

Musterlösungen für Blatt 2

231



Lösung für Aufgabe 1 – RGB-Boxen

```

1 size(200, 200);
2 noStroke();

3 fill(255, 0, 0);
4 rect(0, 0, 200, 200);

5 fill(0, 255, 0);
6 rect(0, 0, 150, 200);

7 fill(0, 0, 255);
8 rect(0, 0, 150, 150);
  
```

232



Lösung für Aufgabe 2 – Summenformel

```

1 int sum = 0;

2 for (int i = 1; i <= 100; i++) {
3   sum += i;
4   println(sum);
5 }
  
```

233



Lösung für Aufgabe 3 – Zeichenprogramm

```

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

4 void draw() {
5   circle(mouseX, mouseY, 10);
6 }
  
```

234



Lösung für Aufgabe 4 – Ausweichen

```

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

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

6   if(mouseX < 200) {
7     circle(325, 200, 50);
8   } else {
9     circle(75, 200, 50);
10  }
11 }
  
```

201



Lösung für Aufgabe 5 – Einstiegsaufgabe

5.1 Punkte im Raum

Mithilfe von z.B. `random(width)` kann ein zufälliger Wert zwischen 0 und `width` erhalten werden. `strokeWeight(i)` kann die Dicke der Pixel (und anderem!) beeinflussen und so diese besser sichtbar machen.

```

1 void setup() {
2   size(400, 400); // Fenstergröße 400x400 Pixel
3   background(255); // Hintergrundfarbe Weiß
4   strokeWeight(3); // Dicke der Pixel
5 }
6
6 void draw() {
7   point(random(width), random(height));
8 }
    
```

5.2 Mehr Punkte!

Schleifen sind schneller als ein Durchlauf der `draw()`-Methode, da dort nicht der Zeichenbereich aktualisiert wird (sondern dies erst am Ende des Methodendurchlaufs erfolgt).

```

6 void draw() {
7   for (int i = 0; i < 100; i++) { // 100 Wiederholungen
8     point(random(width), random(height));
9   }
10 }
    
```

5.3 Randomwalk

Hier gibt es viele Möglichkeiten, die Bewegung des Pixels zu ermitteln. Hier gilt das Probieren: Einige der Möglichkeiten könnten ungewollt bestimmte Richtungen bevorzugen.

```

1 int x, y; // Position des Pixels
2
2 void setup() {
3   size(200, 200); // Fenstergröße 400x400 Pixel
4   background(255); // Hintergrundfarbe Weiß
5   pixelDensity(2); // Dicke der Pixel
6   x = width / 2;
7   y = height / 2;
8 }
9
9 void draw() {
10  for (int i = 0; i < 100; i++) {
11    point(x, y);
12    x += round(random(2)) - 1; // -1, 0 oder 1
13    y += round(random(2)) - 1; // -1, 0 oder 1
14  }
15 }
    
```

5.4 Absperrung

Auch hier gibt es verschiedene Möglichkeiten, die Bewegung zu beeinflussen, welche alle gleichwertig richtig sein können.

```

9 void draw() {
10     for (int i = 0; i < 100; i++) {
11         point(x, y);
12         x += round(random(2)) -1;
13         y += round(random(2)) -1;

14         if(x < 0) x++; // Pixel ist zu weit links
15         if(x > width) x--; // zu weit rechts
16         if(y < 0) y++; // zu weit unten
17         if(y > height) y--; // zu weit oben
18     }
19 }
```

5.5 Farbmuster

Hier ein paar mögliche Farbmuster. `int counter` ist eine globale Variable, die ganz oben deklariert werden muss.

```
11 stroke(random(255)); // Zufälliger Grauton
```

```
11 stroke(random(255), 0, random(255)); // Zufälliger Rot-Blau-Ton
```

```

11 if (counter >= 256) {
12     counter = 0;
13 }
14 stroke(counter); // Farbverlauf Schwarz nach Weiß
15 counter++;
```

