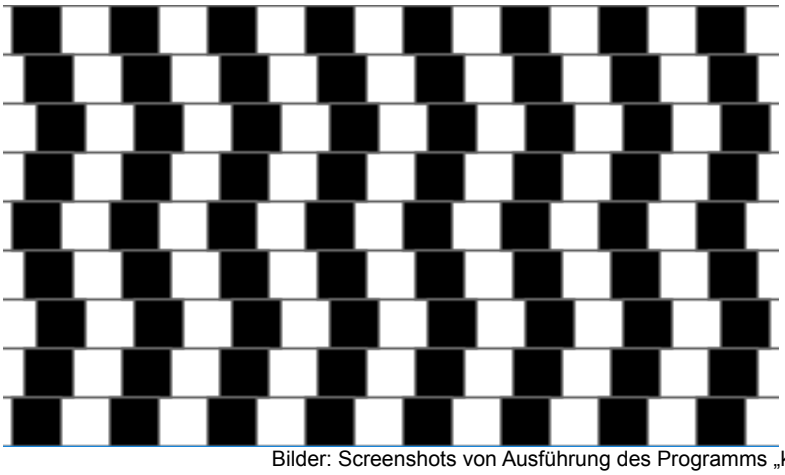
Optische Täuschungen

Eine optische Täuschung ist eine Wahrnehmungstäuschung des Sehsinns. Sie kann viele Aspekte des Sehens betreffen. Es gibt z.B. Tiefenillusionen, Farbillusionen, geometrische Illusionen und Bewegungsillusionen[1].

Vielleicht habt ihr im Mathematikunterricht schon einmal eine optische Täuschung selbst hergestellt, als ihr in Klasse 5 das Zeichnen von parallelen und orthogonalen Linien üben solltet. Es gibt nämlich solche, die aus parallelen (und senkrechten) Geraden bestehen, aber ganz krumm aussehen, wie z.B. die sogenannte Münsterberg- oder auch Kaffeehaus-Täuschung. Diese werden wir uns genauer anschauen und sie dann selbst programmieren.



Bilder: Screenshots von Ausführung des Programms „kaffeehaus“ (Eisenmann)

1. *Überprüfe zunächst mit deinem Geodreieck, ob es sich tatsächlich um parallele Geraden handelt.*
2. *Speichere ein neues Programm unter „kaffeehaus“ ab.*
3. *Schreibe zunächst den Rahmen der setup-Methode, gib die Größe des Fensters (640x400) an und lasse den Hintergrund weiß „färben“.*

```
void setup(){
  size(640, 360);
  background(255); // Hintergrund weiß
}
```

4. *Schreibe eine Methode parallelen(), die zehn graue Parallelen im Abstand von 40 Pixel zeichnet. Nutze dazu eine Zählschleife. Vergleiche dein Ergebnis mit der ausliegenden Lösung.*

```
void parallelen(){
  stroke(125);
  for (int i=0; i<=9; i++){
    line(0,5+i*40, 640, 5+i*40);
  }
}
```

1 Siehe Seite „Optische Täuschung“. In: Wikipedia, Die freie Enzyklopädie. Bearbeitungsstand: 11. September 2018, 16:14 UTC. URL: *https://de.wikipedia.org/w/index.php?title=Optische_T%C3%A4uschung&oldid=180828619* (Abgerufen: 14. September 2018, 15:58 UTC)

Du siehst hier wieder diese grau geschriebenen Hinweise. Diese heißen **Kommentare**. Sie sind dazu da, um dein Programm lesbarer zu gestalten. So kann auch jemand anderes schnell erkennen, was in deinem Programm passiert.

Ein Kommentar wird immer nach zwei Schrägstrichen geschrieben, wenn er in eine Zeile passt. Dein Computer ignoriert beim Übersetzen deines Programms die Kommentare. Bsp.:

```
background(255); // Hintergrund weiß
parallelen();  // Aufruf der Methode
```

Es gibt auch noch eine weitere Möglichkeit, einen Kommentar zu kennzeichnen.

```
/* auch das ist ein Kommentar, der
   sogar über mehrere Zeilen
   gehen kann */
```

Ist etwas zwischen /* und */ eingeschlossen, wird es auch beim Übersetzen ignoriert.

```
background(255); // Hintergrund weiß
parallelen();  // Aufruf der Methode
```

So kannst du auch mal einen Programmteil „auskommentieren“, wenn etwas nicht funktioniert, ohne ihn gleich löschen zu müssen.

