Musterlösungen für Blatt 4

432

Lösung für Aufgabe 1 – Zähler

```
1  void setup() {
2     Counter c = new Counter(40);
3     c.count(); // Konsole: 40
4     c.count(); // Konsole: 41
5     println("Variable next speichert: " + c.next); // Konsole: Variable next
          speichert: 42
6     c.count(); // Konsole: 42
7  }

8  class Counter {
9     int next;

10    Counter(int start) {
11       this.next = start;
12    }

13    void count() {
14       println(next);
15       next = next + 1;
16    }
17 }
```

434

Lösung für Aufgabe 2 – 2D-Bewegung

```
1  int x = 200;
2  int y = 200;

3  void setup() {
4     size(400, 400);
5  }

6  void draw() {
7     background(255);
8     circle(x, y, 20);
9  }

10 void keyPressed() {
11    if (key == CODED) {
12       switch(keyCode) {
13          case UP:
14             y -= 20;
15             break;
16          case RIGHT:
17             x += 20;
18             break;
19          case DOWN:
20             y += 20;
21             break;
22          case LEFT:
23             x -= 20;
24             break;
25       }
26    }
27 }
```

433

Lösung für Aufgabe 3 – Dark-Theme

```
1  boolean isDarkTheme = false;

2  void setup() {
3     size(400, 400);
4     textSize(40);
5  }

6  void draw() {
7     if (isDarkTheme) {
8        background(30);
9        fill(255);
10    } else {
11       background(255);
12       fill(20);
13    }
14    text("Hallo Welt!", 100, 100);
15 }

16 void mouseClicked() {
17    isDarkTheme = !isDarkTheme;
18 }
```

435

Lösung für Aufgabe 4 – Objektorientierter Kreis

```
1  void setup() {
2     size(400, 400);
3     background(255);
4     Circle c1 = new Circle(200, 200, 220);
5     c1.show();
6     c1.d = 60;
7     c1.show();
8     c1.x += 50;
9     c1.y -= 50;
10    c1.show();
11    new Circle(150, 250, 60).show();
12 }

13 class Circle {
14    int x, y, d;

15    Circle(int x, int y, int d) {
16       this.x = x;
17       this.y = y;
18       this.d = d;
19    }

20    void show() {
21       circle(x, y, d);
22    }
23 }
```

401

Lösung für Aufgabe 5 – Konfettikanone

5.1 Konfettipartikel

```
1  Particle p;

2  void setup() {
3     size(250, 250);
4     p = new Particle(width / 2, height / 2, color(255, 0, 0)); //
          Partikel in der Mitte
5  }

6  void draw() {
7     p.show(); // Methode aufrufen
8  }

9  class Particle {
10    float x, y; // Position
11    color c;
12    float vx, vy; // Geschwindigkeit

13    Particle(float x, float y, color c) { // Konstruktor
14       this.x = x;
15       this.y = y;
16       this.c = c;
17    }

18    void show() {
19       fill(c); // Farbe des Partikels anwenden
20       circle(x, y, 20);
21    }
22 }
```

5.2 Der erste Flug

```
1  float gravity = 0.08;
2  Particle p;

3  void setup() {
4     size(250, 250);
5     p = new Particle(width / 2, height / 2, color(255, 0, 0));
6  }

7  void draw() {
8     background(255);
9     p.show();
10    p.move(); // Position aktualisieren
11 }
```

Auf der nächsten Seite geht's weiter...

```
12 class Particle {
13    // ...

14    Particle(float x, float y, color c) {
15       this.x = x;
16       this.y = y;
17       this.c = c;
18       this.vy = random(-2, -0.5); // Zufällige Beschleunigung
19       this.vx = random(-2, 2);
20    }

21    // ...

22    void move() {
23       x += vx; // Geschwindigkeit auf Position
24       y += vy;
25       vy += gravity; // Schwerkraft anwenden
26    }
27 }
```