Dieser Codeblock kann einen Regenbogen erzeugen, vorausgesetzt er ist richtig implementiert. Mit `step` kontrollieren Sie, wie schnell von einer Farbe zur nächsten gewechselt werden soll. Achtung: Kann schnell ändernde Farben hervorrufen!

```

11 int step = 2000; // Höher = langsamerer Farbverlauf

12 if (counter >= step *6)
13     counter = 0;

14 if (counter++ < step) { // Regenbogen-Farbverlauf
15     stroke(255, (counter *255) / step, 0); // Grün aufsteigend
16 } else if (counter < step *2) {
17     stroke(255 - (counter *255) / (step *2), 255, 0); // Rot absteigend
18 } else if (counter < step *3) {
19     stroke(0, 255, (counter *255) / (step *3)); // Blau aufsteigend
20 } else if (counter < step *4) {
21     stroke(0, 255 - (counter *255) / (step *4), 255); // Grün absteigend
22 } else if (counter < step *5) {
23     stroke((counter *255) / (step *5), 0, 255); // Rot aufsteigend
24 } else {
25     stroke(255, 0, 255 - (counter *255) / (step *6)); // Grün absteigend
26 }
```

204



Lösung für Aufgabe 6 – Countdown

6.1 Das Konzept

```

1  int counter = 10;

2  void setup() {
3      size(400, 400);
4      textSize(50); // Schriftgröße
5      textAlign(CENTER); // Schriftposition

6      frameRate(1); // Ein draw()-Durchlauf pro Sekunde
7  }

8  void draw() {
9      background(#FFFFFF);
10     fill(0);

11     text(counter, width / 2, height / 2);

12     if (counter > 0) // Wenn noch nicht bei 0
13         counter--; // Runterzählen!
14 }
    
```

6.2 Liftoff!

```

1  int counter = 10;
2  float x, y, v, a;

3  void setup() {
4      size(400, 400);
5      textSize(50);
6      textAlign(CENTER);
7      fill(0);

8      frameRate(1);

9      x = width / 2; // x-Pos.
10     y = height * 0.75; // y-Pos.
11     v = 0; // Vertikale Geschwindigkeit
12     a = 0.8; // Vertikale Beschleunigung
13 }

14 void draw() {
15     background(#FFFFFF);

16     text(counter, x, y);

17     if (counter > 0) { // Herunterzählen
18         counter--;
19     } else { // Bewegung bei counter = 0
20         fill(255, 0, 0);
21         frameRate(30);
22         y -= v;
23         v += a;
24     }
25 }
    
```

224



Lösung für Aufgabe 7 – Die Musterklasse

Um an Platz zu sparen, sind hier nur einige der Muster als Code hinterlegt. Falls Sie die Musterlösung eines bestimmten Musters sehen möchten, können Sie gerne die Tutor:innen fragen.

7.1 Vorbereitung

Listing 1: Lösung von 1a)

```

1 void setup() {
2     size(400, 400);
3 }
4
5 void draw() {
6     background(255);
7
8     stroke(13, 28, 137);
9     strokeWeight(2); // Dickere Linien
10
11     for(int i = -100; i < width; i += 10) { // Startet bei -100, um auch
12         // Teillinien zu zeichnen
13         line(i, 0, i + 100, height); // Zweite x-Koordinate 100pt nach rechts
14     }
15 }
    
```

Listing 2: Lösung von 1b)

```

4 void draw() {
5     background(255);
6
7     noStroke();
8     fill(198, 39, 159);
9
10    for(int i = -100; i < width; i += 75) {
11        for(int j = -100; j < height; j += 75) {
12            circle(i, j, 50);
13        }
14    }
15 }
    
```

Listing 3: Lösung von 1c)

```

4 void draw() {
5     background(98, 39, 198);
6     noStroke(); // Muster sehen ohne Umrandung oft schöner aus
7
8     for (int i = -50; i < width; i += 50) { // Sowohl horizontal als auch
9         // vertikal...
10
11         for (int j = -50; j < height; j += 50) { // ... in 50-er Schritten.
12
13             if (i % 100 == 0) { // Trifft bei jedem zweiten Schritt zu
14                 fill(39, 140, 198);
15                 square(i, j + 20, 30); // Leichte Verschiebung nach oben
16             } else {
17                 fill(29, 60, 198);
18                 square(i, j, 30);
19             }
20         }
21     }
22 }
    
```

Listing 4: Lösung von 1d)

```
4 void draw() {
5   background(255);
6   noStroke();
7
8   for(int i = -100; i < width; i += 15) {
9     fill(82, 166, 6);
10    square(i, (mouseX * i / 30f) % height, 30);
11    fill(255, 25); // zweiter Wert für alpha (auch 0-255)
12    square(0, 0, width); //Schicht weiß, welche vorherige Rechtege ü
13    bermalt
14  }
15 }
```

