom Programmierer geschrieben werden. Sie wird immer dann vom Browser oder Applet-Viewer aufgerufen, wenn der Bildschirm gezeichnet werden muss. Deshalb muss der Programmierer innerhalb von `paint` alle Anweisungen zum Anzeigen von Text und Grafik angeben. Um `paint` während des Aufrufs zu beobachten, vergrößern Sie das Applet-Fenster mit der Maus und beobachten, wie das Bild neu gezeichnet wird.

3.6 Methoden zum Zeichnen

Neben Linien stellt Ihnen Java noch folgende Möglichkeiten zur Verfügung:

- Rechtecke
- Ellipsen (auch Kreise)
- Bogen

Die folgenden Formen können ebenfalls gezeichnet werden, verlangen aber zusätzliche Java-Kenntnisse. Wir werden die Details über ihre Parameter vernachlässigen und sie in unseren Programmen auch nicht benutzen:

- erhabene (dreidimensionale) Rechtecke
- Rechtecke mit abgerundeten Ecken
- Polygone

Zusätzlich können die Formen mit angegebenen Farben gefüllt werden.

Im Folgenden sehen Sie eine Liste der Parameter für jede Methode und ein Beispielprogramm, das diese Parameter benutzt.

`drawLine`

- der horizontale Wert des Linienanfangs
- der vertikale Wert des Linienanfangs
- der horizontale Wert des Linienendes
- der vertikale Wert des Linienendes

`drawRect`

- der horizontale Wert der oberen rechten Ecke
- der vertikale Wert der oberen linken Ecke
- die Breite des Rechtecks
- die Höhe des Rechtecks

`drawOval`

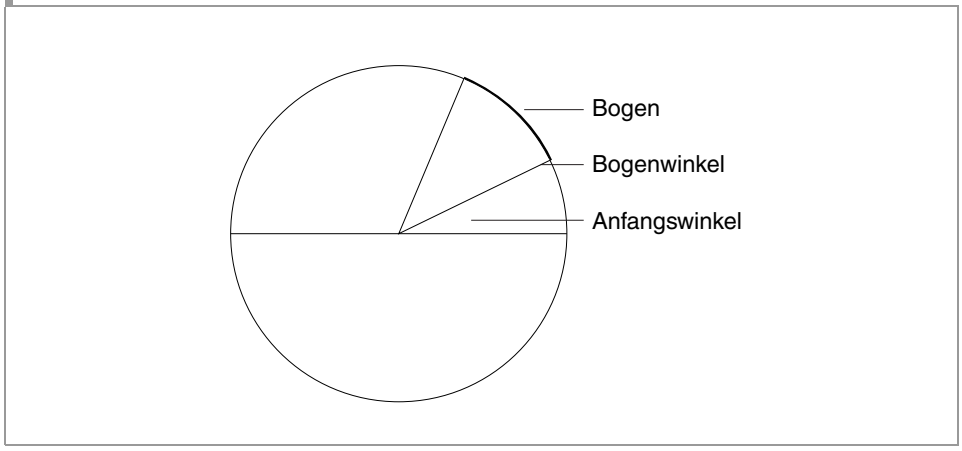
Stellen Sie sich eine Ellipse innerhalb eines Rechtecks vor. Wir brauchen

- den horizontalen Wert der oberen linken Ecke des Rechtecks
- den vertikalen Wert der oberen linken Ecke des Rechtecks
- die Breite des Rechtecks
- die Höhe des Rechtecks

`drawArc`

Ein Bogen ist eine Kurve, ein Teil eines Kreises. Bei dieser Methode werden die Winkel wie in der Mathematik definiert. Null Grad zeigt nach rechts und ein positiver Winkel geht gegen den Uhrzeigersinn. Für den Bogen muss der Startwinkel und die Größe angegeben werden. Wie üblich geben wir ein Rechteck an, das den gesamten 360°-Kreis umschließt. Abbildung 3.3 zeigt die Zeichnung.

Abbildung 3.3 Die Geometrie der Methode `drawArc`



Die Reihenfolge der Parameter lautet:

- der horizontale Wert der oberen linken Ecke des Rechtecks
- der vertikale Wert der oberen linken Ecke des Rechtecks
- die Breite des Rechtecks
- die Höhe des Rechtecks
- der Anfangswinkel
- der Winkel des Bogens

3.7 Formen mit Farben ausfüllen

Es ist möglich, die Farbe, die für die Zeichnung benutzt werden soll, und die Hintergrundfarbe festzulegen. Es gibt 13 Standardfarben:

```
black       blue        cyan        darkGray
gray        green       lightGray   magenta
orange      pink        red         white
yellow
```

(Cyan ist ein dunkles Blaugrün.)

