Musterlösungen für Blatt 1

135

Lösung für Aufgabe 1 – Linien I

```
size(200, 200);
strokeWeight(2);
background(255);

line(0, 0, 200, 200);
line(0, 200, 200, 0);
line(50, 100, 150, 100);
```

```
size(200, 200);
strokeWeight(2);
background(255);

stroke(255, 0, 0);
line(0, 0, 200, 200);
circle(100, 100, 100);
line(0, 200, 200, 0);
```

137

Lösung für Aufgabe 2 – Rechnungen

```
int x = 3;
int y = 7;

println("x = " + x + ", y = " + y);
println("x + y = " + (x + y));
println("x -y = " + (x -y));
println("x *y = " + (x *y));
println("x / y = " + (x / y))

/*
  Wenn man mit Floats versucht durch 0 zu teilen,
  ist das Ergebnis Infinity bzw. -Infinity.
  Bei 0 / 0 ist das Ergebnis NaN (not a number).
  Wenn Intergers verwendet werden,
  löst eine Division durch 0 einen Fehler aus: ArithmeticException.
*/
```

138

Lösung für Aufgabe 3 – Analysis

```
int x = 12;
println("x ist " + x + ".");

if (x < 100) {
   println("x ist kleiner als 100.");
} else {
   println("x ist mindestens 100.");
}

if (x % 2 == 0) {
   println("x ist gerade.");
} else {
   println("x ist ungerade.");
}
```

```
13 if (x *50 < 100) {
14    println("x *5 ist kleiner als 50.");
15 } else {
16    println("x *5 ist mindestens 50.");
17 }
```

136

Lösung für Aufgabe 4 – Linien II

```
1  size(200, 200);
2  strokeWeight(2);
3  background(0);

4  stroke(255);
5  line(0, 0, 200, 200);
6  line(0, 200, 200, 0);

7  fill(0);
8  stroke(255, 0, 0);
9  circle(0, 100, 100);
10 stroke(0, 255, 0);
11 circle(100, 0, 100);
12 stroke(0, 0, 255);
13 circle(200, 100, 100);
14 stroke(255, 0, 255);
15 circle(100, 200, 100);
```

```
1  size(200, 200);
2  strokeWeight(14);
3  background(0);

4  stroke(255);
5  line(0, 0, 100, 100);

6  noFill();
7  stroke(0, 0, 255);
8  circle(100, 100, 100);

9  stroke(255);
10 line(100, 100, 200, 200);
```

```
1  size(200, 200);
2  strokeWeight(14);
3  background(0);

4  stroke(255,0,0);
5  line(0, 0, 200, 200);

6  stroke(255);
7  noFill();
8  circle(100, 100, 100);

9  stroke(0, 255, 0);
10 line(200, 0, 100, 100);
11 line(0, 200, 100, 100);

12 stroke(255, 0, 0);
13 line(75, 75, 125, 125);
```

101

Lösung für Aufgabe 5 – Einstiegsaufgabe

5.1 Zeichnung

```
1 size(200, 200); // Bildebene: Höhe, Weite

2 circle(100, 100, 100); // x, y, Durchmesser
3 circle(100, 100, 50);
4 circle(100, 100, 25);
```

5.2 Positionierung

Hier sollten die Variablen `x` und `y` angelegt und verwendet werden.

```
1 size(200, 200);
2 background(255); // Weißer Hintergrund

3 int x = 70; // Deklarierung der Variablen x und y
4 int y = 110; // > legen die Position aller Kreise fest

5 circle(x, y, 120); // Alle drei Kreise verwenden nun x und y
6 circle(x, y, 60); // als Positionen.
7 circle(x, y, 30);
```

5.3 Größeneinstellung

Schwierigkeit war hier die Erkenntnis, dass der Radius die Hälfte des Durchmessers darstellt, und durch `r *2` in einem Ausdruck berechnet werden kann.

```
3 int x, y, r; // Alternative der Variablendeklaration:
4 x = 80; // Erst deklarieren, dann setzen
5 y = 110;
6 r = 75; // Radius

7 circle(x, y, r *2); // Das Doppelte vom Radius ist der Durchmesser
8 circle(x, y, r); // r *2 / 2 = r
9 circle(x, y, r / 2); // r *2 / 4 = r / 2
```

5.4 Anmalen

Hier galt es, für jede der Kreise eine Farbe zu wählen und vor den jeweiligen `circle(x, y, d)`-Befehl zu setzen.

```
7 fill(255, 255, 0); // Gelb
8 circle(x, y, r *2);
9 fill(0, 0, 255); // Blau
10 circle(x, y, r);
11 fill(255, 0, 0); // Rot
12 circle(x, y, r / 2);
```