7.2 Muster-Designer:in

Listing 5: Lösung von 2a)

```

1 void setup() {
2     size(400, 400);
3 }
4 void draw() {
5     background(#F8D210);
6     noStroke();
7     for (int i = -200; i < width + 200; i += 10) {
8         for (int j = -200; j < width + 200; j += 10) {
9             if (i % 100 == 0 && j % 100 == 0) { // Aufrechtes Dreieck
10                pushMatrix();
11                fill(#F51720);
12                translate(i, j -i *0.3f); // Übersetzung zur Position...
13                triangle(0, 0, -25, -50, 25, -50);
14                popMatrix();
15            } else if (i % 100 == 50 && j % 100 == 50) { // Dreieck überkopf
16                pushMatrix();
17                fill(#2FF3E0);
18                translate(i, j -i *0.3f); //... inkl. Verschiebung
19                triangle(0, 0, -25, 50, 25, 50);
20                popMatrix();
21            }
22        }
23    }
24 }
    
```

Listing 6: Lösung von 2b)

```

4 void draw() {
5     background(#EAEAE0);
6     strokeWeight(15);
7     stroke(#1C4670);
8     for (int i = -100; i < height + 100; i += 75) {
9         line(0, i, width, i + 200);
10    }
11    stroke(#1DC690);
12    for (int i = -100; i < width + 100; i += 40) {
13        line(i -100, height, i, 0);
14    }
15    noStroke();
16    fill(#EAEAE0);
17    rect(100, 100, 200, 200, 25);
18    stroke(#278AB0);
19    for (int i = -100; i < height + 100; i += 50) {
20        line(0, i + 25, width, i);
21    }
22 }
23 \end{}
24 \begin{lstlisting}[caption=Lösung von 2c), firstnumber=4]
    
```

```

25 void draw() {
26     background(#e4c1f9);
27     noStroke();

28     for (int i = 0; i < width; i += 100) {
29         for (int j = 0; j < height; j += 100) {
30             fill(#ff99c8);
31             rect(i, j, random(0, 100), random(0, 100));
32             fill(#fcf6bd);
33             rect(i + 100, j, -random(0, 100), random(0, 100));
34             fill(#d0f4de);
35             rect(i + 100, j + 100, -random(0, 100), -random(0, 100));
36             fill(#a9def9);
37             rect(i, j + 100, random(0, 100), -random(0, 100));
38         }
39     }

40     noLoop();
41 }
42 \end{}

43 \begin{lstlisting}[caption=Lösung von 2d), firstnumber=4]
44 void draw() {
45     background(#50514f);
46     noStroke();

47     int offset = -40; // Offset wird bei jedem Durchlauf verschoben
48     color[] colors = new color[] { #f25f5c, #ffe066, #247ba0, #70c1b3 };

49     for (color c : colors) { // für alle 4 Farben ausführen

50         fill(c);
51         for (int k = offset; k < height; k += 40) {
52             for (int i = -100; i < width; i += random(0, 50)) {
53                 rect(i, k, random(0, 25), 40); // Zufällige Länge
54             }
55         }

56         offset += 10;
57     }

58     noLoop(); // Anti-Epilepsie, draw() nur einmal durchführen
59 }

```


208



Lösung für Aufgabe 8 – Funktionen zeichnen

8.1 Koordinatensystem

```

1 void setup() {
2     size(600, 600);
3     textAlign(RIGHT); // Text an der rechten Seite verankern
4     textSize(15); // Textgröße
5 }

6 void draw() {
7     background(255); // Weißer Hintergrund
8     translate(width / 2, height / 2); // Nullstelle zur Bildmitte
9     fill(0); // Textfarbe Schwarz
10    stroke(0); // Koordinatensystem Schwarz

11    strokeWeight(3);
12    line(-width, 0, width, 0); // x-Nullstelle
13    line(0, -height, 0, height); // y-Nullstelle

14    strokeWeight(1);
15    for (int i = -width; i < width; i += 50) {
16        line(i, -height, i, height); // Vertikale Linien
17        text(i, i - 2, 15); // Beschriftung (15 für unterhalb der Mittellinie)
18    }

19    for (int i = -height; i < height; i += 50) {
20        line(-width, i, width, i); // Horizontale Linien
21        text(i, -2, i - 2); // Beschriftung
22    }
23 }
    
```