5. *Ergänze in deinem Programm sinnvolle Kommentare. Du kannst die aus der ausliegenden Lösung als Beispiel nehmen.*

Individuelle Lösungen

Den Abstand der Linien haben wir auf 40 festgelegt. Diesen Wert brauchen wir auch später als Seitenlänge der vielen Quadrate, die wir noch zeichnen lassen werden. Sollten wir auf die Idee kommen, die Quadrate kleiner oder größer haben zu wollen, müssten wir an vielen Stellen etwas ändern.

Deshalb nutzen wir eine **Variable**, die im ganzen Programm bekannt ist. Im Gegensatz zu der Zählvariable `i`, die nur in der Schleife (**lokal**) bekannt ist, ist die neue Variable überall (**global**) bekannt. Wir nennen sie `s` für Seitenlänge und geben ihr den Wert 40. Auch diese Variable ist eine ganze Zahl, sie bekommt deshalb auch den Typ `int`.

Die Deklaration globaler Variablen (auch möglich mit Initialisierung) wird an den Anfang des Programms geschrieben.

Es kommt immer zuerst der Datentyp (z.B. int für eine ganze Zahl), dann der Name der Variable, dann ein Strichpunkt oder schon die Initialisierung der Variable.

Beispiele: `int start;` oder: `int start = 1;`

6. *Verändere deinen Programmcode, indem du die Variable s für die Seitenlänge der Quadrate bzw. den Abstand der Parallelen einführst und nutzt. Vergleiche dein Ergebnis mit der ausliegenden Lösung.*

Lösung siehe unten.

Blatt 2 – Die Quadrate...

Als Nächstes füllen wir die Reihen mit Quadraten.

7. *Schreibe den Rahmen einer Methode zeichneQuadrate(). Stelle darin zunächst schwarz als Füllfarbe ein. (Die Linienfarbe bleibt grau.)*

```
void zeichneQuadrate(){
  fill(0);
}
```

In einer Reihe sollen jeweils acht Quadrate gezeichnet werden, die wieder den Abstand der eigenen Seitenlänge haben.

8. *Programmiere das Zeichnen der Quadrate in der ersten Reihe. Nutze wieder eine Zählschleife und nutze auch die Variable s für die Seitenlänge. Starte dein Programm und ändere deinen Programmcode ab, wenn etwas noch nicht stimmt. (Die Lösung liegt aus.)*

```
void zeichneQuadrate(){
  fill(0);  // Füllfarbe schwarz
  // fülle die erste Reihe mit Quadraten
  for (int i=0; i<=7; i++){
    rect(10+2*s*i,5,s,s);
  }
}
```

In jeder Reihe startet das erste Quadrat aber an einer anderen Stelle. In der ersten, mittleren und letzten Reihe bei 10 Pixeln, jeweils daneben bei 20 und in der dritten und siebten Reihe bei 30 Pixel.

9. *Überlege dir, was du für die nächsten Reihen ändern musst.*

   Die Seitenlänge bleibt gleich, zur y-Koordinate muss für jede Reihe einmal die Seitenlänge s addiert werden, die x-Koordinate lässt sich wie bei Reihe 1 berechnen, nur dass der Startwert jeweils wechselt.

10. *Programmiere das Zeichnen der Quadrate in der zweiten Reihe.*
11. *Programmiere das Zeichnen der Quadrate in der dritten Reihe.*

Vergleiche mit der ausliegenden Lösung (s.u.) und hol dir dann das nächste Blatt.

Es ist sehr umständlich, für jede Reihe eine neue Schleife programmieren zu müssen. Eigentlich wiederholen wir ja wieder neun Mal das Zeichnen einer Reihe von Quadraten.