5.5 Passgenauigkeit

Hier sollte eine Bedingung gefunden werden, welche prüft, ob der Radius derzeit zu groß ist, um in die Bildebene zu passen.

```
3  int x, y, r;
4  x = 100; // Um diesen Ansatz zu ermöglichen, muss die Zielscheibe
5  y = 100; // zentriert sein.
6  r = 75;

7  if (r *2 <= width) { // Ist der Durchmesser kleiner als der Canvas?
8     fill(255, 255, 0); // ...dann zeichne die Zielscheibe.
9     circle(x, y, r *2);
10    fill(0, 0, 255);
11    circle(x, y, r);
12    fill(255, 0, 0);
13    circle(x, y, r / 2);
14 }
```

5.6 Flexible Passgenauigkeit

Um die Passgenauigkeit bei allen möglichen Positionen zu prüfen, muss `r` sowohl mit `y` und `x` (welche den Abstand zwischen der oberen bzw. linken Seite und der Zielscheibe beschreiben) als auch mit `height` und `width` (unten, rechts) abgeglichen werden.

```
3  int x, y, r;
4  x = 150; // Nun können x und y verändert werden,
5  y = 50; // und die Zielscheibe wird weiterhin nur gezeichnet,
6  r = 50; // wenn sie inkl. Radius nicht aus der Bildebene ragt.

7  if (r <= x && r <= y && r <= width -x && r <= height -y) {
8     fill(255, 255, 0);
9     circle(x, y, r *2);
10    fill(0, 0, 255);
11    circle(x, y, r);
12    fill(255, 0, 0);
13    circle(x, y, r / 2);
14 }
```

5.7 Größenanpassung

Für die Anpassung von `r` ist es sinnvoll, den Ausdruck so umzuändern, dass er nun wahr ist, sobald der Radius zu groß ist. Dann kann der kleinste Abstand zu einer Wand für den neuen Radius gewählt werden. Die Vorlage für den `min(int[] a)`-Befehl wurde in der Aufgabenstellung gegeben.

```
7  if (r > x || r > y || r > width -x || r > height -y) {
8     r = min(new int[]{x, y, width -x, height -y});
9  }

10 fill(255, 255, 0);
11 circle(x, y, r *2);
12 fill(0, 0, 255);
13 circle(x, y, r);
14 fill(255, 0, 0);
15 circle(x, y, r / 2);
```

129

Lösung für Aufgabe 6 – Dreidimensionaler Würfel

```
size(400, 400);
background(255);
strokeWeight(2);
noFill();

int oX = 100; // offset X (Verschiebung des Würfels)
int oY = 100; // offset Y
int extent = 150; // Weite und Höhe des Würfels

int fX = 50; // foreground X (Wie "tief" der Würfel erscheint)
int fY = 50; // foreground Y

square(oX, oY, extent); // hinten

line(oX, oY, oX + fX, oY + fY); // oben links
line(oX + extent, oY, oX + fX + extent, oY + fY); // oben rechts
line(oX, oY + extent, oX + fX, oY + fY + extent); // unten links
line(oX + extent, oY + extent, oX + fX + extent, oY + fY + extent); // unten rechts

square(oX + fX, oY + fY, extent); // vorne
```

102

Lösung für Aufgabe 7 – THM-Logo

7.1 Würfel

```
size(500, 500);
float e = 100; // Höhe und Weite des Würfels

square(0, 0, e); // Würfel oben links
```

7.2 Zwei Würfel mit Abstand

Um `x` zu ermitteln, musste hier die Breite des ersten Würfels (`e`) plus der Hälfte von `e` als Abstand gerechnet werden.

```
square(0, 0, e);
square(e + e / 2, 0, e); // rechter Würfel
```

7.3 Vertikaler Nachbar und Offset

Hier muss `offsetX` zu jedem `x`-Wert der Würfel, und `offsetY` zu jedem `y`-Wert der Würfel addiert werden.

```
size(500, 500);
float e = 100;
float offsetX = 50; // Verschiebung nach rechts
float offsetY = 50; // Verschiebung nach unten

square(offsetX + 0, offsetY + 0, e);
square(offsetX + e + e / 2, offsetY + 0, e);
square(offsetX + e + e / 2, offsetY + e + e / 2, e); // unten rechts
```