5.3 Gib mir mehr!

```
1  float gravity = 0.08;
2  int amount = 500;
3  Particle p;
4  Particle[] particles;

5  void setup() {
6     size(500, 500);
7     particles = new Particle[amount];
8  }

9  void draw() {
10    background(0);

11    for (Particle p : particles) {
12       if (p == null)
13          continue; // Nichtexistierende Partikel überspringen

14       p.show();
15       p.move();
16    }

17    if(mousePressed)
18       loadNewParticles(10);
19 }
```

Auf der nächsten Seite geht's weiter...

```
20 void loadNewParticles(int n) {
21    int c = 0;
22    for (int i = 0; i < particles.length; i++) {
23       if (particles[i] == null || particles[i].isOutside()) { // Prü
             fen, ob das Partikel überschieben werden kann
24          particles[i] = new Particle(mouseX, mouseY, color(random(255)
                , random(255), random(255)));
25          c++;
26       }
27       if (c >= n) // Genug neue Partikel erstellt
28          return;
29    }
30 }
31 class Particle {
32    // ...
33    boolean isOutside() {
34       return x < 0 || x > width || y < 0 || y > height; // gibt
             Wahrheitswert zurück
35    }
36 }
```

5.4 Wellen

```
1  float gravity = 0.08;
2  int amount = 2000;
3  Particle p;
4  Particle[] particles;
5  void setup() {
6     size(500, 500);
7     particles = new Particle[amount];
8  }
9  void draw() {
10    background(0);
11    for (Particle p : particles) {
12       if (p == null)
13          continue;
14       p.show();
15       p.move();
16    }
17    if (mousePressed)
18       loadNewParticles(10);
19 }
```

Auf der nächsten Seite geht's weiter...

```
20 void loadNewParticles(int n) {
21    int c = 0;

22    for (int i = 0; i < particles.length; i++) {

23       if (particles[i] == null || particles[i].isOutside()) {
24          particles[i] = new Particle(new PVector(mouseX, mouseY),
                color(20, 20, 100 + random(150)));
25          c++;
26       }

27       if (c >= n)
28          return;
29    }
30 }

31 class Particle {
32    PVector pos;
33    PVector vel;
34    color c;

35    Particle(PVector pos, int c) {
36       this.pos = pos;
37       this.c = c;

38       vel = PVector.fromAngle(random(TWO_PI));
39    }

40    void show() {
41       fill(c);
42       circle(pos.x, pos.y, 20);
43    }

44    void move() {
45       pos.add(vel);
46    }

47    boolean isOutside() {
48       return pos.x < 0 || pos.x > width || pos.y < 0 || pos.y > height
             ;
49    }
50 }
```

414

Lösung für Aufgabe 6 – Hau den Maulwurf

```
1  Mole[] moles;
2  int score;

3  void setup() {
4     size(300, 200);

5     score = 0;

6     textAlign(CENTER, CENTER);
7     textSize(25);

8     moles = new Mole[6];
9     moles[0] = new Mole(50, 50);
10    moles[1] = new Mole(150, 50);
11    moles[2] = new Mole(250, 50);
12    moles[3] = new Mole(50, 150);
13    moles[4] = new Mole(150, 150);
14    moles[5] = new Mole(250, 150);
15 }

16 void draw() {
17    background(255);

18    for (Mole m : moles) {
19       m.update();
20       m.show();
21    }

22    fill(0);
23    text(score, width / 2, height / 2);
24 }

25 void mouseReleased() {
26    for (Mole m : moles) {
27       if (dist(mouseX, mouseY, m.x, m.y) < 50) {
28          if (m.hit())
29          score++;
30       }
31    }
32 }

33 class Mole {

34    int x, y;
35    boolean active;
36    int counter;

37    Mole(int x, int y) {
38       active = false;
39       counter = round(random(50));
40       this.x = x;
41       this.y = y;
42    }
```

