Musterlösungen für Blatt 3

325

Lösung für Aufgabe 1 – Zufallsdreieck

```
1  void setup() {
2     size(200, 200);
3     frameRate(1); // damit einzelne Dreiecke gut erkennbar sind, bevor sie
          ersetzt werden
4  }

5  void draw() {
6     background(255);

7     // sechs zufällige Zahlen, die innerhalb des Fensters liegen, als Array
          speichern
8     float[] arr = {random(0, width), random(0, height), random(0, width),
          random(0, height), random(0, width), random(0, height)};

9     // ein Dreieck aus den sechs Zahlen erstellen
10    triangle(arr[0], arr[1], arr[2], arr[3], arr[4], arr[5]);
11 }
```

321

Lösung für Aufgabe 2 – Array-Ausgabe

```
1  int[] arr = {1, 42, 2, 9, -5};

2  for (int i = 0; i < arr.length; i++) {
3     println(arr[i]);
4  }
```

322

Lösung für Aufgabe 3 – Maximum

Die optionale `maxOfTwo`-Funktion

```
1  void setup() {
2     println(maxOfTwo(11, 31));
3  }

4  int maxOfTwo(int a, int b) {
5     if (a > b) {
6        return a;
7     } else {
8        return b;
9     }
10 }
```

Einige `else`-Blöcke können ohne `else` geschrieben werden, da ein `return` die Ausführung der Funktion sofort beendet.

```
1  void setup() {
2     int x = 11;
3     int y = 310;
4     int z = 67;
5     println("Numbers : " + x + ", " + y + ", " + z);
6     println("Max: " + maxOfThree(11, 310, 67));
7  }
```

```
8  int maxOfThree(int a, int b, int c) {
9     if (a > b) {
10       if (a > c) {
11          return a;
12       }
13       return c;
14    } else {
15       if (b > c) {
16          return b;
17       }
18       return c;
19    }
20 }
```

323

Lösung für Aufgabe 4 – Zeichenprogramm II

```
1  void setup(){
2     size(400, 400);
3  }

4  void draw() { }

5  void mousePressed() {
6     circle(mouseX, mouseY, 20);
7  }
```

324

Lösung für Aufgabe 5 – Scrollen

```
1  int y = 200;

2  void setup() {
3     size(400, 400);
4  }

5  void draw() {
6     background(255);
7     circle(200, y, 20);
8  }

9  void mouseWheel(MouseEvent event) {
10    y += event.getCount();
11 }
```

301

Lösung für Aufgabe 6 – Pferderennen

6.1 Darstellung

```
1  void setup() {
2     size(800, 350);
3  }

4  void draw() {
5     strokeCap(PROJECT); // Rechteckige Punkte
6     background(#fee69c);

7     drawFinishLine(); // Ziellinie Zeichnen

8     fill(#f87a63);
9     circle(50, 100, 50); // Pferd 1
10    fill(#427aa1);
11    circle(50, height -100, 50); // Pferd 2
12 }

13 void drawFinishLine() {
14    stroke(255);
15    strokeWeight(20);
16    line(width -45, 0, width -45, height); // Weißer Grund

17    strokeWeight(10);
18    stroke(0);
19    for (int i = 0; i < height + 10; i += 20) { // Schwarze Flecken
20       point(width -50, i);
21       point(width -40, i + 10);
22    }
23 }
```

6.2 Bewegung

```
1  int x1 = 50, x2 = 50;

2  void setup() {
3     size(800, 350);
4  }

5  void draw() {
6     strokeCap(PROJECT);
7     background(#fee69c);

8     drawFinishLine();

9     fill(#f87a63);
10    circle(x1, 100, 50);
11    fill(#427aa1);
12    circle(x2, height -100, 50);
13 }
```

Auf der nächsten Seite geht's weiter...

```
15 void keyReleased() {
16    if (key == 'a' || key == 'd')
17    x1 += 5;
18    else if (key == CODED && (keyCode == LEFT || keyCode == RIGHT))
19    x2 += 5;
20 }
```