8.2 Funktionen

```

1 void setup() {
2     size(600, 600);
3     textAlign(RIGHT); // Text an der rechten Seite verankern
4     textSize(15); // Textgröße
5 }

6 void draw() {
7     float a = mouseX - width / 2;
8     float b = mouseY - height / 2;

9     background(255); // Weißer Hintergrund
10    translate(width / 2, height / 2); // Nullstelle zur Bildmitte
11    fill(0); // Textfarbe Schwarz
12    stroke(0); // Koordinatensystem Schwarz

13    strokeWeight(3);
14    line(-width, 0, width, 0); // x-Nullstelle
15    line(0, -height, 0, height); // y-Nullstelle

16    strokeWeight(1);
17    for (int i = -width; i < width; i += 50) {
18        line(i, -height, i, height); // Vertikale Linien
19        text(i, i - 2, 15); // Beschriftung (15 für unterhalb der Mittellinie)
20    }

21    for (int i = -height; i < height; i += 50) {
22        line(-width, i, width, i); // Horizontale Linien
23        text(i, -2, i - 2); // Beschriftung
24    }

25    strokeWeight(2);
26    for (int x = -width; x < width; x++) {
27        stroke(255, 0, 0); // Rot
28        point(x, a * 2 + b); // Funktion 1
29        stroke(0, 255, 0); // Grün
30        point(x, b * sin(x / 100f)); // Funktion 2
31        stroke(0, 0, 255); // Blau
32        point(x, 10 - pow(a, x / b)); // Funktion 3
33    }
34 }
    
```

225



Lösung für Aufgabe 9 – Programme zum Leben erwecken

Hier werden keine vollständigen Lösungen gezeigt, weil die einzelnen Aufgaben im Musterlösungsblatt von Tag 1 zu finden sind.

Listing 7: Mit der Maus bewegliche Zielscheibe

```

1  int x, y, r;
2  void setup() {
3      size(400, 400);
4  }
5  void draw() {
6      background(255);
7      x = mouseX; // Position der
8      y = mouseY; // Zielscheibe
9      r = 100; // Initiale Größe
10
11     // ...
11 }
    
```

Listing 8: 3D-Würfel mit verstellbarer / ausrichtbarer Länge

```

1  void setup() {
2      size(400, 400);
3      background(255);
4      strokeWeight(2);
5      noFill();
6  }
7  void draw() {
8      background(255);
9      int oX = 50; // offset X (Verschiebung des Würfels)
10     int oY = 50; // offset Y
11     int extent = 200; // Weite und Höhe des Würfels
12
13     int fX = mouseX; // foreground X (Wie "tief" der Würfel erscheint)
14     int fY = mouseY; // foreground Y
15
16     // ...
15 }
    
```

Listing 9: Distanzmessung mit verschiebbarem Punkt

```

1  void setup() {
2      size(400, 400);
3  }
4  void draw() {
5      background(255);
6
7      int x1 = mouseX; // Punkt 1
8      int y1 = mouseY;
9
10     int x2 = 50; // Punkt 2
11     int y2 = 250;
12
13     // ...
11 }
    
```

Listing 10: Kollisionserkennung mit verschiebbarer Box

```

1 void setup() {
2     size(400, 400);
3 }
4
4 void draw() {
5     background(255);
6
6     int x1 = mouseX; // Box 1:
7     int y1 = mouseY; // Positionen
8     int dx1 = 250; // Weite
9     int dy1 = 175; // Höhe
10
10    int x2 = 150; // Box 2:
11    int y2 = 50; // Positionen
12    int dx2 = 100; // Weite
13    int dy2 = 200; // Höhe
14
14    // ...
15 }
  
```

226



Lösung für Aufgabe 10 – Browsertabs

```

1 int currentTab = 0;
2 PImage ghost, kitty, dolphin;
3 String[] websiteTitles = new String[]{"BooTube", "Flipper", "PawHub"};
4
4 void setup() {
5     size(600, 400);
6     textAlign(CENTER);
7     imageMode(CENTER);
8     textSize(20);
9
9     ghost = loadImage("ghost.png"); // Diese Bilder müssen im
10    kitty = loadImage("kitty.png"); // gleichen Ordner wie das
11    dolphin = loadImage("dolphin.png"); // Programm liegen!
12 }
13
13 void draw() {
14     // currentTab aktualisieren
15     if (mouseY < 50) { // Ist die Maus auf der Tab-Leiste?
16         if (mouseX < 200) currentTab = 0;
17         else if (mouseX < 400) currentTab = 1;
18         else currentTab = 2;
19     }
  