Auf der nächsten Seite geht's weiter...

```
44   void update() {
45     counter--;
46     if (counter < 0) {
47       active = !active;
48       if (active)
49         counter = round(random(10, 50));
50       else
51         counter = round(random(30, 400));
52     }
53   }

54   boolean hit() {
55     if (!active)
56       return false;

57     counter = 0;
58     return true;
59   }

60   void show() {
61     if (active)
62       fill(255, 0, 0);
63     else
64       fill(200);
65     circle(x, y, 50);
66   }
67 }
```

402

Lösung für Aufgabe 7 – Räuber-Beute-Simulation

```
1  int gridSize = 20;
2  int preyCount = 30;
3  int hunterCount = 5;

4  Animal[] prey;
5  Animal[] hunters;

6  void setup() {
7     size(700, 700);
8     frameRate(5);

9     prey = new Animal[preyCount];
10    hunters = new Animal[hunterCount];

11    for (int i = 0; i < prey.length; i++)
12       prey[i] = new Animal(new PVector(round(random(gridSize)), round(
              random(gridSize))), 1);

13    for (int i = 0; i < hunters.length; i++)
14       hunters[i] = new Animal(new PVector(round(random(gridSize)), round(
              random(gridSize))), 2);
15 }
```

Auf der nächsten Seite geht's weiter...

```
16 void draw() {
17    background(255);

18    for (Animal p : prey) {
19       p.move();
20       p.show();
21    }

22    for (Animal h : hunters) {
23       h.move();
24       h.attack(prey); // versucht, alle tiere anzugreifen
25       h.show();
26    }
27 }

28 PVector posToGrid(PVector pos) {
29    return new PVector((width / gridSize) / 2 + pos.x *(width / gridSize), (
          height / gridSize) / 2 + pos.y *(height / gridSize));
30 }

31 class Animal {
32    PVector pos;
33    int type;
34    boolean dead;

35    Animal(PVector pos, int type) {
36       this.pos = pos;
37       this.type = type;
38       this.dead = false;
39    }

40    void show() {
41       if (dead)
42       return;

43       PVector dPos = posToGrid(pos);
44       if (type == 2)
45          fill(255, 0, 0, 150);
46       else
47          fill(0, 0, 255, 150);

48       ellipse(dPos.x, dPos.y, width / gridSize, height / gridSize);
49    }
```

Auf der nächsten Seite geht's weiter...

```
50      void move() {
51          int dir = floor(random(4)); // zufällige Richtung

52          switch(dir) {
53              case 0:
54                  pos.x += 1;
55                  break;
56              case 1:
57                  pos.y += 1;
58                  break;
59              case 2:
60                  pos.x -= 1;
61                  break;
62              case 3:
63                  pos.y -= 1;
64          }

65          if (pos.x < 0) // verhindert das aus dem Spielfeld Bewegen
66              pos.x = 0;
67          if (pos.y < 0)
68              pos.y = 0;
69          if (pos.x >= gridSize)
70              pos.x = gridSize -1;
71          if (pos.y >= gridSize)
72              pos.y = gridSize -1;
73      }

74      void attack(Animal[] preyArray) {
75          for (Animal p : preyArray) {
76              if (p.pos.equals(pos))
77                  p.eat();
78          }
79      }

80      void eat() {
81          dead = true;
82      }
83  }
```

418

Lösung für Aufgabe 8 – Türme von Hanoi

8.1 Stack

```
1  class Stack {
2     int[] values;
3     int size;
4     Stack() {
5        values = new int[10];
6        size = 0;
7     }
8     void push(int i) {
9        values[size++] = i;
10    }
11    int pop() {
12       return values[--size]; // size wird verkleinert, dadurch sieht das
             Stack die anderen Werte nicht mehr
13    }
14    int[] toArray() {
15       int[] out = new int[size]; // neues Array
16       for (int i = 0; i < size; i++)
17          out[i] = values[i];
18       return out;
19    }
20 }
```

