Weitere optische Täuschungen

Du hast schon die sog. Kaffeehaus-Täuschung kennen gelernt und selbst programmiert. Es gibt noch weitere optische Täuschungen, die das Gefühl vermitteln, dass parallele Geraden krumm verlaufen.

Beispiel 1:

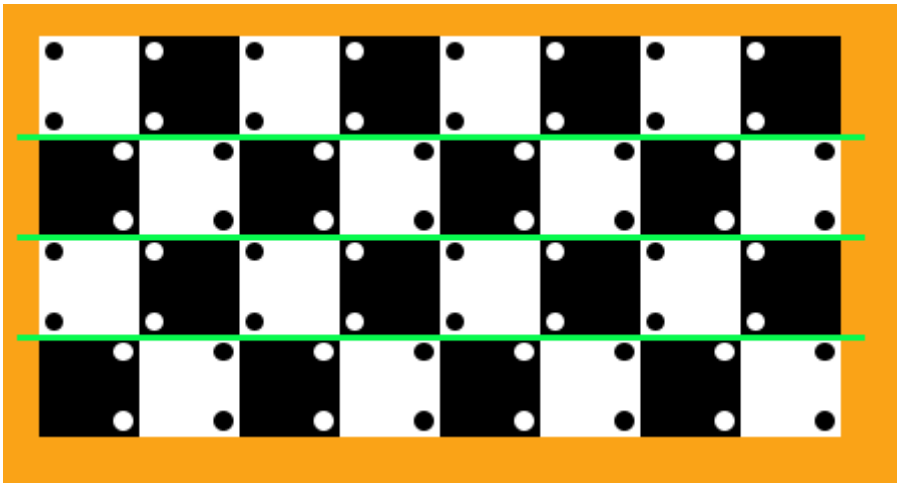
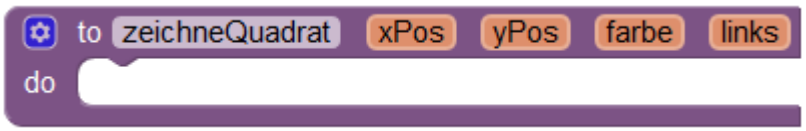


Bild: Screenshot von Ausführung des Programms „wellen1_mit_farbwechsel“ (Eisenmann)

Das Zeichnen des Bildes wäre mit dem, was du bisher in Java gelernt hast, einfach zu programmieren, wenn die kleinen Kreise in den Quadraten nicht wären. Auch diese könnten wir mit Schleifen extra zeichnen lassen, das wäre aber ziemlich umständlich.

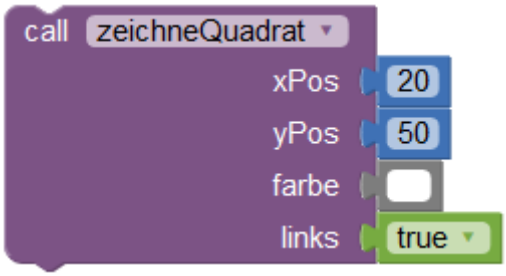
In Scratch oder dem MIT-App-Inventor hast du **Unterprogramme** kennen gelernt. Wir haben Codeteile, die wir häufiger gebraucht haben, zusammengefasst.

Zur Erinnerung:



Das Unterprogramm hier heißt „zeichneQuadrat“ und es können ihm vier Werte übergeben werden. Die Koordinaten der linken oberen Ecke, die Farbe und die Information, ob die Kreise links (true) oder rechts (false) liegen.

Der Aufruf des Unterprogramms könnte z.B. folgendermaßen aussehen:



Unterprogramme gibt es auch in Java. Man nennt sie auch **Methoden**. Wir haben schon einige fertige Methoden genutzt: `ellipse, rect, fill`, usw. Außerdem haben wir schon einfache Methoden selbst geschrieben.

Um eine Methode zu schreiben, müssen wir uns zunächst überlegen, ob einfach etwas ausgeführt werden soll, oder ob wir etwas ausführen lassen und einen Rückgabewert erwarten.

Allgemein sieht der Rahmen einer **Methode** folgendermaßen aus:

```
<Typ Rückgabewert> <Name der Methode> (<mögliche Übergabeparameter>)
{

 … }
```

Wird keine Rückgabe erwartet, steht vor dem Namen der Methode `void`.