6.3 Gewinner:in

```
1  int x1 = 50, x2 = 50;
2  int finishLine;

3  void setup() {
4     size(800, 350);
5     finishLine = width -45;
6     textSize(50);
7     textAlign(CENTER);
8     strokeCap(PROJECT);
9  }

10 void draw() {

11    // ... (bleibt gleich)

12    if (x1 > finishLine) { // x1 hat das Ziel überschritten
13       fill(0, 150);
14       rect(0, 0, width, height);
15       fill(255);
16       text("Pferd 1 (oben) hat gewonnen!", width / 2, height / 2);
17       noLoop();
18    } else if (x2 > finishLine) {
19       fill(0, 150);
20       rect(0, 0, width, height);
21       fill(255);
22       text("Pferd 2 (unten) hat gewonnen!", width / 2, height / 2);
23       noLoop();
24    }
25 }

26 void keyReleased() {
27    // ... (bleibt gleich)
28 }

29 void drawFinishLine() {
30    stroke(255);
31    strokeWeight(20);
32    line(finishLine, 0, finishLine, height);

33    strokeWeight(10);
34    stroke(0);
35    for (int i = 0; i < height + 10; i += 20) {
36       point(finishLine -5, i);
37       point(finishLine + 5, i + 10);
38    }
39 }
```

302

Lösung für Aufgabe 7 – Fantasy-Namensgenerator

```
1  String[] femaleName = new String[]{"Anna", "Emilia", "Gudrun", "Ursela
       ", "Olga", "Maya", "Gisela", "Knort"};
2  String[] maleName = new String[]{"Olaf", "Peter", "Thor", "Nero", "
       Gnaruff", "Ronald", "Gregor", "Rüdiger"};
3  String[] connective = new String[]{"Unermüdliche", "Schläfrige", "
       Hungrige", "Wütende", "Verschnupfte", "Blutende", "Unsportliche",
       "Schnelle", "Faule", "Fleißige", "Programmierende"};
4  String[] surname = new String[]{"von und zu Kunz", "von Drenor", "", "
       von Gießen", "von Atlantis", "aus der Hölle"};

5  String name;

6  void setup() {
7     size(800, 150);
8     textSize(35);
9     textAlign(CENTER);
10    name = getName();
11 }

12 void draw() {
13    background(0);
14    fill(255);
15    text(name, width / 2, 100);
16 }

17 void keyPressed() {
18    name = getName();
19 }

20 String getName() {
21    if (random(2) > 1)
22       return femaleName[(int) random(femaleName.length)] + " die " +
             connective[(int) random(connective.length)] + " " + surname
             [(int) random(surname.length)];
23    else
24       return maleName[(int) random(maleName.length)] + " der " +
             connective[(int) random(connective.length)] + " " + surname
             [(int) random(surname.length)];
25 }
```

314

Lösung für Aufgabe 8 – Galgenmännchen II

8.1 Arrays über Arrays

```
String[] words = new String[]{"Hello", "World", "Mystery", "Question", "
    Hangman", "Java", "Processing", "Secret", "Duck", "Frog", "Alpaca", "
    Lion", "Jazz", "Lasers", "Processing", "THM"};

char[] guesses; // getätigte Tipps
char[] chars; // Gesuchte Buchstaben
char[] display; // Derzeitige Anzeige
int x, // Fortschritt des Galgenmännchens
guessCount; // Tipps gesamt
String word; // Gesuchtes Wort

void setup() {
   size(600, 600);
   background(255);

   x = 0;
   guesses = new char[26];
   word = words[(int) random(words.length)]; // Zufälliges Wort
   chars = word.toLowerCase().toCharArray(); // Buchstaben des Wortes
   display = new char[chars.length];

   for (int i = 0; i < display.length; i++)
      display[i] = '_'; // Display mit Unterstrichen befüllen
}

void draw() {
   switch(x) {
      case 10:
         line(200, 200, 220, 240); // rechter Fuß
      case 9:
         line(200, 200, 180, 240); // linker Fuß
      case 8:
         line(200, 140, 230, 170); // rechte Hand
      case 7:
         line(200, 140, 170, 170); // linke Hand
      case 6:
         line(200, 120, 200, 200); // Körper
      case 5:
         ellipse(200, 100, 40, 40); // Kopf
      case 4:
         line(200, 50, 200, 80); // Seil
      case 3:
         line(100, 80, 130, 50); // Stützbalken
      case 2:
         line(100, 50, 200, 50); // oberer Balken
      case 1:
         line(100, 300, 100, 50); // linker Balken
   }
}
```