8.2 Das Spiel

```
1  Stack s1, s2, s3; // die Türme
2  Stack selected;
3  void setup() {
4     size(600, 300);
5     s1 = new Stack();
6     s2 = new Stack();
7     s3 = new Stack();
8     for (int i = 6; i > 0; i--)
9        s1.push(i); // mit absteigenden Werten befüllen
10    fill(0);
11    rectMode(CENTER);
12 }
13 void draw() {
14    background(255);
15    drawTower(s1, 100);
16    drawTower(s2, 300);
17    drawTower(s3, 500);
18 }
```

Auf der nächsten Seite geht's weiter...

```
19 void drawTower(Stack s, int x) {
20    int[] arr = s.toArray();
21    for (int i = 0; i < arr.length; i++)
22       rect(x, height -(30 + i *25), arr[i] *30, 20); // Zeichnet Scheiben

23    rect(x, height / 2, 10, height -20); // Stab
24 }

25 void mouseReleased() {
26    if (selected == null) { // Noch kein 'von' Turm
27       if (mouseX < 200)
28          selected = s1;
29       else if (mouseX < 400)
30          selected = s2;
31       else
32          selected = s3;
33    } else {

34       Stack target; // Ziel-Turm aussuchen
35       if (mouseX < 200)
36          target = s1;
37       else if (mouseX < 400)
38          target = s2;
39       else
40          target = s3;

41       target.push(selected.pop()); // Scheibe verschieben
42       selected = null; // Auswahlturm zurücksetzen
43    }
44 }
```

431

Lösung für Aufgabe 9 – Rennauto

```
1  Car car;
2  boolean up, left, right;

3  void setup() {
4     size(600, 600);
5     rectMode(CENTER);

6     car = new Car(new PVector(width / 2, height / 2), 0); // Auto in
          Bildmitte positionieren
7  }

8  void draw() {
9     fill(255, 15); // Durchsichtig!
10    rect(0, 0, width *2, height *2); // Hintergrund-Ersatz

11    car.move(up, left, right);
12    car.show();
13 }

14 void keyPressed() {
15    if (key != CODED)
16       return; // Keine Pfeiltaste

17    switch(keyCode) {
18       case UP:
19          up = true;
20          break;
21       case LEFT:
22          left = true;
23          break;
24       case RIGHT:
25          right = true;
26          break;
27    }
28 }

29 void keyReleased() {
30    if (key != CODED)
31       return; // Keine Pfeiltaste

32    switch(keyCode) {
33       case UP:
34          up = false;
35          break;
36       case LEFT:
37          left = false;
38          break;
39       case RIGHT:
40          right = false;
41          break;
42    }
43 }
```

Auf der nächsten Seite geht's weiter...

```
43 class Car {

44    PVector pos;
45    float facing;

46    Car(PVector pos, float facing) {
47       this.pos = pos;
48       this.facing = facing;
49    }

50    void move(boolean forward, boolean left, boolean right) {
51       if (forward)
52          pos.add(PVector.fromAngle(facing).setMag(2)); // Bewegung nach
              vorne

53       if (left)
54          facing -= 0.02; // Drehung

55       if (right)
56          facing += 0.02;
57    }

58    void show() {
59       pushMatrix();
60       translate(pos.x, pos.y);
61       rotate(facing);

62       fill(255, 50, 50);
63       rect(0, 0, 40, 15); // Auto

64       popMatrix();
65    }
66 }
```

407

Lösung für Aufgabe 10 – Roboterarm

```
1  Segment[] segments;
2  int selected;

3  void setup() {
4     size(600, 600);
5     selected = 0;

6     segments = new Segment[]{ // Armabschnitte
7        new Segment(0, 200, 20), new Segment(0.7, 175, 18),
8        new Segment(0.9, 150, 16), new Segment(0.4, 100, 14),
9        new Segment(0.2, 0, 0) // unsichtbares Segment ermöglicht Bewegung
             der Hand
10    };
11 }
```