Gibt es keine Übergabeparameter, so bleibt die Klammer leer.

1. *Lies dir das Blatt „Infos zu Methoden in Java“ durch, wenn du es noch nicht bei den Übungen zur Zählschleife gemacht hast.*

Bei der optischen Täuschung von oben möchten wir einfach etwas ausführen lassen (das Zeichnen eines Quadrates mit zwei Punkten), es gibt also keinen Rückgabewert. Um ein Quadrat zeichnen zu können, brauchen wir die Position (Koordinaten der linken oberen Ecke), die Hintergrundfarbe und die Information, ob die Kreise links oder rechts liegen.

Die benötigten Typen hast du schon kennen gelernt: `int` für eine ganze Zahl (auch Farben werden durch ganze Zahlen dargestellt) und `boolean` für die Wahrheitswerte `true` und `false`.

Der Rahmen unserer Methode sieht also folgendermaßen aus:

```
void zeichneQuadrat(int x, int y, int farbe, boolean links) {

}
```

(Die Variablen werden in der Entwicklungsumgebung übrigens unterkringelt, weil sie in der Methode noch nicht genutzt wurden.)

2. *Speichere eine neue Datei unter „wellen1“ ab.*
3. *Schreibe dann zunächst den Rahmen der setup()-Methode.*
4. *Schreibe den Rahmen der eigenen Methode „zeichneQuadrat“ (s.o.).*

Lösungen siehe im Codeteil unten.

Noch sind beide Methoden leer.

5. *Überlege dir mit deiner Sitznachbarin / deinem Sitznachbarn, was in der Methode „zeichneQuadrat“ passieren soll.*

Habt ihr es herausgefunden? Dann holt euch das nächste Blatt.

Es soll ein Quadrat an der vorgegebenen Position mit der vorgegebenen Farbe (weiß oder schwarz) gezeichnet werden.

In dem Quadrat sollen zwei kleine Kreise in der jeweils anderen Farbe entweder links oder rechts gezeichnet werden. Je nachdem, ob `true` (für links) oder `false` (für rechts) übergeben wird.

Um ein Quadrat zeichnen zu können, fehlt uns noch die Seitenlänge.

6. *Deklariert und initialisiert eine globale Variable s für die Seitenlänge.*

Euer Code sollte bisher etwa so aussehen:

```
// globale Variable
int s = 40; // Seitenlänge der Quadrate

void zeichneQuadrat(int x, int y, int farbe, boolean links) {

}

void setup() {

}
```

Bevor wir in der Methode zeichneQuadrat() ein Quadrat zeichnen lassen, müssen wir die Füllfarbe festlegen.

Dann wird die Methode für das Zeichnen eines Rechtecks mit den nötigen Werten aufgerufen.

```
void zeichneQuadrat(int x, int y, int farbe, boolean links) {
  fill(farbe); // der übergebene Wert der Variable farbe wird als Füllfarbe übergeben
  rect(x, y, x+s, y+s); // ein Quadrat mit Seitenlänge s wird an der Position (x|y) gezeichnet
}
```

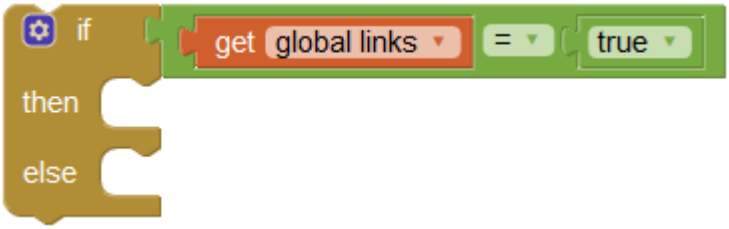
7. *Ergänze dein Programm entsprechend.*

Jetzt fehlen noch die beiden kleinen Kreise auf der linken oder rechten Seite. Auf welcher Seite sie gezeichnet werden sollen, legt der Übergabeparameter `links` fest. Beim Aufruf der Methode wird entweder `true` oder `false` übergeben. Welche Farbe sie haben, hängt von der Farbe des Quadrates ab.