```

Auf der nächsten Seite geht's weiter...

```

20 // Webseite zeichnen
21 switch(currentTab) {
22   case 0: // BooTube
23     background(#212121);
24     fill(0);
25     rect(50, 100, 400, 200); // Videofläche
26
27     image(ghost, width / 2 -50, height / 2, 200, 200); // Geist
28
29     fill(255, 0, 0);
30     rect(70, 265, 360, 15); // Balken
31
32     fill(230);
33     rect(70, 310, 200, 15); // Titel
34
35     fill(0);
36     for(int y = 100; y < height; y += 50)
37       rect(470, y, 100, 40); // Videoleiste
38
39     break;
40
41   case 1: // Flipper
42     background(#004e64);
43
44     fill(#348aa7);
45     rect(0, 0, 150, height); // Seitenleiste
46
47     fill(#004e64);
48     circle(75, 125, 130); // Logo
49
50     image(dolphin, 75, 125, 100, 100); // Delphin
51
52     fill(#00a5cf);
53     for(int y = 85; y < height + 50; y += 100)
54       rect(200, y, 350, 80); // Beiträge-Leiste
55
56     break;
57
58   case 2: // PawHub
59     background(#1b1b1b);
60
61     fill(0);
62     rect(80, 100, 440, 200); // Videofläche
63
64     image(kitty, width / 2, height / 2, 150, 150); // Katze
65
66     fill(#ffa31a);
67     rect(80, 80, 80, 15); // Logo
68
69     fill(255);
70     rect(80, 310, 200, 15); // Titel
71 }

```

Auf der nächsten Seite geht's weiter...

```

56 // Tab-Leiste
57 noStroke();
58 for (int i = 0; i < 3; i++) {
59     if (currentTab == i)
60         fill(#ccc5b9); // Angewählt
61     else
62         fill(#fffcf2); // Nicht angewählt
63     rect(i *200, 0, 200, 50);
64     fill(50);
65     text(websiteTitles[i], 100 + i *200, 30); // Tab-Titel
66 }
67 }
    
```

222



Lösung für Aufgabe 11 – Fibonacci-Blumen

11.1 Blüten

```

1 void setup() {
2     size(400, 400);
3 }
4 void draw() {
5     background(255);
6     translate(width / 2, height / 2); // 0, 0 in die Bildmitte
7     int fib = fibonacci(mouseX / 30); // Fibonacci-Zahl
8     float rot = TWO_PI / fib; // Rotationsschritt
9
10    for (int i = 0; i < fib; i++) {
11        rotate(rot); // Vorherige Rotation + rot
12        ellipse(0, 100, 100 / (fib + 1) + 50, 200);
13    }
14 int fibonacci(int n) {
15     if (n == 0 || n == 1) // Abbruchbedingung
16         return n;
17     return fibonacci(n - 1) + fibonacci(n - 2); // Rekursiver Aufruf
18 }
    
```

11.2 Samen

Die Fibonacci-Methode bleibt erhalten wie oben.

```

4 void draw() {
5     background(255);
6     translate(width / 2, height / 2);
7     int fib = fibonacci(mouseX / 30);
8     float rot = TWO_PI / fib;
9     float y = 0;
10    for (int i = 0; i < 250; i++) {
11        rotate(mouseY / 100f); // Um Bruchteil von mouseY rotieren
12        circle(0, y, 7);
13        y += 0.4; // Schritt nach außen
14    }
15    for (int i = 0; i < fib; i++) {
16        rotate(rot);
17        ellipse(0, 150, 100 / (fib + 1) + 50, 150);
18    }
19 }
    
```