Auf der nächsten Seite geht's weiter...

```
12 void draw() {
13    if (key -48 >= 0 && key -48 < segments.length)
14       selected = key -48; // Ausgewählte Taste

15    background(255);

16    pushMatrix();
17    translate(100, height);
18    rotate(PI);

19    for (Segment s : segments)
20       s.show(); // Alle Segmente anzeigen

21    triangle(0, 0, -10, 20, 10, 20); // Hand
22    popMatrix();
23 }

24 void keyPressed() {
25    segments[selected].move(keyCode); // Bewegung auf gewählten Abschnitt
          anwenden
26 }

27 class Segment {
28    float angle;
29    float l, w;

30    Segment(float angle, float l, float w) {
31       this.angle = angle;
32       this.l = l;
33       this.w = w;
34    }

35    void move(int key) { // Arm bewegen
36       if (key == LEFT)
37          angle -= 0.01;
38       else if (key == RIGHT)
39          angle += 0.01;
40    }

41    void show() {
42       rotate(angle);
43       rect(-w / 2, 0, w, l);
44       translate(0, l);
45    }
46 }
```

408

Lösung für Aufgabe 11 – Kanonenspiel

```
Cannon cannon;
Ball ball;
Target target;

float gravity = 0.1;
float initialBallSpeed = 10;
int score;

void setup() {
   size(600, 500);
   cannon = new Cannon(new PVector(50, height -50));
   target = new Target();
   score = 0;

   fill(0);
   textSize(50);
   textAlign(CENTER, CENTER);
}

void draw() {
   background(255);
   cannon.update();
   cannon.show();
   target.show();

   text(score, width / 2, height / 2);

   if (ball != null) {
      ball.update();
      ball.show();

      if (PVector.dist(ball.pos, target.pos) < 50) {
         score++;
         target = new Target();
         ball = null;
      }

      if (ball == null || ball.pos.x > width || ball.pos.y > height)
      ball = null;
   }
}

void mouseReleased() {
   if (ball != null)
   return;

   PVector ballVel = PVector.fromAngle(TWO_PI -cannon.facing);
   ballVel.setMag(initialBallSpeed);
   ball = new Ball(cannon.pos.copy(), ballVel);
}
```

Auf der nächsten Seite geht's weiter...

```
41 class Cannon {
42    PVector pos;
43    float facing = QUARTER_PI;

44    Cannon(PVector pos) {
45       this.pos = pos;
46    }

47    void update() {
48       if (pos.x < mouseX && mouseY < pos.y) {
49          facing = acos((mouseX -pos.x) / PVector.dist(pos, new PVector(
                mouseX, mouseY)));
50       }
51    }

52    void show() {
53       pushMatrix();
54       translate(pos.x, pos.y);
55       rotate(TWO_PI -facing);

56       rect(0, 0, 50, 10);

57       popMatrix();
58    }
59 }

60 class Ball {
61    PVector pos;
62    PVector vel;

63    Ball(PVector pos, PVector vel) {
64       this.pos = pos;
65       this.vel = vel;
66    }

67    void update() {
68       pos.add(vel);
69       vel.add(0, gravity);
70    }

71    void show() {
72       circle(pos.x, pos.y, 25);
73    }
74 }

75 class Target {
76    PVector pos;

77    Target() {
78       this.pos = new PVector(width -50, random(50, height -50));
79    }

80    void show() {
81       circle(pos.x, pos.y, 40);
82    }
83 }
```