7.4 Das Logo

```
1  size(500, 500);
2  background(255);
3  fill(128, 186, 36);

4  float e = 75;
5  float offsetX = 100;
6  float offsetY = 50;

7  square(offsetX + e + e / 2, offsetY, e); // Zeile 1
8  square(offsetX + 2 *( e + e / 2), offsetY, e);

9  offsetY += e + e / 2;
10 square(offsetX + 0, offsetY, e); // Zeile 2
11 square(offsetX + e + e / 2, offsetY, e);
12 square(offsetX + 2 *( e + e / 2 ), offsetY, e);

13 offsetY += e + e / 2;
14 square(offsetX + 0, offsetY, e); // Zeile 3
15 square(offsetX + e + e / 2, offsetY, e);
16 square(offsetX + 2 *( e + e / 2 ), offsetY, e);

17 offsetY += e + e / 2;
18 square(offsetX + 0, offsetY, e); // Zeile 4
19 square(offsetX + e + e / 2, offsetY, e);
20 square(offsetX + 2 *( e + e / 2 ), offsetY, e);
```

128

Lösung für Aufgabe 8 – Geheimkombination

```
1  size(400, 400);

2  boolean secretBoolean = false;
3  char secretCharacter = 'g';
4  String secretString = "theSecretRecipe";
5  int secretInt = 42;
6  float secretFloat = 4.8;

7  boolean combinationWrong = false;

8  if(secretBoolean == true) { // == als vergleich
9     combinationWrong = true;
10 }

11 if(secretCharacter != 'g') { // != bedeutet "ungleich"
12    combinationWrong = true;
13 }

14 if(!secretString.equals("theSecretRecipe")) { // Ausrufezeichen verneint
       derzeitigen Wahrheitswert (true -> false, false -> true)
15    combinationWrong = true;
16 }

17 if(secretInt != 42) {
18    combinationWrong = true;
19 }

20 if(secretFloat < 4 || secretFloat > 5) { // kleiner als 4 oder größer als
       5?
21    combinationWrong = true;
22 }

23 rectMode(CENTER);
24 if(combinationWrong) {
25    fill(255, 0, 0); // rot
26 } else {
27    fill(0, 255, 0); // grün
28 }
29 rect(width / 2, height / 2, 100, 100);
```

103

Lösung für Aufgabe 9 – Galgenmännchen

Da die größeren Werte auch die anderen Zeichenbefehle aufrufen sollen, werden die größeren Werte über die kleineren gestellt, und es wird kein `break` verwendet.

```
size(300, 300);
background(255);

int x = 10; // kontrolliert den derzeitigen Fortschritt
            // des Galgenmännchen. Fertig: x = 10.

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
}
```

105

Lösung für Aufgabe 10 – Positionierung

Optional hätte hier auch mit `width` und `height` gearbeitet werden können.

```
size(600,600);
background(255);

strokeWeight(1); // dünn
stroke(0); // schwarze Umrandung
fill(255); // weiße Füllung
ellipse(50,50,50,50);
ellipse(50,100,50,50);
rect(525,480,50,50);
rect(525,530,50,50);
line(500,25,500,75);
line(25,500,75,500);

fill(0); // schwarze Füllung
ellipse(100,50,50,50);
rect(475,480,50,50);

strokeWeight(3); // dick
line(550,25,550,75);
line(25,550,75,550);

stroke(color(#ff0000)); // rote Umrandung
line(75,550,125,550);
line(550,75,550,125);
```

130

Lösung für Aufgabe 11 – Kunst des Zufalls

```
size(400, 400);
background(255);

int type = floor(random(2)); // Kreis (0) oder Quadrat (1)
int x = floor(random(300)); // x-Position
int y = floor(random(300)); // y-Position
int d = floor(50 + random(50)); // Größe

int r = floor(random(255)); // Rot
int g = floor(random(255)); // Grün
int b = floor(random(255)); // Blau

fill(r, g, b);

if(type == 0) {
   circle(x, y, d);
} else {
   square(x, y, d);
}
```

131

Lösung für Aufgabe 12 – Distanzberechnung

```
1  size(400, 400);
2  background(255);

3  int x1 = 350; // Punkt 1
4  int y1 = 300;

5  int x2 = 50; // Punkt 2
6  int y2 = 250;

7  float dist = sqrt(pow(x1 -x2, 2) + pow(y1 -y2, 2)); // https://www.
       mathsisfun.com/algebra/distance-2-points.html

8  line(x1, y1, x2, y2); // Linie zwischen den Punkten
9  circle(x1, y1, 20);
10 circle(x2, y2, 20);

11 fill(0);
12 textSize(20);
13 text(dist, 5, 20); // Angabe der Distanz
```