207



Lösung für Aufgabe 12 – Reaktionstests

12.1 Positionswechsel

Wichtig zum Verständnis: Die `millis()`-Methode gibt die Anzahl Millisekunden seit Programmstart an.

```

1 float x = 0; // Position des Kreises
2 float y = 0;
3 float radius = 60; // Größe des Kreises
4 int counter = 0; // Zähler für Treffer

5 void setup() {
6     size(500, 500);
7     textSize(32);

8     x = random(width); // Zufällige Startposition
9     y = random(height);
10 }

11 void draw() {
12     background(255);

13     float reactionTime = (millis() / 1000f) / counter;
14     float distX = abs(x - mouseX); // Abstand berechnen
15     float distY = abs(y - mouseY); // abs() für Betrag

16     if (distX < radius // x/y-Abstand kleiner als der Radius?
17         && distY < radius) {
18         x = random(width); // Neue zufällige Position
19         y = random(height);
20         counter++; // Treffer zählen
21     }

22     fill(0);
23     text("Reaktionszeit: " + reactionTime, 40, 80);

24     fill(0, 255, 0); // Grün
25     circle(x, y, radius*2); // Kreis zeichnen
26 }
    
```

12.2 Farbwechsel

Um die Farbe des Kreises beizubehalten, wird `background()` nur zu Beginn und am Ende des Spiels aufgerufen. Mit `noLoop()` beendet man das wiederholte Ausführen von der `draw()`-Methode.

```

1 float x, y;
2 float radius = 60; // Größe des Kreises
3 int counter = 0; // Zähler für Treffer
4 int targetMillis; // Zeitpunkt, ab welchem der Kreis aktiv wird
5 boolean isActive; // Kreis aktiv
6 boolean mouseInside; // Maus im Kreis
7 float reactionTime; // Reaktionszeit
8 float totalReactionTime;
    
```

Auf der nächsten Seite geht's weiter...

```

9 void setup() {
10   size(500, 500);
11   textSize(32);
12   targetMillis = millis() + round(random(1000, 5000)); // Startzeit

13   x = width / 2; // Position des Kreises
14   y = height / 2;
15   totalReactionTime = 0;

16   background(255);
17 }

18 void draw() {
19   noFill();

20   float distX = abs(x - mouseX); // Abstand berechnen
21   float distY = abs(y - mouseY); // abs() für Betrag

22   mouseInside = distX < radius && distY < radius; // Maus im Kreis?
23   isActive = millis() > targetMillis; // Kreis aktiv?

24   if (mouseInside && isActive) { // Maus im Kreis & Kreis aktiv
25     counter++; // Treffer zählen
26     totalReactionTime += millis() - targetMillis; // Reaktionszeit
27     // zusammenzählen
28     targetMillis = millis() + round(random(1000, 5000)); // Neue Zeit
29     // berechnen
30     fill(0, 255, 0);

31   } else if (mouseInside && !isActive) { // Maus im Kreis & Kreis nicht
32     // aktiv
33     targetMillis = millis() + round(random(1000, 5000));

34   } else if (isActive) { // Kreis aktiv, Maus nicht im Kreis
35     fill(0, 0, 255);

36   } else { // Kreis nicht aktiv, Maus nicht im Kreis
37     fill(255);
38   }

39   circle(x, y, radius*2); // Kreis zeichnen

40   if (counter >= 5) {
41     background(255);
42     fill(0);
43     reactionTime = (totalReactionTime / 1000f) / counter; //
44     // Reaktionszeit berechnen
45     text("Reaktionszeit: " + reactionTime, 20, height / 2);
46     noLoop();
47   }
48 }

```


223



Lösung für Aufgabe 13 – Lissajous-Figuren

13.1 Kreis zeichnen

```

1 float pos = 0; // Position des 'Stiftes'
2 void setup() {
3     size(400, 400);
4     background(255);
5 }
6 void draw() {
7     translate(width / 2, height / 2); // 0, 0 soll in der Bildmitte sein
8     circle(150 *sin(pos), 150 *sin(pos + PI / 2), 5);
9     pos += 0.01f; // Stift wandert weiter
10 }
    
```

13.2 Variationen

```

6 void draw() {
7     translate(width / 2, height / 2);
8     circle(150 *sin(pos *2), 150 *sin(pos + PI / 2), 5);
9     pos += 0.01f;
10 }
    
```

```

6 void draw() {
7     translate(width / 2, height / 2);
8     circle(150 *sin(pos *2), 150 *sin(pos *1.5), 5);
9     pos += 0.01f;
10 }
    
```