8.2 Ist der Charakter im Array?

```
boolean isInArray(char c, char[] array) {
   for (char a : array) { // über alle chars iterieren
      if (a == c)
         return true; // Enthalten
   }
   return false; // Nicht enthalten
}
```

8.3 Action!

```
String[] words = new String[]{"Hello", "World", "Mystery", "Question", "Hangman", "Java", "Processing", "Secret", "Duck", "Frog", "Alpaca", "Lion", "Jazz", "Lasers", "Processing", "THM"};

char[] guesses;
char[] chars;
char[] display;
int x, guessCount;
String word;

void setup() {
   size(600, 600);
   textSize(80);
   textAlign(CENTER);
   strokeWeight(2);

   x = 0;
   guesses = new char[26];
   word = words[(int) random(words.length)];
   chars = word.toLowerCase().toCharArray();
   display = new char[chars.length];
   for (int i = 0; i < display.length; i++)
   display[i] = '_';
}

void draw() {
   background(255);
   fill(0);
   text(String.valueOf(display), width / 2, height *0.75);

   fill(255);
   switch(x) {
      // ... (wie gehabt)
   }
}

boolean isInArray(char c, char[] array) {
   for (char a : array) {
      if (a == c)
      return true;
   }
   return false;
}

void keyReleased() {
   if (isInArray(key, guesses)) // eingegebener Buchstabe ist bereits geraten worden
      return;

   if (isInArray(key, chars)) { // Buchstabe ist gesucht
      for (int i = 0; i < chars.length; i++) {
         if (key == chars[i])
            display[i] = key;
      }
      guesses[guessCount] = key;
   } else { // Falscher Buchstabe
      x++; // Hangman fortsetzen
      guesses[guessCount] = key;
   }
}
```

313

Lösung für Aufgabe 9 – Lights Out

```
1  boolean[][] lamps;
2  int d = 70; // Durchmesser der Lampen

3  void setup() {
4     size(600, 600);
5     textAlign(CENTER, CENTER);
6     background(255);
7     lamps = new boolean[5][5];
8     for (int i = 0; i < lamps.length; i++) {
9        for (int k = 0; k < lamps[i].length; k++) {
10          lamps[i][k] = true; // Alle Lampen werden angeschaltet
11       }
12    }
13 }

14 void draw() {
15    background(74, 92, 102);

16    for (int i = 0; i < lamps.length; i++) {
17       for (int k = 0; k < lamps[i].length; k++) {
18          if (lamps[i][k])
19             fill(128, 168, 36);
20          else
21             fill(74, 92, 102);

22          circle(width / 6 *(i + 1), height / 6 *(k + 1), d);
23       }
24    }

25    if (hasWon())
26    drawGameOver();
27 }

28 boolean hasWon() {
29    for (int i = 0; i < lamps.length; i++)
30       for (int k = 0; k < lamps[i].length; k++)
31          if (lamps[i][k])
32             return false;
33    return true;
34 }

35 void drawGameOver() {
36    background(128, 186, 36);
37    textSize(100);
38    text("Gewonnen", width / 2, height / 2);
39    textSize(20);
40    text("R drücken für Neustart", width / 2, height / 2 + 100);
41 }
```

Auf der nächsten Seite geht's weiter...

```
42 void mousePressed() {
43    if (hasWon()) // wenn Spiel vorbei, keine Inputs mehr akzeptieren
44       return;

45    for (int i = 0; i < lamps.length; i++) {
46       for (int k = 0; k < lamps[i].length; k++) {
47          // Pro Lampe wird geprüft, ob mit der Maus darauf geklickt wurde
48          if (dist(width / 6 *(i + 1), height / 6 *(k + 1), mouseX, mouseY)
                <= d / 2) {

49             lamps[i][k] = !lamps[i][k]; // Angeklickte Lampe

50             if (i > 0) // Nachbarn
51                lamps[i -1][k] = !lamps[i -1][k];
52             if (i < lamps.length -1)
53                lamps[i + 1][k] = !lamps[i + 1][k];
54             if (k > 0)
55                lamps[i][k -1] = !lamps[i][k -1];
56             if (k < lamps.length -1)
57                lamps[i][k + 1] = !lamps[i][k + 1];

58             return; // Lampe wurde gefunden, muss nicht weitersuchen
59          }
60       }
61    }
62 }

63 void keyPressed() {
64    if (key != 'r')
65       return;

66    for (int i = 0; i < lamps.length; i++) {
67       for (int l = 0; l < lamps[i].length; l++) {
68          lamps[i][l] = true;
69       }
70    }
71 }
```