Achten Sie auf die Schreibweise – beachten Sie die Benutzung von Großbuchstaben in der Mitte der Namen. Nachstehend finden Sie ein Beispiel dafür, wie Sie Farben benutzen können:

```
g.setBackground(Color.lightGray);
g.setColor(Color.red);
```

Wenn Sie keine Farbe auswählen, nimmt Java eine vorgegebene Farbe.

Zusätzlich haben wir einen Satz von Methoden zum Füllen von Formen:

```
fillRect
fillArc
fillOval
```

Ihre Parameter sind identisch mit denen der `draw`-Methode, der einzige Unterschied besteht darin, dass `fillArc` ein Tortensegment erzeugt.

3.8 Das Konzept der Anweisungsfolge

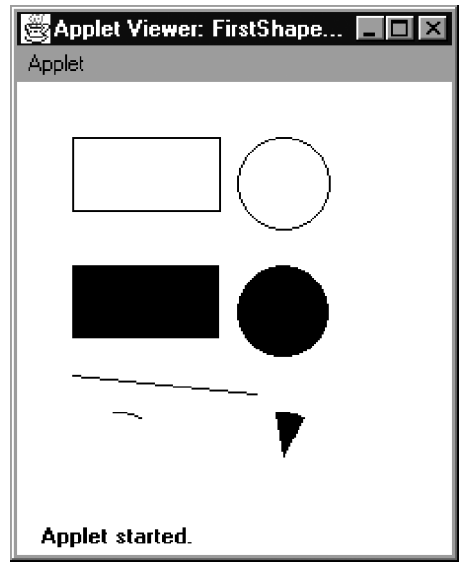
Wenn wir eine Folge von Anweisungen in einem Programm haben, werden sie nacheinander von oben nach unten ausgeführt (solange wir nicht die weiter unten vorgestellten Konzepte der Auswahl und Wiederholung verwenden). Im Folgenden sehen Sie ein Programm, das eine Vielzahl von Formen zeichnet. Abbildung 3.4 zeigt das Ergebnis.

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

public class FirstShapes extends Applet {
    public void paint(Graphics g) {
        g.drawRect(30, 30, 80, 40);
```

```
            g.drawOval(120, 30, 50, 50);
            g.setColor(Color.black);
            g.fillRect(30, 100, 80, 40);
            g.fillOval(120, 100, 50, 50);
            g.drawLine(30, 160, 130, 170);
            g.drawArc(30, 180, 50, 50, 60, 40);
            g.fillArc(120, 180, 50, 50, 60, 40);
        }
    }
```

Abbildung 3.4 Die Ausgabe des Applets `First Shapes`



Die Anweisungen werden von oben nach unten ausgeführt (auch wenn dies auf einem schnellen Computer schwer zu beobachten ist).

FRAGE ZUM SELBSTTEST

3.1 Schreiben und starten Sie ein Applet, das die Form eines großen »T« auf den Bildschirm zeichnet.

3.9 Zeichen anzeigen

In unseren ersten Programmen machen wir ausführlichen Gebrauch von den Java-Fähigkeiten, Meldungen zusammen mit Grafiken anzuzeigen. Wir benutzen die Methode `drawString`, wie in:

```
g.drawString("Hallo Welt", 30, 40);
```

Die Parameter von `drawString` sind:

- eine Zeichenkette; die einfachste Form einer Zeichenkette ist die Folge von Zeichen zwischen doppelten Anführungszeichen
- die horizontale Position des Anfangs der Zeichenkette
- die vertikale Position einer imaginären Linie, auf der die Zeichen stehen; in vielen Fonts haben Zeichen wie »g« eine Unterlänge (ein Teil des Zeichens steht unterhalb der imaginären Linie)

Ein Nachteil beim Kombinieren von Text und Grafik im gleichen Bildschirmbereich besteht darin, dass die Größe der Zeichen (und ihr Font, d.h. ihr »style«) von Ihrem Computer abhängt. Wir setzen voraus, dass der voreingestellte Font und die Zeichengröße ausreichend für unsere einführenden Zwecke sind. Der erfahrene Java-Programmierer würde jedoch für ein Programm einen passenden Font in entsprechender Größe wählen.

3.10 Zeichenketten mit + verbinden

Wir haben die Verwendung von Zeichenketten gezeigt, wie in:

```
g.drawString("Hallo Welt", 40, 40);
```

In Java können Sie verschiedene Zeichenketten aneinander reihen, um eine einzelne Zeichenkette zu bilden, indem Sie das Pluszeichen (+) verwenden. Im Java-Jargon wird das Zeichen + als *Operator* bezeichnet.

Hier ist ein Beispiel:

```
g.drawString("Hallo " + "hier " + "Mike", 100, 100);
```

Die Verwendung von + ergibt »Hallo hier Mike«, das dann von `drawString` benutzt wird.

Zeichenkettenverknüpfungen sind nützlich bei Methoden, die nur eine einzelne Zeichenkette als Parameter akzeptieren (wie `drawString`). In Kapitel 4 wird diese Technik häufig verwendet.

3.11 Kommentieren

Was macht folgendes Programm?