8. *Ergänze den folgenden Satz für unsere Abbildung: „Falls die Farbe des Quadrates weiß ist, ist die Füllfarbe der Kreise …, sonst … .“*
   Lösung: „… ist die Füllfarbe der Kreise schwarz, sonst weiß.“
9. *Ergänze den folgenden Satz: „Falls die Variable `links` den Wert `true` besitzt, dann …, sonst … .“*
   Lösung: „… werden die beiden Kreise links gezeichnet, sonst rechts.“
10. *Welche Kontrollstruktur brauchst du hier jeweils? - Lösung: Eine Verzweigung.*

Wir brauchen eine **Verzweigung**.

Zur Erinnerung:



Falls die Bedingung erfüllt ist (hier: Variable `links` besitzt den Wahrheitswert `true`), wird ausgeführt, was hinter `then` steht, falls nicht, das was hinter `else` steht.

Eine **Verzweigung** in Java sieht folgendermaßen aus:

```
if (<Bedingung>) {
  …
} else {
  …
}
```

In unserem Fall:

```
if (farbe == 255) {
  // Quadrat ist weiß, Füllfarbe muss schwarz werden
} else {
  // Quadrat ist nicht weiß (schwarz), Füllfarbe muss weiß werden
}
```

oder:

```
if (links == true){
  // zeichne die beiden Kreise links
} else {
  // zeichne die beiden Kreise rechts
}
```

Achtung: Für eine Zuweisung genügt ein einfaches Gleichheitszeichen. Möchte man in der Bedingung aber auf Gleichheit überprüfen wie hier, so muss man zwei Gleichheitszeichen schreiben.

11. Schreibe zunächst in deine Methode zeichneQuadrat(…) die erste Verzweigung zur Farbauswahl der kleinen Kreise. Vergleiche, wenn nötig, mit der ausliegenden Lösung.

```
void zeichneQuadrat(int x, int y, int farbe, boolean links) {
  fill(farbe); // der übergebene Wert der Variable farbe wird als Füllfarbe übergeben
  rect(x, y, x+s, y+s); // ein Quadrat mit Seitenlänge s wird an der Position (x|y) gezeichnet

  // Farbauswahl Kreise
  if (farbe == 255) {  // Quadrat ist weiß
    fill(0); // Füllfarbe schwarz wird gewählt
  } else {  // Quadrat ist nicht weiß (schwarz)
    fill(255);  // Füllfarbe weiß wird gewählt
  }

}
```

12. Für die zweite Verzweigung, die sich um das Zeichnen der Kreise kümmern soll, ist eine Skizze (auf Papier) sehr hilfreich. Fertige eine solche Skizze eines Quadrates mit zwei kleinen Kreisen innerhalb an. Überlege dir dabei, welche Werte die Methode ellipse() jeweils übergeben bekommen muss. Nutze den Modus ellipseMode(CORNERS)[1].

Der andere Modus ist hier einfacher, weil der Mittelpunkt schwerer zu beschreiben ist.

Wenn man möchte, kann man auch eine weitere Variable d für s : 4 einführen. Dann würden z.B. die Koordinaten beim rechten unteren Kreis (x + 3d | y + 3d) heißen.

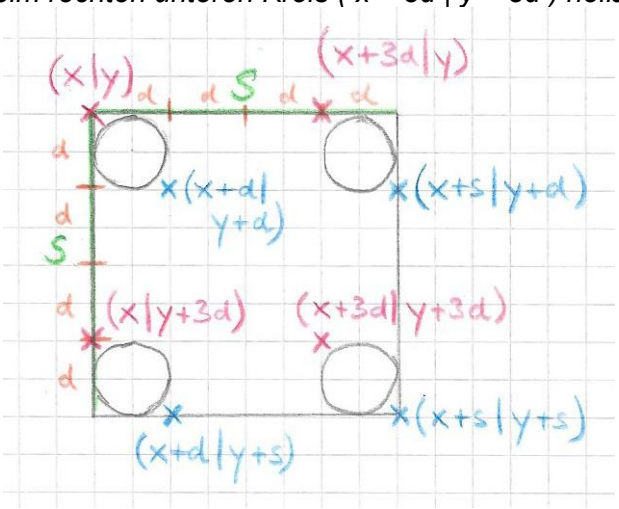


Bild: eigenes (Eisenmann)

(Weitere Aufträge findest du auf dem nächsten Blatt.)