310

Lösung für Aufgabe 10 – Zellulärer Automat

```
1 int[] cells = new int[128]; //Die Zellen einer Generation
2 int generation = 0; //Der Index der Generation
3 int size = 8; //Die Größe jeder Zelle

4 //Das Regelset, was anhand der Grafik erstellt wurde. Kann beliebig
      bearbeitet werden
5 int[] rules = {0, 0, 0, 1, 1, 1, 1, 0};

6 void setup() {
7    size(1000, 1000);
8    cells[64] = 1 ; //Die "Startzelle" der Generation 0
9 }
```

Auf der nächsten Seite geht's weiter...

```
10 void draw() {
11    noStroke();
12    if (generation >= cells.length)
13    return; // Nur zeichnen, bis ein Quadrat gezeichnet wurde

14    for (int i=0; i<cells.length; i++) {
15       if (cells[i] == 1)
16          fill(0); //Lebende Zellen werden schwarz gezeichnet
17       else
18          fill(255);

19       //Die generation gibt die Höhe an, wobei der Index im Array die Breit
             angibt
20       rect(i *size, generation *size, size, size);
21    }

22    cells = makeNewGeneration();
23 }

24 int[] makeNewGeneration() {
25    generation++;
26    int[] next = new int[cells.length]; // nächste Generation

27    for (int i=1; i<cells.length-1; i++) { //Anwendung der Regeln
28       next[i] = getNextCell(cells[i-1], cells[i], cells[i+1]);
29    }

30    return next;
31 }

32 //Wendet die Regeln an, gibt je nach a b und c die nächste Zelle zurück
33 int getNextCell(int a, int b, int c) {
34    switch(a + " " + b + " " + c) {
35       case "1 1 1":
36          return rules[0];
37       case "1 1 0":
38          return rules[1];
39       case "1 0 1":
40          return rules[2];
41       case "1 0 0":
42          return rules[3];
43       case "0 1 1":
44          return rules[4];
45       case "0 1 0":
46          return rules[5];
47       case "0 0 1":
48          return rules[6];
49       case "0 0 0":
50          return rules[7];
51       default:
52          return 0;
53    }
54 }
```

305

Lösung für Aufgabe 11 – Barcode

```
1  boolean[] stripes = new boolean[50];

2  void setup() {
3     size(500, 200);
4     noStroke();

5     for (int i = 0; i < stripes.length; i++)
6     stripes[i] = random(1) >= 0.5;
7  }

8  void draw() {
9     for (int i = 0; i < stripes.length; i++) {
10       if (stripes[i])
11       fill(0);
12       else
13       fill(255);

14       rect(i *10, 0, 10, height);
15    }
16 }
```

11.1 Erweiterung

```
19 void mousePressed() {
20    int index = floor(mouseX / 10f);
21    stripes[index] = !stripes[index];
22 }
```

307

Lösung für Aufgabe 12 – Glücksrad

```
PImage wheel;
float friction = 0.99;
float velocity = 0;
float rotation = 0;
String[] colors = new String[]{"Purple", "Blue", "Light Blue",
   "Green", "Light Green", "Yellow", "Orange", "Red" };

void setup() {
   size(500, 500);
   wheel = loadImage("wheel.png");
   imageMode(CENTER);
   textAlign(CENTER);
   textSize(40);
}

void draw() {
   rotation = (rotation + velocity) % TWO_PI;
   velocity *= friction;

   background(255);

   pushMatrix();
   translate(width / 2, height / 2 -40);
   rotate(rotation);

   image(wheel, 0, 0, 400, 400);

   popMatrix();

   fill(0);
   float rotStep = TWO_PI / colors.length; // Schrittweite der Farbflächen
   text(colors[floor(rotation / rotStep)], width / 2, height -50); // Farbe
        angeben
}

void keyPressed() {
   velocity += 0.05; // Geschwindigkeit erhöhen
}
```