12. Gib an, was gleich bleibt und überlege dir, was sich ändert. Fülle dazu folgende Tabelle weiter aus:

Reihe	x	y	b	h
Erste Reihe	10 + 2 * s * i	5	s	s
Zweite Reihe	20 + 2 * s * i	5 + s	s	s
Dritte Reihe	*30 + 2 * s * i*	*5 + 2 * s*	*s*	*s*
Vierte Reihe	*20 + 2 * s * i*	*5 + 3 * s*	*s*	*s*
Fünfte Reihe	*10 + 2 * s * i*	*5 + 4 * s*	*s*	*s*
Sechste Reihe	*20 + 2 * s * i*	*5 + 5 * s*	*s*	*s*
Siebte Reihe	*30 + 2 * s * i*	*5 + 6 * s*	*s*	*s*
Achte Reihe	*20 + 2 * s * i*	*5 + 7 * s*	*s*	*s*
Neunte Reihe	*10 + 2 * s * i*	*5 + 8 * s*	*s*	*s*

Wir ignorieren zunächst einmal die unterschiedlichen Abstände zum linken Rand und nehmen bei allen Reihen die Zahl 10. Um die Änderung kümmern wir uns später.

Dann ändert sich von Reihe zu Reihe nur die y-Koordinate.

13. Überlege dir, wie du mit einer weiteren Zählvariable j auf die y-Koordinaten kommst. Vergleiche deine Idee mit der ausliegenden Lösung.

Wir bekommen die Quadrate einer Reihe mithilfe einer Zählschleife, die sich in der vereinfachten Form nur in der y-Koordinate unterscheidet.

Da wir aber neun solcher Reihen brauchen, nutzen wir eine weitere Zählschleife, die neunmal unsere bisherige Zählschleife durchläuft.

Sind zwei oder mehr Zählschleifen ineinander verschachtelt, so nennt man das eine **verschachtelte Zählschleife**.

Es gibt eine äußere Schleife mit der Zählvariablen j und eine innere Schleife mit der Zählvariablen i.

Die verschachtelte Zählschleife sieht jetzt folgendermaßen aus:

```
for (int j=0; j<=8; j++){
  for (int i=0; i<=7; i++){
    rect(10+2*s*i,5+j*s,s,s);
  }
}
```

14. *Schreibe eine neue Methode zeichneAlleQuadrate(), die diese verschachtelte Zählschleife nutzt. Teste dann dein Programm.*

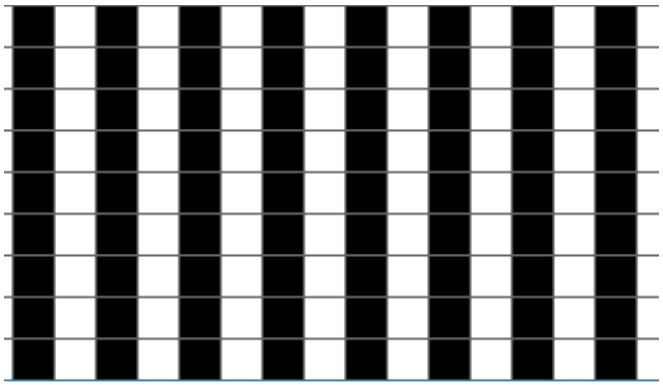
```
void zeichneAlleQuadrate(){
  fill(0); // Füllfarbe schwarz
  for (int j=0; j<=8; j++){
    for (int i=0; i<=7; i++){
      rect(10+2*s*i,5+j*s,s,s);
    }
  }
}


void setup(){
  size(640, 400);
  background(255); // Hintergrund weiß
  parallelen();  // Aufruf der Methode zum Zeichnen der Parallelen
  zeichneAlleQuadrate();
  //zeichneQuadrate(); // Aufruf der Methode zum Zeichnen der Quadrate
}
```

15. *Überlege dir jetzt noch einmal ganz genau, was Schritt für Schritt passiert. Wenn es dir noch nicht klar ist, hole dir zur Hilfe das **Extrablatt** und fülle es Schritt für Schritt aus.*

Es wird neunmal nacheinander die innere Schleife aufgerufen, die jeweils acht Quadrate nebeneinander zeichnet.

Leider ist es jetzt keine optische Täuschung mehr, weil wir links immer nur 10 Pixel Abstand lassen.