1 ellipse(x1,y1,x2,y2) – (x1|y1) linke obere u. (x2|y2) rechte untere Ecke des umschließenden Rechtecks

13. *Schreibe jetzt mithilfe deiner Skizze die Verzweigung, in der die beiden kleinen Kreise gezeichnet werden. (In der vorliegenden Lösung wurde als Kreisdurchmesser ein Viertel der Seitenlänge des Quadrates gewählt und der Einfachheit halber wurden die Kreise zunächst direkt an den Quadratrand gesetzt.)*

```
// Zeichnen der beiden kleinen Kreise
ellipseMode(CORNERS);
noStroke();
int d = s/4;
if (links == true) {
  ellipse(x, y, x+d, y+d);  // links oben
  ellipse(x, y + 3*d, x+d, y+s); // links unten
} else {
  ellipse(x + 3*d, y, x+s, y+d);  // rechts oben
  ellipse(x + 3*d, y + 3*d, x+ s, y+s);  // rechts unten
}
```

14. *Teste dein Programm, indem du in der setup()-Methode zunächst die Größe des Fensters (z.B. 600 auf 400 Pixel) einstellst und dann die Methode „zeichneQuadrat()“ mit verschiedenen Werten aufrufst.*

```
void setup() {
  size(600,400);
  zeichneQuadrat(100,100,0,true);
  zeichneQuadrat(100, 200, 255, false);

}
```

15. *Vergleiche deinen Programmcode mit der ausliegenden Lösung.*

```
// globale Variable
int s = 40; // Seitenlänge der Quadrate

void zeichneQuadrat(int x, int y, int farbe, boolean links) {
  fill(farbe); // der übergebene Wert der Variable farbe wird als Füllfarbe übergeben
  rect(x, y, s, s); // ein Quadrat mit Seitenlänge s wird an der Position (x|y) gezeichnet

  // Farbauswahl Kreise
  if (farbe == 255) {  // Quadrat ist weiß
    fill(0); // Füllfarbe schwarz wird gewählt
  } else {  // Quadrat ist nicht weiß (schwarz)
    fill(255);  // Füllfarbe weiß wird gewählt
  }

  // Zeichnen der beiden kleinen Kreise
  ellipseMode(CORNERS);
  noStroke();
  int d = s/4;
  if (links == true) {
    ellipse(x, y, x+d, y+d);  // links oben
    ellipse(x, y + 3*d, x+d, y+s); // links unten
  } else {
    ellipse(x + 3*d, y, x+s, y+d);  // rechts oben
    ellipse(x + 3*d, y + 3*d, x+ s, y+s);  // rechts unten
  }
}

void setup() {
  size(600, 400);
  zeichneQuadrat(100, 100, 0, true);
  zeichneQuadrat(100, 200, 255, false);
}
```

16. *Schreibe jetzt den Rahmen der Methode alleQuadrate(), in der dann die optische Täuschung gezeichnet werden soll.*

```
void alleQuadrate() {
}
```

17. *Nutze eine Zählschleife, um die erste Reihe von Quadraten zu zeichnen. Achte darauf, dass von Quadrat zu Quadrat die Füllfarbe wechselt, die Position der Kreise aber gleich bleibt. Nutze dazu eine Variable `aktuellefarbe` vom Typ `int`.*

```
void alleQuadrate() {
  noStroke();
  // erste Reihe
  int aktuellefarbe = 255; // Farbe des ersten Quadrates ist weiß
  for (int i=0; i<=7; i++) {
    zeichneQuadrat(i*s, 0, aktuellefarbe, true);
    // Farbwechsel für nächstes Quadrat
    if (aktuellefarbe == 255) {
      aktuellefarbe = 0;
    } else {
      aktuellefarbe = 255;
    }
  }
}
```