306

Lösung für Aufgabe 13 – Statistik ohne Excel

```
1  int students = 200;
2  int[] gradePercentages;
3  float[] gradeValues;

4  void setup() {
5     size(600, 400);
6     calculateNewGrades();
7     textSize(25);
8  }

9  void draw() {
10    background(255);
11    fill(0);
12    text("Students: " + students, 25, 50);
13    text("Average passing grade: " + nf(getAverageGrade(gradeValues), 1, 1),
           25, 100);
14    text("Average passing percentage: " + nf(getAveragePercentage(
          gradePercentages), 2, 0), 25, 150);

15    text("sehr gut: " + countInRange(gradeValues, 0, 1.6), 25, 250);
16    text("gut: " + countInRange(gradeValues, 1.61, 2.5), 25, 300);
17    text("befriedigend: " + countInRange(gradeValues, 2.51, 3.5), 325, 250);
18    text("ausreichend: " + countInRange(gradeValues, 3.51, 4), 325, 300);
19    text("mangelhaft: " + countInRange(gradeValues, 5, 5), 150, 350);
20 }

21 void calculateNewGrades() { // initialisiert Arrays, berechnet Noten
22    gradePercentages = new int[students];
23    gradeValues = new float[students];
24    for (int i = 0; i < students; i++) {
25       gradePercentages[i] = round(random(100));
26       gradeValues[i] = getGrade(gradePercentages[i]);
27    }
28 }

29 float getGrade(int percentage) {
30    float grade = 4 -(percentage -50) / (50 / 3f); // Anwendung der Formel
31    return grade > 4 ? 5 : grade;
32 }


33 float getAverageGrade(float[] grades) {
34    float gradeSum = 0;
35    int passingGrades = 0;
36    for (int i = 0; i < grades.length; i++) {
37       if (grades[i] < 5) { // Nur bestandene Studierende werden berü
             cksichtigt
38          gradeSum += grades[i];
39          passingGrades++;
40       }
41    }
42    return gradeSum / passingGrades;
43 }
```

Auf der nächsten Seite geht's weiter...

```
44 int getAveragePercentage(int[] percentages) {
45    float percentSum = 0;
46    int passingPercentages = 0;
47    for (int i = 0; i < percentages.length; i++) {
48       if (percentages[i] >= 50) {
49          percentSum += percentages[i];
50          passingPercentages++;
51       }
52    }
53    return round(percentSum / passingPercentages);
54 }

55 int getFailingGrades(float[] grades) {
56    int failSum = 0;
57    for (int i = 0; i < grades.length; i++) {
58       if (grades[i] == 5)
59          failSum++;
60    }
61    return failSum;
62 }

63 int countInRange(float[] values, float min, float max) {
64    int counter = 0;
65    for (int i = 0; i < values.length; i++) {
66       if (values[i] >= min && values[i] <= max)
67          counter++; // ermittelt, wie viele Werte zwischen min und max
               liegen
68    }
69    return counter;
70 }

71 void keyPressed() {
72    calculateNewGrades(); // neue Werte berechnen
73 }
```

317

Lösung für Aufgabe 14 – Tic Tac Toe

In dem `fields`-Array wird der Zustand der einzelnen Spielfelder gespeichert. Dabei werden die Spielerwerte `'1'` für X und `'-1'` für O verwendet. Die Felder werden wie folgt nummeriert:

0	1	2
3	4	5
6	7	8

```
1  // player nimmt zur Unterscheidung der Spieler die Werte -1 und 1 an
2  int player = 1;

3  // initiale Belegung des Arrays mit 0 (== leer)
4  int []fields = {0, 0, 0, 0, 0, 0, 0, 0, 0};

5  float oB = 50; // Abstand oben links zwischen Rand und Board
6  float boardS = 300; // größe des Spielbrettes
7  float cW = boardS / 3; // größe einer Zelle

8  boolean isOver = false; // ist das Spiel vorbei (Sieg oder Unentschieden)?

9  void setup() {
10    size(400, 400);
11    background(0);
12    strokeWeight(8);
13 }

14 void draw() {
15    drawGrid();
16    drawSigns();
17    gameOver();
18 }

19 void drawGrid(){
20    background(0);
21    noFill();
22    stroke(255);
23    rect(oB, oB, boardS, boardS);
24    line(oB, oB + cW, boardS + oB, oB + cW);
25    line(oB, oB + cW *2, boardS + oB, oB + cW *2);
26    line(oB + cW, oB, oB + cW, boardS + oB);
27    line(oB + cW *2, oB, oB + cW *2, boardS + oB);
28 }

29 void drawSigns(){
30    for (int x = 0; x < fields.length; x++) {
31       if (fields[x] == 1) {
32          stroke(255, 0, 0);
33          line(65 + x%3 *cW, 65 + floor(x/3) *cW, 135 + x%3 *cW, 135 + floor
                (x/3) *cW);
34          line(135 + x%3 *cW, 65 + floor(x/3) *cW, 65 + x%3 *cW, 135 + floor
                (x/3) *cW);
35       } else if (fields[x] == -1) {
36          stroke(50, 50, 255);
37          circle((x%3 + 1) *cW, (floor(x/3) + 1) *cW, cW *0.8);
38       }
39    }
40 }
```

Auf der nächsten Seite geht's weiter...

```
41 void mouseClicked() {
42    int pos = findTile(mouseX, mouseY);
43    if ((pos != -1) && (fields[pos] == 0) && !isOver){
44       fields[pos] = player;
45       // Spieler wechseln
46       player = -player;
47    }
48 }

49 int findTile(float mX, float mY) {
50    if (mouseX < oB || mouseX > oB + boardS || mouseY < oB || mouseY > oB +
          boardS){
51       // Maus nicht im Spielfeld
52       return -1;
53    } else {
54       // mouseX und mouseY bekommen
55       mX -= oB;
56       // den linken Rand abgezogen
57       mY -= oB;
58       return floor(mX / cW) + floor(mY / cW) *3;
59    }
60 }

61 int findWin() {
62    for(int i = 0; i < 3; i++){
63       // prüfe Zeilen
64       if(fields[i*3] != 0 && fields[i*3] == fields[i*3 + 1] && fields[i*3]
             == fields[i*3 + 2]){
65          isOver = true;
66          return fields[i*3];
67       }
68       // prüfe Spalten
69       else if(fields[i] != 0 && fields[i] == fields[i+3] && fields[i] ==
             fields[i+6]){
70          isOver = true;
71          return fields[i];
72       }
73    }

74    // prüfe Diagonalen
75    if(fields[0] != 0 && fields[0] == fields[4] && fields[0] == fields[8]){
76       isOver = true;
77       return fields[0];
78    }
79    else if(fields[4] != 0 && fields[2] == fields[4] && fields[2] == fields
          [6]){
80       isOver = true;
81       return fields[2];
82    }
83    // alle Felder sind belegt (daher gibt es keine 0 mehr) -> Unentschieden
84    else if(!checkForZero()){
85       return 9;
86    }
87    else{
88       return 0;
89    }
90 }
```

Auf der nächsten Seite geht's weiter...

```
91  boolean checkForZero(){
92     for(int i = 0; i < fields.length; i++){
93        if(fields[i] == 0){
94           return true;
95        }
96     }
97     return false;
98  }

99  void gameOver(){
100    int winner = findWin();
101    fill(255);
102    stroke(0);
103    textSize(35);
104    if (winner == 1) {
105       text("Game Over! X wins!", 0, 35);
106    }
107    else if (winner == -1) {
108       text("Game Over! O wins!", 0, 35);
109    }
110    else if(winner == 9){
111       text("Game Over! IT'S A DRAW", 0, 35);
112    }
113 }
```