13.3 Figuren generieren lassen

```

1 void setup() {
2     size(400, 400);
3 }
4 void draw() {
5     background(255); // Nun wird die ganze Figur immer gezeichnet
6     translate(width / 2, height / 2);
7     float a = mouseX / 500f; // mouseX/Y wird verkleinert,
8     float b = mouseY / 500f; // damit die Änderungen langsamer sind
9
10    for(float t = 0; t < 200; t += 0.01f) {
11        float x = 150 *sin(a *t);
12        float y = 150 *sin(b *t);
13        point(x, y);
14    }
    
```

214



Lösung für Aufgabe 14 – Binäruhr

14.1 Binäres Modul

```

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

4 void draw() {
5     background(255);
6     drawModule(second(), 150, 100); // Aufruf der Methode
7 }

8 void drawModule(int value, int x, int y) { // value = second(), x = 150, ..
9     fill(255);
10    rect(x, y, 100, 200); // Hintergrund
11    String binaryValue = binary(value, 8); // z.B. 00101011

12    for (int i = 0; i < 8; i++) { // geht über alle Werte
13        boolean isActive = binaryValue.charAt(i) == '1';
14        if (isActive)
15            fill(0); // 1
16        else
17            fill(255); // 0

18        int xPos = x + (i < 4 ? 25 : 75); // Ternärer Operator
19        int yPos = y + (i % 4) * 50 + 25; // Vertikale Position

20        circle(xPos, yPos, 40);
21    }
22 }
    
```

14.2 Die Zeit läuft

Dank der Methode `drawModule(value, x, y)` müssen hier nur kleine Änderungen durchgeführt werden. Die Methode `drawModule(value, x, y)` bleibt unverändert und erhalten, ist hier aus Platzgründen nur nicht mehr gezeigt.

```

1 void setup() {
2     size(675, 300);
3 }

4 void draw() {
5     background(255);
6     drawModule(second(), 550, 50); // Sekunde, ganz rechts
7     drawModule(minute(), 425, 50); // Minute
8     drawModule(hour(), 300, 50); // Stunde
9     drawModule(day(), 150, 50); // Tag
10    drawModule(month(), 25, 50); // Monat, ganz links
11 }
    
```

227



Lösung für Aufgabe 15 – Mission-Control-Simulator

```

1  int secondsSinceStart, liftOffTime, lastSecond;
2  float acceleration, velocity, pos;
3  PImage rocket;

4  void setup() {
5      size(600, 400);
6      textAlign(CENTER);
7      rectMode(CENTER);
8      imageMode(CENTER);
9      textSize(32);

10     secondsSinceStart = 0;
11     lastSecond = second();
12     liftOffTime = 60;
13     acceleration = 0.01;
14     velocity = 0;
15     pos = 0;

16     rocket = loadImage("rocket.png");
17 }

18 void draw() {
19     if (lastSecond != second()) {
20         lastSecond = second();
21         secondsSinceStart++;
22     }

23     if (secondsSinceStart > liftOffTime)
24         moveRocket();

25     background(70);

26     drawBoxes();
27     drawText();
28     drawRocket();
29 }

30 void drawBoxes() {
31     stroke(0);
32     strokeWeight(2);

33     fill(#2d00f7);
34     rect(85, 25, 160, 40);
35     fill(#6a00f4);
36     rect(85, 70, 160, 40);
37     fill(#8900f2);
38     rect(85, 115, 160, 40);
39     fill(#a100f2);
40     rect(85, 160, 160, 40);
41 }
    
```

Auf der nächsten Seite geht's weiter...

```

42 void drawText() {
43     fill(255);
44     text(nf(hour(), 2) + ":"
45         + nf(minute(), 2) + ":"
46         + nf(second(), 2), 85, 38);
47
48     text("T" + nfp(secondsSinceStart - liftOffTime, 4), 85, 83);
49
50     text("v: " + nf(velocity, 3, 2), 85, 128);
51
52     text("p: " + nf(pos, 3, 2), 85, 173);
53 }
54
55 void drawRocket() {
56     strokeWeight(4);
57     fill(#c4fff9);
58     rect(width / 2 + 80, height / 2, width - 180, height - 20);
59     noStroke();
60     fill(#2b9348);
61     rect(width / 2 + 80, height / 2 + 150, width - 184, 80);
62
63     image(rocket, width / 2 + 80, height - (85 + pos), 100, 100);
64 }
65
66 void moveRocket() {
67     pos += velocity;
68     velocity += acceleration;
69 }

```