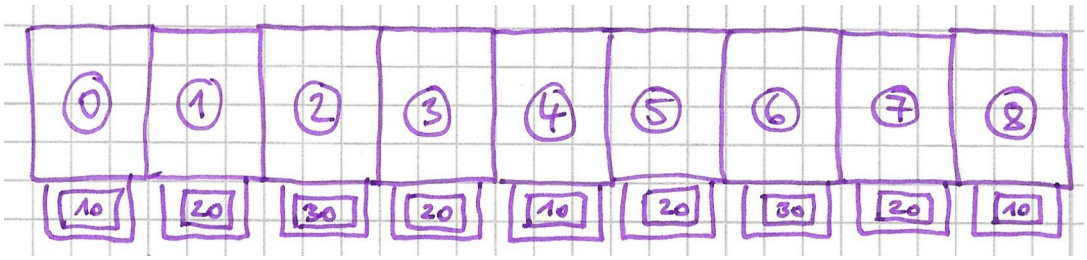
Statt dem Wert 10 als Abstand zum linken Rand, hätten wir gerne in jeder Reihe einen anderen Wert. Da sich der Wert mal erhöht, mal erniedrigt, können wir ihn nicht durch die Schleifenvariable j darstellen, wie es bei dem Abstand nach oben funktioniert hat.

Wir nutzen hier ein sogenanntes **Feld** (**Array**), das mehr als einen Wert gleichzeitig speichern kann. Du hast eine solche Datenstruktur schon im letzten Jahr kennen gelernt.

Zur Erinnerung:

Denken wir bei einer Variable an das Schachtelmodell (für jede Variable gibt es eine Schachtel, in der der aktuelle Wert der Variable liegt), können wir uns hier im Vergleich mehrere aneinandergeklebte Schachteln, die durchnummeriert sind, vorstellen.

In unserem Fall würde in den Schachteln jeweils der Abstand zum linken Rand als Wert liegen.



In Java müssen wir das Feld bekannt machen und können es auch gleich mit den Werten füllen.

```
int[] abstaende = {10, 20, 30, 20, 10, 20, 30, 20, 10};
```

Die beiden eckigen Klammern nach dem Datentyp `int` zeigen an, dass `abstaende` ein Feld aus ganzen Zahlen ist.

Der Zugriff auf ein Element des Feldes geht auch mit eckigen Klammern: z.B. liefert `abstaende[0]` den Wert 10.

16. Ändere dein Programm so ab, dass statt dem Abstand 10 der jeweilige Wert des Feldelementes genutzt wird (alte Zeile: `rect(~~10~~+2*s*i,5+j*s,s,s);`*). Teste dein Programm und vergleiche mit der Lösung.*

In jeder Reihe muss statt dem Wert 10 der entsprechende Wert aus dem Feld genommen werden: in der ersten Reihe (j = 0): abstaende[0], in der zweiten Reihe (j = 1): abstaende[1], usw.

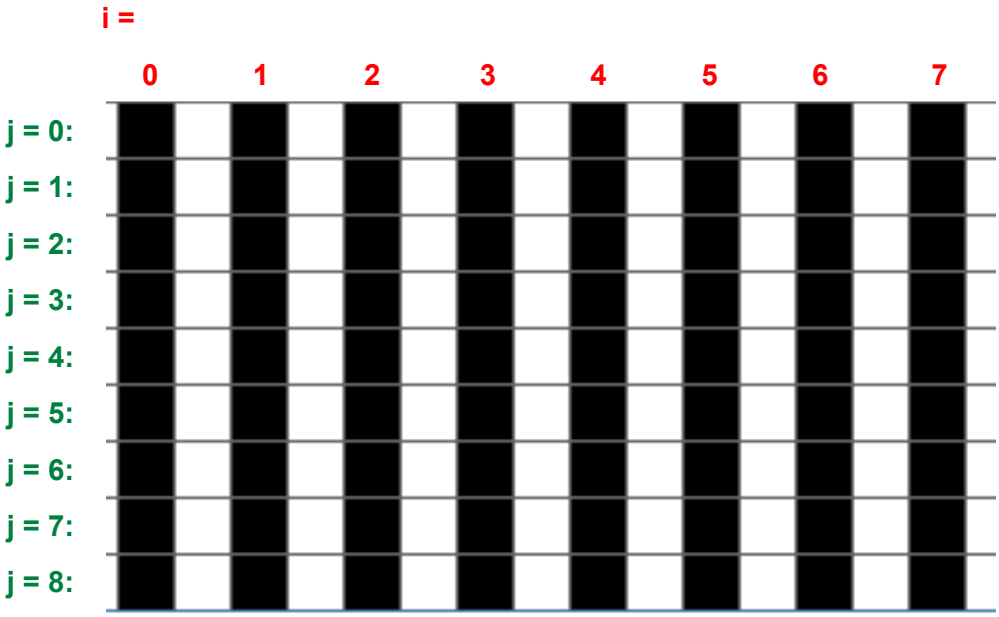
Man kann also den Wert 10 durch abstaende[j] ersetzen.