416

Lösung für Aufgabe 12 – Förderband

```
Box[] b;
ProductionLine [] pl;

void setup() {
   size(500, 500);
   b = new Box[2]; // Boxen
   pl = new ProductionLine[4]; // Förderbänder

   b[0] = new Box(100, 400, 40);
   b[1] = new Box(300, 400, 75);

   pl[0] = new ProductionLine(100, 100, 200, 30, 2);
   pl[1] = new ProductionLine(300, 100, 30, 200, 3);
   pl[2] = new ProductionLine(130, 300, 200, 30, 0);
   pl[3] = new ProductionLine(100, 130, 30, 200, 1);
}

void draw() {
   background(255);

   for (int i = 0; i < b.length; i++) {
      for (int j = 0; j < pl.length; j++) {
         pl[j].show();
         b[i].moveOnLine(pl[j]);
      }
   }

   for (int i = 0; i < b.length; i++) {
      b[i].updateIfDragged();
      b[i].show();
   }
}

class ProductionLine {
   int x, y, w, h; // position, breite, länge

   int dir; //d=0 right, d=1 up, d=2 right, d=3 down

   ProductionLine(int x, int y, int w, int h, int dir) {
      this.x = x;
      this.y = y;
      this.w = w;
      this.h = h;
      this.dir = dir;
   }

   void show() { // Umrandung und Fließband-Rillen
      fill(150);
      rect(x, y, w, h);

      if (dir == 1 || dir == 3) {
         for (int i = 1; i < h / 10; i++)
            line(x + 5, y + i *10, x + w -5, y + i *10);

      } else if (dir == 0 || dir == 2) {
         for (int i = 1; i < w / 10; i++)
            line(x + i *10, y + 5, x + i *10, y + h -5);

      } else
         print("Invalid Direction!");
   }
}
```

```
49 class Box {
50    int x;
51    int y;
52    int w;
53    boolean selected; // Box mit drag & drop ausgewählt

54    Box(int x, int y, int w) {
55       this.x = x;
56       this.y = y;
57       this.w = w;
58    }

59    void show() {
60       fill(240, 220, 160);
61       rect(x, y, w, w);
62       line(x, y, x + w, y + w);
63       line(x + w, y, x, y + w);
64    }

65    void updateIfDragged() {
66       selected = false; // Zurücksetzen

67       if (!(mouseX >= this.x && mouseX <= this.x + this.w &&
68             mouseY >= this.y && mouseY <= this.y + this.w && mousePressed))
69          return; // Box nicht ausgewählt

70       selected = true;
71       int xDist = mouseX -pmouseX;
72       int yDist = mouseY -pmouseY;
73       this.x += xDist; // Bewegen
74       this.y += yDist;
75    }

76    void moveOnLine(ProductionLine p) {
77       if (selected)
78          return; // ausgewählt, also nicht vom Fließband bewegen lassen

79       if (this.x > p.x + p.w
80             || this.x + this.w < p.x
81             || this.y > p.y + p.h
82             || this.y + this.w < p.y)
83          return; // Band berührt nicht die Box

84       switch (p.dir) {
85          case 0:
86             this.x-=1;
87             break;
88          case 1:
89             this.y-=1;
90             break;
91          case 2:
92             this.x+=1;
93             break;
94          case 3:
95             this.y+=1;
96             break;
97       }
98    }
99 }
```

430

Lösung für Aufgabe 13 – Fraktale Bäume

```
1  float angle = 0.005;

2  void setup() {
3     size(800, 600);
4  }

5  void draw() {
6     background(51);
7     stroke(255);
8     translate(width / 2, height);
9     branch(200);
10 }

11 void branch(float len) {
12    line(0, 0, 0, -len);
13    translate(0, -len);

14    if (len > 2) {
15       pushMatrix();

16       rotate(angle *frameCount);
17       branch(0.67 *len);

18       popMatrix();
19       pushMatrix();

20       rotate(-angle *frameCount);
21       branch(0.67 *len);

22       popMatrix();
23    }
24 }
```