316

Lösung für Aufgabe 15 – Barnsley Farn

```
1  // Variablen für die Funktionen:
2  float[] a, b, c, d, e, f, p;

3  // Koordinaten des aktuellen Punktes:
4  float x, y;

5  // Hilfsvariablen um den nächsten Punkt zu berechnen
6  float newX, newY;

7  // Anzahl der gezeichneten Punkte:
8  int points = 50000;

9  // Faktor um den der Farn vergrößert wird (sonst ist das Bild winzig)
10 float factor = 50;

11 void setup() {
12    size(400, 600);

13    /* Für grüne Punkte auf schwarzem Hintergrund */
14     background(0);
15    stroke(0, 255, 0);

16    strokeWeight(2); // Punktgröße vergrößern

17    //Variablen für den Farn: Schwarzstieliger Streifenfarn
18    a = new float[]{0, 0.85, 0.2, -0.15};
19    b = new float[]{0, 0.04, -0.26, 0.28};
20    c = new float[]{0, -0.04, 0.23, 0.26};
21    d = new float[]{0.16, 0.85, 0.22, 0.24};
22    e = new float[]{0, 0, 0, 0};
23    f = new float[]{0, 1.6, 1.6, 0.44};

24    // Wahrscheinlichkeiten für die vier Funktionen
25    p = new float[]{0.01, 0.85, 0.07, 0.07};

26    // ALTERNATIV: Variablen für Cyclosorus (Sumpffarngewächs):
27    /*
28    // a = new float[]{0, 0.95, 0.035, -0.04};
29    // b = new float[]{0, 0.005, -0.2, 0.2};
30    // c = new float[]{0, -0.005, 0.16, 0.16};
31    // d = new float[]{0.25, 0.93, 0.04, 0.04};
32    // e = new float[]{0, -0.002, -0.09, 0.083};
33    // f = new float[]{-0.4, 0.5, 0.02, 0.12};
34    // p = new float[]{0.02, 0.84, 0.07, 0.07};
35    */
36    // Der Startpunkt ist (0,0)
37    x = 0;
38    y = 0;

39    // In jedem Schleifendurchlauf wird ein Punkt gezeichnet:
40    drawPoints(points);

41    noLoop();
42 }
```

Auf der nächsten Seite geht's weiter...

```
38 int chooseFunction(float i) {
39    float max = 0;
40    int index = -1;
41    while (i > max) {
42       max += p[++index];
43    }
44    return index;
45 }

46 void drawPoints(int points) {
47    for (int i = 0; i < points; i++) {

48       int index = chooseFunction(random(1));

49       newX = a[index] *x + b[index] *y + e[index];
50       newY = c[index] *x + d[index] *y + f[index];

51       // Erst nachdem beide neuen Werte bestimmt wurden werden diese in x
             und y gespeichert
52       x = newX;
53       y = newY;

54       // x und y werden mit den Faktor multipliziert (y mit -factor, da der
              Farn sonst nach unten wächst) und der Startpunkt mittig an den
             unteren Bildrand verlegt.
55       point(x *factor + width/2, y *-factor + height);
56    }
57 }

58 void draw() {
59    // Wird nicht benötigt!
60 }
```

319

Lösung für Aufgabe 16 – Planetenmodell

```
PVector p = new PVector(200, 500); // p_p
PVector v = new PVector(0, 2.5); // v_p
float pSize = 30; // Größe des Planeten

PVector sun = new PVector(500, 500); // p_s
float g = 0.001; // Gravitationskonstante
float sunSize = 100; // Größe der Sonne
float mass = 3000000; // Masse der Sonne

void setup() {
   size(1000, 1000);
   strokeWeight(3); // Dickere Linien
}

void draw() {
   background(0);

   float grav = g *(mass / pow(PVector.dist(p, sun), 2)); // Formel 1
   float dir = atan2(sun.y -p.y, sun.x -p.x); // Formel 2

   PVector result = PVector.fromAngle(dir); // Richtungsvektor
   result.setMag(grav); // Skalar

   v.add(result); // Auf Geschwindigkeit addieren
   p.add(v); // Geschwindigkeit auf Position addieren

   // Linien
   PVector line = PVector.add(p, v.copy().mult(50));
   PVector line2 = PVector.add(p, result.mult(2000));

   stroke(255, 0, 0);
   line(p.x, p.y, line.x, line.y);
   stroke(0, 0, 255);
   line(p.x, p.y, line2.x, line2.y);

   // Körper
   stroke(150);
   circle(sun.x, sun.y, sunSize);
   circle(p.x, p.y, pSize);
}
```