132

Lösung für Aufgabe 13 – Kollisionserkennung

Hier galt es, sich die Bedingungen zu visualisieren, welche Voraussetzung für eine Kollision sind:

1. ...die linke Kante von Box 1 muss weiter links als die rechte Kante von Box 2 sein.
2. ...die rechte Kante von Box 1 muss weiter rechts als die linke Kante von Box 2 sein.
3. ...die obere Kante von Box 1 muss weiter oben als die untere Kante von Box 2 sein.
4. ...die untere Kante von Box 1 muss weiter unten als die obere Kante von Box 2 sein.

```
1  size(400, 400);
2  background(255);

3  int x1 = 50; // Box 1:
4  int y1 = 200; // Positionen
5  int dx1 = 250; // Weite
6  int dy1 = 175; // Höhe

7  int x2 = 150; // Box 2:
8  int y2 = 50; // Positionen
9  int dx2 = 100; // Weite
10 int dy2 = 200; // Höhe

11 rect(x1, y1, dx1, dy1);

12 if (x1 < x2 + dx2 // 1
13    && x1 + dx1 > x2 // 2
14    && y1 < y2 + dy2 // 3
15    && y1 + dy1 > y2) // 4
16 {
17    fill(color(255, 0, 0)); // -> es gibt eine Kollision!
18 } else {
19    fill(color(0, 255, 0)); // keine Kollision
20 }

21 rect(x2, y2, dx2, dy2);
```

110

Lösung für Aufgabe 14 – Rotationen

14.1 Einleitung zu Rotationen

```
1 size(400, 400);
2 background(255);
3 noFill();
4 // Ursprung in die Mitte verschieben, damit um den Mittelpunkt des Fensters
      rotiert werden kann
5 translate(width / 2, height / 2);
6 // Rotation um einen Achtelkreis (45 grad)
7 rotate(PI/4);
8 ellipse(0, 0, 200, 100);
```

14.2 push- und popMatrix()

```
1 size(400, 400);
2 background(255);
3 translate(width / 2, height / 2);
4 pushMatrix(); // Ablegen der Matrix
5 rotate(PI/4);
6 ellipse(0, 0, 200, 100);
7 popMatrix(); // Abrufen der Matrix
8 rect(-100, -25, 200, 50);
```

134

Lösung für Aufgabe 15 – Die dritte Dimension

```
1 size(800, 800, P3D);
2 background(255);
3 translate(width / 2, height / 2); // Ursprung in die Mitte verschieben
4 noFill(); // Nur Umrandung zeichnen
5 sphere(300);
6 rotateY(QUARTER_PI *0.75); // Nach "rechts" rotieren
7 rotateX(-QUARTER_PI / 2); // Leicht nach unten neigen
8 fill(0, 0, 255);
9 strokeWeight(5);
10 box(300);
```

133

Lösung für Aufgabe 16 – Mondlandung

```
PImage space = loadImage("space.jpg");
PImage lunarModule = loadImage("lunar_module.png");

size(1000, 683);
background(space);
imageMode(CENTER); // Bilder werden ausgehend vom Zentrum gezeichnet
textSize(30);

int moonX = width / 2; // Mond Position
int moonY = height / 2;
float moonR = height / 4; // Radius
float moonD = moonR *2; // Durchmesser

int moduleX = width / 2 -100; // Modul Position
int moduleY = height / 2 -190;
int moduleDX = 100; // Weite
int moduleDY = 100; // Höhe

PVector moduleToMoon = new PVector(moonX -moduleX, moonY -moduleY); // Vektor vom Modul zum Mond
float facing = moduleToMoon.heading() -HALF_PI; // Ausrichtung des Moduls
float distance = moduleToMoon.mag() -(moonR + moduleDY / 2); // Abstand Modul zum Mond, abzüglich Radius des Mondes und die Hälfte der Höhe des Moduls

String status = ""; // Auswahl Text oben links
if(distance > 500) {
   status = "im Orbit";
} else if (distance > 10) {
   status = "Landung";
} else if (distance > -10) {
   status = "gelandet";
} else {
   status = "abgestürzt";
}

pushMatrix(); // Rotation und Platzierung des Moduls
translate(moduleX, moduleY);
rotate(facing);
image(lunarModule, 0, 0, moduleDX, moduleDY);
popMatrix();

fill(color(#937f73));
circle(moonX, moonY, moonD); // Mond

fill(255);
text(status, 30, 50); // Status
```