422

Lösung für Aufgabe 14 – Logische Schaltkreise

14.1 Logisches Modul

```
1  class Module {
2     PVector pos;
3     Module[] outgoing; // Module, welche von diesem Modul ein Signal
          erhalten
4     Module(PVector pos) {
5        this.pos = pos;
6        this.outgoing = new Module[10];
7     }
8  }
```

14.2 Die Klassenerweiterung

```
1  class Module {
2     PVector pos;
3     Module[] outgoing;
4     Module(PVector pos) {
5        this.pos = pos;
6        this.outgoing = new Module[10];
7     }
8     void input(boolean input) {
9        print("Modul kann keine Inputs aufnehmen!");
10    }
11    void reset() {
12    }
13 }
14 class OrGate extends Module {
15    boolean val1, val2;
16    int counter;
17    OrGate(PVector pos) {
18       super(pos);
19       counter = 0;
20    }
21    void input(boolean input) {
22       if (counter == 0)
23          val1 = input; // erstes Input
24       else if (counter == 1)
25          val2 = input; // Zweites Input
26       else
27          print("Gate kann keine weiteren Inputs aufnehmen!");
28       counter++;
29    }
30    void reset() {
31       val1 = false;
32       val2 = false;
33       counter = 0;
34    }
35 }
```

14.3 Visualisierung

```
Module m1, m2, m3;

void setup() {
   size(750, 750);
   m1 = new OrGate(new PVector(100, 100));
   m2 = new OrGate(new PVector(500, 200));
   m3 = new OrGate(new PVector(400, 600));

   rectMode(CENTER);
   textAlign(CENTER);
   textSize(20);
}

void draw() {
   background(255);

   m1.show();
   m2.show();
   m3.show();
}

class Module {

   // ...

   void show() {
      fill(255);
      square(pos.x, pos.y, 50);
   }
}

class OrGate extends Module {

   // ...

   void show() {
      fill(255);
      square(pos.x, pos.y, 50);

      fill(0);
      text("OR", pos.x, pos.y);
   }
}
```

14.4 Mehr Tricksereien und Input

```
1  Module m1, m2, m3;

2  void setup() {
3     size(750, 750);
4     m1 = new Input(new PVector(100, 100), true);
5     m2 = new OrGate(new PVector(500, 200));
6     m3 = new OrGate(new PVector(400, 600));

7     m1.outgoing[0] = m2;

8     rectMode(CENTER);
9     textAlign(CENTER, CENTER);
10    textSize(20);
11 }

12 void draw() {
13    // ...
14 }

15 class Module {

16    // ...

17    void show() {
18       if (output)
19          fill(255, 100, 100); // Signal an
20       else
21          fill(255); // Signal aus

22       square(pos.x, pos.y, 50);

23       for (Module m : outgoing) {
24          if (m == null)
25             continue;
26          m.show();
27          line(pos.x, pos.y, m.pos.x, m.pos.y);
28       }
29    }

30    void send(boolean output) { // Output schicken
31       this.output = output;

32       for (Module m : outgoing) {
33          if (m != null)
34             m.input(output);
35       }
36    }

37    void calculateSignal() {
38       print("Modul kann kein Signal versenden!");
39    }
40 }
```

Auf der nächsten Seite geht's weiter...

```
42 class OrGate extends Module {
43    // ...
44    void input(boolean input) {
45       // ...
46       calculateSignal();
47    }
48    // ...
49    void calculateSignal() {
50       if (counter == 2) // Nur einmal schicken!
51          send(val1 || val2);
52    }
53 }
54 class Input extends Module {
55    boolean output;
56    Input(PVector pos, boolean output) {
57       super(pos);
58       this.output = output;
59    }
60    void calculateSignal() {
61       send(output);
62    }
63    void show() {
64       super.show();
65       fill(0);
66       text(output ? 1 : 0, pos.x, pos.y);
67    }
68 }
```