VERSCHACHTELTE ZÄHLSCHLEIFE - LOESUNG

```
for (int j=0; j<=8; j++){
  for (int i=0; i<=7; i++){
    rect(10+2*s*i,5+j*s,s,s);
  }
}
```

Schreibe hier Schritt für Schritt auf, was in der verschachtelten Zählschleife passiert und überlege dir auch, was Schritt für Schritt gezeichnet wird.

Äußere Schleife:	Innere Schleife:	
j = 0:	**i = 0:**	rect(10 + 2 * s * **0**, 5 + **0** * s, s, s), also rect(10, 5, s, s);
	i = 1:	rect(10 + 2 * s * **1**, 5 + **0** * s, s, s), also rect(10 + 2s, 5, s, s);
	i = 2:	rect(10 + 2 * s * **2**, 5 + **0** * s, s, s), also rect(10 + 4s, 5, s, s);
	i = 3:	rect(10 + 2 * s * **3**, 5 + **0** * s, s, s), also rect(10 + 6s, 5, s, s);
	i = 4:	rect(10 + 2 * s * **4**, 5 + **0** * s, s, s), also rect(10 + 8s, 5, s, s);
	i = 5:	rect(10 + 2 * s * **5**, 5 + **0** * s, s, s), also rect(10 + 10s, 5, s, s);
	i = 6:	rect(10 + 2 * s * **6**, 5 + **0** * s, s, s), also rect(10 + 12s, 5, s, s);
	i = 7:	rect(10 + 2 * s * **7**, 5 + **0** * s, s, s), also rect(10 + 14s, 5, s, s);
j = 1:	**i = 0:**	rect(10 + 2 * s * **0**, 5 + **1** * s, s, s), also rect(10, 5 + s, s, s);
	i = 1:	rect(10 + 2 * s * **1**, 5 + **1** * s, s, s), also rect(10 + 2s, 5 + s, s, s);
	i = 2:	rect(10 + 2 * s * **2**, 5 + **1** * s, s, s), also rect(10 + 4s, 5 + s, s, s);
	…	
	i = 7:	rect(10 + 2 * s * **7**, 5 + **1** * s, s, s), also rect(10 + 14s, 5 + s, s, s);
j = 2:	**i = 0:**	rect(10 + 2 * s * **0**, 5 + **2** * s, s, s), also rect(10, 5 + 2s, s, s);
	i = 1:	rect(10 + 2 * s * **1**, 5 + **2** * s, s, s), also rect(10 + 2s, 5 + 2s, s, s);
	…	
	i = 7:	rect(10 + 2 * s * **7**, 5 + **2** * s, s, s), also rect(10 + 14s, 5 + 2s, s, s);
…		
j = 8:	**i = 0:**	rect(10 + 2 * s * **0**, 5 + **8** * s, s, s), also rect(10, 5 + 8s, s, s);
	i = 1:	rect(10 + 2 * s * **1**, 5 + **8** * s, s, s), also rect(10 + 2s, 5 + 8s, s, s);
	i = 2:	rect(10 + 2 * s * **2**, 5 + **8** * s, s, s), also rect(10 + 4s, 5 + 8s, s, s);
	…	
	i = 7:	rect(10 + 2 * s * **7**, 5 + **8** * s, s, s), also rect(10 + 14s, 5 + 8s, s, s);


i =
0
1
2
3
4
5
6
7
j = 0:
j = 1:
j = 2:
j = 3:
j = 4:
j = 5:
j = 6:
j = 7:
j = 8:

Mögliche Lösungen:

Teil 1:

```
void parallelen(){ // zeichnet 10 graue Parallelen
  stroke(125); // Linienfarbe Graustufe
  for (int i=0; i<=9; i++){
    line(0,5+i*40, 640, 5+i*40);
  }
}

void setup(){
  size(640, 400);
  background(255); // Hintergrund weiß
  parallelen();  // Aufruf der Methode
}
```

Teil 1 mit Variable:

```
// globale Variablen
int s = 40; // Seitenlänge der Quadrate bzw. Abstand der Parallelen

void parallelen(){ // zeichnet 10 graue Parallelen
  stroke(125); // Linienfarbe Graustufe
  for (int i=0; i<=9; i++){
    line(0,5+i*s, 640, 5+i*s);
  }
}

void setup(){
  size(640, 400);
  background(255); // Hintergrund weiß
  parallelen();  // Aufruf der Methode
}
```

Teil 2 – Vorüberlegungen für Aufgabe 8:

Alle Quadrate sind gleich groß, d.h. sowohl Breite als auch Höhe haben den Wert s (Variable der Seitenlänge).

In der ersten Reihe haben auch alle Quadrate die gleiche y-Koordinate des Startpunktes. Wenn wir einen kleinen Abstand nach oben haben wollen, ist das z.B. 5.

Nur die x-Koordinate des Startpunktes ändert sich.

Quadrat 1 in der ersten Reihe: x = 10 (Vorgabe in der Aufgabe)

Das nächste Quadrat startet zwei Seitenlängen weiter rechts, da ja immer ein schwarzes Quadrat gezeichnet wird und dann ein Quadrat weiß bleibt.

Quadrat	1	2	3	4	
x-Koordinate	10	10 + 2*s	10+(2*s)*2	10+(2*s)*3	10+(2*s)*i
Zählvariable i	0	1	2	3	i

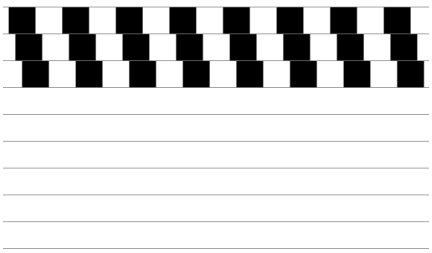
```
void zeichneQuadrate(){
  fill(0);  // Füllfarbe schwarz
  // fülle die erste Reihe mit Quadraten
  for (int i=0; i<=7; i++){
    rect(10+2*s*i,5,s,s);
  }
}
```

Teil 2:

```
void zeichneQuadrate(){
  fill(0);  // Füllfarbe schwarz
  // fülle die erste Reihe mit Quadraten
  for (int i=0; i<=7; i++){
    rect(10+2*s*i,5,s,s);
  }

  // fülle die zweite Reihe mit Quadraten
  for (int i=0; i<=7; i++){
    rect(20+2*s*i,5+s,s,s);
  }

  // fülle die dritte Reihe mit Quadraten
  for (int i=0; i<=7; i++){
    rect(30+2*s*i,5+2*s,s,s);
  }
}
```



Teil 3:

```
// globale Variablen
int s = 40; // Seitenlänge der Quadrate bzw. Abstand der Parallelen

int[] abstaende = {10, 20, 30, 20, 10, 20, 30, 20, 10}; // Abstände des 1. Quadrates nach links

void parallelen(){ // zeichnet 10 graue Parallelen
  stroke(125); // Linienfarbe Graustufe
  for (int i=0; i<=9; i++){
    line(0,5+i*s, 640, 5+i*s);
  }
}

void zeichneAlleQuadrate(){
  fill(0); // Füllfarbe schwarz
  for (int j=0; j<=8; j++){
    for (int i=0; i<=7; i++){
      // rect(10+2*s*i,5+j*s,s,s);  // Abstand bei allen Reihen gleich: d=10
      rect(abstaende[j]+2*s*i, 5+j*s, s, s);  // Abstand wird jeweils aus dem Feld gelesen
    }
  }
}

void setup(){
  size(640, 400);
  background(255); // Hintergrund weiß
  parallelen();  // Aufruf der Methode zum Zeichnen der Parallelen
  zeichneAlleQuadrate();
  //zeichneQuadrate(); // Aufruf der Methode zum Zeichnen der Quadrate
}
```