18. Programmiere auch das Zeichnen der zweiten, dritten und vierten Reihe.

```
// zweite Reihe
aktuellefarbe = 0; // Farbe des ersten Quadrates ist schwarz
for (int i=0; i<=7; i++) {
  zeichneQuadrat(i*s, s, aktuellefarbe, false);
  // Farbwechsel für nächstes Quadrat
  if (aktuellefarbe == 255) {
    aktuellefarbe = 0;
  } else {
    aktuellefarbe = 255;
  }
}
  // dritte Reihe
aktuellefarbe = 255; // Farbe des ersten Quadrates ist weiß
for (int i=0; i<=7; i++) {
  zeichneQuadrat(i*s, 2*s, aktuellefarbe, true);
  // Farbwechsel für nächstes Quadrat
  if (aktuellefarbe == 255) {
    aktuellefarbe = 0;
  } else {
    aktuellefarbe = 255;
  }
}
// vierte Reihe
aktuellefarbe = 0; // Farbe des ersten Quadrates ist schwarz
for (int i=0; i<=7; i++) {
  zeichneQuadrat(i*s, 3*s, aktuellefarbe, false);
  // Farbwechsel für nächstes Quadrat
  if (aktuellefarbe == 255) {
    aktuellefarbe = 0;
  } else {
    aktuellefarbe = 255;
  }
}
```

19. Vergleiche mit der ausliegenden Lösung.

Lösung siehe unten.

*20. ***** Hast du eine Idee, wie wir* alle *Reihen mithilfe einer verschachtelten Zählschleife zeichnen lassen können? Besprich dich mit deiner Sitznachbarin / deinem Sitznachbarn und probiert eure Ideen aus.*

Mögliche Lösung:

In jedem Schritt wird die Farbe gewechselt, im Aufruf der Methode zeichneQuadrat() verändert sich die y-Koordinate und es wechselt zwischen links = true und links = false.

```
void alleQuadrate() {
  noStroke();
  int aktuellefarbe = 255; // Farbe des ersten Quadrates ist weiß
  boolean pos_aktuell = true; // Kreise sind zunächst links
  for (int j=0; j<=3; j++) {
    for (int i=0; i<=7; i++) {
      zeichneQuadrat(i*s, j*s, aktuellefarbe, pos_aktuell);
      // Farbwechsel für nächstes Quadrat
      if (aktuellefarbe == 255) {
        aktuellefarbe = 0;
      } else {
        aktuellefarbe = 255;
      }
    } // end Zeichnen einer Reihe
    // Wechsel Startfarbe
    if (aktuellefarbe == 255) {
      aktuellefarbe = 0;
    } else {
      aktuellefarbe = 255;
    }
    // Wechsel Position der Kreise
    if (pos_aktuell == true) {
      pos_aktuell = false;
    } else {
      pos_aktuell = true;
    }
  }
}
```

Weitere Aufgaben:

21. Programmiert das Zeichnen der grünen Trennlinien zwischen den Reihen.

```
//zeichne die drei Linien zwischen die Quadrate
for (int i=0; i<=2; i++) {
  stroke(8,250,78);  // Linienfarbe hellgrün
  strokeWeight(2);
  line(0, s + s*i, 8*s, s + s*i);
}
```

22. Führt eine Variable a für den Abstand der Kreise zum Quadratrand ein und sucht einen Wert, für den die optische Täuschung am wirkungsvollsten ist.

```
// globale Variable
int s = 40; // Seitenlänge der Quadrate
int a = 2; // Abstand Kreise - Quadratrand


// Zeichnen der beiden kleinen Kreise
ellipseMode(CORNERS);
noStroke();
int d = s/4;
if (links == true) {
  ellipse(x+a, y+a, x+d+a, y+d+a);  // links oben
  ellipse(x+a, y + 3*d-a, x+d+a, y+s-a); // links unten
} else {
  ellipse(x + 3*d-a, y+a, x+s-a, y+d+a);  // rechts oben
  ellipse(x + 3*d-a, y + 3*d-a, x+ s-a, y+s-a);  // rechts unten
}
```