14.5 Der erste Schaltkreis

```
1  Module[] baseModules;

2  void setup() {
3     size(500, 500);
4     baseModules = new Module[10];
5     baseModules[0] = new Input(new PVector(100, 100), false);
6     baseModules[1] = new Input(new PVector(100, 200), true);
7     baseModules[2] = new Input(new PVector(100, 300), false);
8     baseModules[3] = new Input(new PVector(100, 400), false);

9     Module or = new OrGate(new PVector(200, 150));
10    Module or2 = new OrGate(new PVector(200, 250));
11    Module or3 = new OrGate(new PVector(200, 350));

12    Module or4 = new OrGate(new PVector(300, 200));
13    Module or5 = new OrGate(new PVector(300, 300));

14    baseModules[0].outgoing[0] = or;
15    baseModules[1].outgoing[0] = or;
16    baseModules[1].outgoing[1] = or2;
17    baseModules[2].outgoing[0] = or2;
18    baseModules[2].outgoing[1] = or3;
19    baseModules[3].outgoing[0] = or3;

20    or.outgoing[0] = or4;
21    or2.outgoing[0] = or4;
22    or2.outgoing[1] = or5;
23    or3.outgoing[0] = or5;

24    rectMode(CENTER);
25    textAlign(CENTER, CENTER);
26    textSize(20);
27 }

28 void draw() {
29    background(255);

30    for (Module m : baseModules) {
31       if (m != null)
32          m.reset(); // Alles zurücksetzen
33    }

34    for (Module m : baseModules) {
35       if (m != null)
36          m.calculateSignal(); // Alle Signale schicken
37    }

38    for (Module m : baseModules) {
39       if (m != null)
40          m.show(); // Alles malen
41    }
42 }
```

Auf der nächsten Seite geht's weiter...

```
46 class Module {

47    PVector pos;
48    Module[] outgoing;
49    boolean output;

50    Module(PVector pos) {
51       this.pos = pos;
52       this.outgoing = new Module[10];
53    }

54    void input(boolean input) {
55       print("Modul kann keine Inputs aufnehmen!");
56    }

57    void reset() {
58       for (Module m : outgoing) {
59          if (m != null)
60             m.reset();
61       }
62    }

63    void show() {
64       if (output)
65          fill(255, 100, 100);
66       else
67          fill(255);

68       square(pos.x, pos.y, 50);

69       for (Module m : outgoing) {
70          if (m == null)
71             continue;

72          m.show();
73          line(pos.x, pos.y, m.pos.x, m.pos.y);
74       }
75    }

76    void send(boolean output) {
77       this.output = output;

78       for (Module m : outgoing) {
79          if (m != null)
80             m.input(output);
81       }
82    }

83    void calculateSignal() {
84       print("Modul kann kein Signal versenden!");
85    }
86 }
```

Auf der nächsten Seite geht's weiter...

```
89  class OrGate extends Module {

90      boolean val1, val2;
91      int counter;

92      OrGate(PVector pos) {
93          super(pos);
94          counter = 0;
95      }

96      void input(boolean input) {
97          if (counter == 0)
98              val1 = input;
99          else if (counter == 1)
100             val2 = input;
101         else
102             print("Gate kann keine weiteren Inputs aufnehmen!");

103         counter++;
104         calculateSignal();
105     }

106     void reset() {
107         super.reset();

108         val1 = false;
109         val2 = false;
110         counter = 0;
111     }

112     void show() {
113         super.show();

114         fill(0);
115         text("OR", pos.x, pos.y);
116     }

117     void calculateSignal() {
118         if (counter == 2)
119             send(val1 || val2);
120     }
121 }

122 class Input extends Module {

123     boolean output;

124     Input(PVector pos, boolean output) {
125         super(pos);
126         this.output = output;
127     }

128     void calculateSignal() {
129         send(output);
130     }

131     void show() {
132         super.show();

133         fill(0);
134         text(output ? 1 : 0, pos.x, pos.y);
135     }
136 }
```