Musterlösungen für Blatt 2

231

Lösung für Aufgabe 1 – RGB-Boxen

```
size(200, 200);
noStroke();

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

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

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

232

Lösung für Aufgabe 2 – Summenformel

```
int sum = 0;

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

233

Lösung für Aufgabe 3 – Zeichenprogramm

```
void setup() {
   size(400, 400);
}

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

234

Lösung für Aufgabe 4 – Ausweichen

```
void setup() {
   size(400, 400);
}

void draw() {
   background(255);

   if(mouseX < 200) {
      circle(325, 200, 50);
   } else {
      circle(75, 200, 50);
   }
}
```

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

Listing 2: Lösung von 1b)

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

Listing 3: Lösung von 1c)

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

Listing 4: Lösung von 1d)

```
4  void draw() {
5     background(255);
6     noStroke();

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

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

```
int x, y, r;

void setup() {
   size(400, 400);
}

void draw() {
   background(255);
   x = mouseX; // Position der
   y = mouseY; // Zielscheibe
   r = 100; // Initiale Größe

   // ...

}
```

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

```
void setup() {
   size(400, 400);
   background(255);
   strokeWeight(2);
   noFill();
}

void draw() {
   background(255);
   int oX = 50; // offset X (Verschiebung des Würfels)
   int oY = 50; // offset Y
   int extent = 200; // Weite und Höhe des Würfels

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

   // ...

}
```

Listing 9: Distanzmessung mit verschiebbarem Punkt

```
void setup() {
   size(400, 400);
}

void draw() {
   background(255);

   int x1 = mouseX; // Punkt 1
   int y1 = mouseY;

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

   // ...

}
```

Listing 10: Kollisionserkennung mit verschiebbarer Box

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

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

6     int x1 = mouseX; // Box 1:
7     int y1 = mouseY; // Positionen
8     int dx1 = 250; // Weite
9     int dy1 = 175; // Höhe

10    int x2 = 150; // Box 2:
11    int y2 = 50; // Positionen
12    int dx2 = 100; // Weite
13    int dy2 = 200; // Höhe

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  void setup() {
5     size(600, 400);
6     textAlign(CENTER);
7     imageMode(CENTER);
8     textSize(20);

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 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          image(ghost, width / 2 -50, height / 2, 200, 200); // Geist
27          fill(255, 0, 0);
28          rect(70, 265, 360, 15); // Balken
29          fill(230);
30          rect(70, 310, 200, 15); // Titel
31          fill(0);
32          for(int y = 100; y < height; y += 50)
33             rect(470, y, 100, 40); // Videoleiste
34          break;
35       case 1: // Flipper
36          background(#004e64);
37          fill(#348aa7);
38          rect(0, 0, 150, height); // Seitenleiste
39          fill(#004e64);
40          circle(75, 125, 130); // Logo
41          image(dolphin, 75, 125, 100, 100); // Delphin
42          fill(#00a5cf);
43          for(int y = 85; y < height + 50; y += 100)
44             rect(200, y, 350, 80); // Beiträge-Leiste
45          break;
46       case 2: // PawHub
47          background(#1b1b1b);
48          fill(0);
49          rect(80, 100, 440, 200); // Videofläche
50          image(kitty, width / 2, height / 2, 150, 150); // Katze
51          fill(#ffa31a);
52          rect(80, 80, 80, 15); // Logo
53          fill(255);
54          rect(80, 310, 200, 15); // Titel
55    }
```

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       for (int i = 0; i < fib; i++) {
10          rotate(rot); // Vorherige Rotation + rot
11          ellipse(0, 100, 100 / (fib + 1) + 50, 200);
12      }
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  }
```

Lösung für Aufgabe 12 – Reaktionstests

12.1 Positionswechsel

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

```
float x = 0; // Position des Kreises
float y = 0;
float radius = 60; // Größe des Kreises
int counter = 0; // Zähler für Treffer

void setup() {
   size(500, 500);
   textSize(32);

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

void draw() {
   background(255);

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

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

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

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

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.

```
float x, y;
float radius = 60; // Größe des Kreises
int counter = 0; // Zähler für Treffer
int targetMillis; // Zeitpunkt, ab welchem der Kreis aktiv wird
boolean isActive; // Kreis aktiv
boolean mouseInside; // Maus im Kreis
float reactionTime; // Reaktionszeit
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
             zusammenzählen
27       targetMillis = millis() + round(random(1000, 5000)); // Neue Zeit
             berechnen
28       fill(0, 255, 0);

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

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

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

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

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

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     for(float t = 0; t < 200; t += 0.01f) {
10       float x = 150 *sin(a *t);
11       float y = 150 *sin(b *t);
12       point(x, y);
13    }
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

```
int secondsSinceStart, liftOffTime, lastSecond;
float acceleration, velocity, pos;
PImage rocket;

void setup() {
   size(600, 400);
   textAlign(CENTER);
   rectMode(CENTER);
   imageMode(CENTER);
   textSize(32);

   secondsSinceStart = 0;
   lastSecond = second();
   liftOffTime = 60;
   acceleration = 0.01;
   velocity = 0;
   pos = 0;

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

void draw() {
   if (lastSecond != second()) {
      lastSecond = second();
      secondsSinceStart++;
   }

   if (secondsSinceStart > liftOffTime)
      moveRocket();

   background(70);

   drawBoxes();
   drawText();
   drawRocket();
}

void drawBoxes() {
   stroke(0);
   strokeWeight(2);

   fill(#2d00f7);
   rect(85, 25, 160, 40);
   fill(#6a00f4);
   rect(85, 70, 160, 40);
   fill(#8900f2);
   rect(85, 115, 160, 40);
   fill(#a100f2);
   rect(85, 160, 160, 40);
}
```

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    text("T" + nfp(secondsSinceStart -liftOffTime, 4), 85, 83);

48    text("v: " + nf(velocity, 3, 2), 85, 128);

49    text("p: " + nf(pos, 3, 2), 85, 173);
50 }

51 void drawRocket() {
52    strokeWeight(4);
53    fill(#c4fff9);
54    rect(width / 2 + 80, height / 2, width -180, height -20);
55    noStroke();
56    fill(#2b9348);
57    rect(width / 2 + 80, height / 2 + 150, width -184, 80);

58    image(rocket, width / 2 + 80, height -(85 + pos), 100, 100);
59 }

60 void moveRocket() {
61    pos += velocity;
62    velocity += acceleration;
63 }
```