```
g.drawLine(20, 80, 70, 10);
g.drawLine(70, 10, 120, 80);
g.drawLine(20, 80, 120, 80);
```

Die Bedeutung ist nicht sofort offensichtlich und Sie haben es vielleicht mithilfe von Bleistift und Papier versucht herauszubekommen. Die Antwort ist, dass ein Dreieck

mit horizontaler Basis gezeichnet wird, aber das ist aus den drei Anweisungen nicht offensichtlich zu erkennen. In Java kann man durch Voranstellen von »//« *Kommentare* (eine Art von Anmerkung) zu den Anweisungen geben, beispielsweise:

```
// ein Dreieck zeichnen
g.drawLine(20, 80, 70, 10);
g.drawLine(70, 10, 120, 80);
g.drawLine(20, 80, 120, 80);
```

Ein Kommentar kann alles enthalten – es gibt keine Regeln. Es liegt an Ihnen, Ihr Programm mit Kommentaren zu versehen, um es verständlich zu machen.

Kommentare können auch am Ende einer Zeile stehen, wie in:

```
// ein Dreieck zeichnen
g.drawLine(20, 80, 70, 10);
g.drawLine(70, 10, 120, 80);
g.drawLine(20, 80, 120, 80); // Basis zeichnen
```

Übertreiben Sie es nicht mit den Kommentaren. Es ist nicht normal, jede Zeile zu kommentieren, weil dies häufig dazu führt, dieselbe Information mehrfach zu geben. Der folgende Kommentar ist überflüssig:

```
g.drawString("Hallo", 100, 100); // Hallo ausgeben
```

Hier ist der Anweisung auch ohne Kommentar klar zu entnehmen, was sie tut. Benutzen Sie Kommentare eher dazu, das allgemeine Thema eines Programmabschnitts zu definieren, als Details zu jeder Anweisung zu nennen. In Kapitel 22 wird noch einmal darauf eingegangen, wie Kommentare geschrieben werden sollten, die für die Programmdokumentation von Nutzen sind.

PROGRAMMIERFALLE

Achten Sie auf die Zeichensetzung. Kommata, Semikola, runde und geschweifte Klammern müssen genau so wie in den Beispielen geschrieben werden.

HINWEIS ZUR GRAMMATIK

Die Reihenfolge und die Parametertypen müssen für jede Methode richtig angegeben werden.

NEUE SPRACHELEMENTE

- `"..."` zum Kennzeichnen einer Zeichenkette
- `+` zum Verknüpfen von Zeichenketten
- `( )` zum Einschließen von Parametern
- `//` zum Kennzeichnen von Kommentaren

ZUSAMMENFASSUNG

- Anweisungen werden nacheinander ausgeführt, von oben nach unten (wenn nicht anders festgelegt).
- Java verfügt über eine Menge von `draw`-Methoden, die Sie zum Anzeigen von Grafik abrufen können. Diese Methoden können innerhalb der `paint`-Methode stehen.
- Das Positionieren von Grafiken beruht auf Pixel-Koordinaten.
- Parameterwerte können an Methoden übergeben werden.

ÜBUNGEN

Im Folgenden empfehlen wir Ihnen, grobe Skizzen und Berechnungen zu erstellen, bevor Sie das Programm schreiben. Sie können das gleiche HTML und Programm für jedes Problem benutzen. Sie müssen lediglich Ihre neuen Anweisungen zur `paint`-Methode hinzufügen.

3.1 Zeichnen Sie ein Dreieck mit einer vertikalen Seite.

3.2 Zeichnen Sie ein leeres »Tic-Tac-Toe«- Spielbrett (Kreise und Kreuze).

3.3 Entwerfen Sie ein einfaches Haus und zeichnen Sie es.

3.4 Hier sind die Angaben über die Niederschlagsmenge für das Land Xanadu:

1994	150 cm
1995	175 cm
1996	120 cm

a. Stellen Sie die Daten durch eine Serie horizontaler Linien dar.
b. Benutzen Sie `drawString`, um die Zahlen seitlich der Linien hinzuzufügen.
c. Benutzen Sie Rechtecke mit unterschiedlichen Farben statt der Linien.

3.5 Entwerfen Sie eine Zielscheibe mit konzentrischen Kreisen. Fügen Sie dann unterschiedliche Farben hinzu und benutzen Sie `drawString`, um das Ergebnis für jeden Ring anzuzeigen. Die Anschaffung eines Spielzeuggewehrs mit Gummipfeilen zum Schießen auf die Scheibe ist optional.

ANTWORT AUF DIE TESTFRAGE

```
3.1    import java.awt.*;
       import java.applet.Applet;
       public class ErsteLinie extends Applet {
           public void paint(Graphics g) {
              g.drawLine(20, 20, 120, 20);
              g.drawLine(80, 20, 80, 120);
          }
       }
```