23. Verändert euer Programm so, dass die optische Täuschung Abstand zu den Rändern hat.

```
// globale Variable
int s = 40; // Seitenlänge der Quadrate
int a = 2; // Abstand Kreise - Quadratrand
int d = 20; // Abstand der optischen Täuschung zum Rand


for (int j=0; j<=3; j++) {
  for (int i=0; i<=7; i++) {
    zeichneQuadrat(d + i*s, d + j*s, aktuellefarbe, pos_aktuell);


//zeichne die drei Linien zwischen die Quadrate
for (int i=0; i<=2; i++) {
  stroke(8,250,78);  // Linienfarbe hellgrün
  strokeWeight(2);
  line(d, d + s + s*i, d + 8*s, d + s + s*i);
}
```

24. Wir haben mehrere Male den Wechsel der Farbe von schwarz auf weiß und umgekehrt benutzt. Schreibe eine Methode „farbwechsel“, der man die Farbe schwarz (0) oder weiß (255) übergeben kann und die die jeweils andere Farbe zurückgibt.

```
int farbwechsel (int farbe) {
  if (farbe == 255) { // Farbe ist weiß
    farbe = 0;
  } else {  // Farbe ist nicht weiß
    farbe = 255;
  }
  return farbe;
}
```

25. Nutze die Methode überall dort in deinem Programm, wo ein Farbwechsel stattfindet.

```
void zeichneQuadrat(int x, int y, int farbe, boolean links) {
  fill(farbe); // der übergebene Wert der Variable farbe wird
  rect(x, y, s, s); // ein Quadrat mit Seitenlänge s wird an d

  // Farbauswahl Kreise
  fill(farbwechsel(farbe));


void alleQuadrate() {
  noStroke();
  int aktuellefarbe = 255; // Farbe des ersten Quadrates ist weiß
  boolean pos_aktuell = true; // Kreise sind zunächst links
  for (int j=0; j<=3; j++) {
    for (int i=0; i<=7; i++) {
      zeichneQuadrat(d + i*s, d + j*s, aktuellefarbe, pos_aktuell);
      // Farbwechsel für nächstes Quadrat
      aktuellefarbe = farbwechsel(aktuellefarbe);
    } // end Zeichnen einer Reihe
    // Wechsel Startfarbe
    aktuellefarbe = farbwechsel(aktuellefarbe);
    // Wechsel Position der Kreise
    if (pos_aktuell == true) {
      pos_aktuell = false;
    } else {
      pos_aktuell = true;
    }
  }
```

26. Teste dein Programm und vergleiche dann mit der ausliegenden Lösung.

Lösung nach Aufgabe 11:

```
void zeichneQuadrat(int x, int y, int farbe, boolean links) {
  fill(farbe); // der übergebene Wert der Variable farbe wird als Füllfarbe übergeben
  rect(x, y, x+s, y+s); // ein Quadrat mit Seitenlänge s wird an der Position (x|y) gezeichnet

  // Farbauswahl Kreise
  if (farbe == 255) {  // Quadrat ist weiß
    fill(0); // Füllfarbe schwarz wird gewählt
  } else {  // Quadrat ist nicht weiß (schwarz)
    fill(255);  // Füllfarbe weiß wird gewählt
  }

}
```

Lösung nach Aufgabe 14:

```
// globale Variable
int s = 40; // Seitenlänge der Quadrate

void zeichneQuadrat(int x, int y, int farbe, boolean links) {
  fill(farbe); // der übergebene Wert der Variable farbe wird als Füllfarbe übergeben
  rect(x, y, s, s); // ein Quadrat mit Seitenlänge s wird an der Position (x|y) gezeichnet

  // Farbauswahl Kreise
  if (farbe == 255) {  // Quadrat ist weiß
    fill(0); // Füllfarbe schwarz wird gewählt
  } else {  // Quadrat ist nicht weiß (schwarz)
    fill(255);  // Füllfarbe weiß wird gewählt
  }

  // Zeichnen der beiden kleinen Kreise
  ellipseMode(CORNERS);
  noStroke();
  int d = s/4;
  if (links == true) {
    ellipse(x, y, x+d, y+d);  // links oben
    ellipse(x, y + 3*d, x+d, y+s); // links unten
  } else {
    ellipse(x + 3*d, y, x+s, y+d);  // rechts oben
    ellipse(x + 3*d, y + 3*d, x+ s, y+s);  // rechts unten
  }
}

void setup() {
  size(600, 400);
  zeichneQuadrat(100, 100, 0, true);
  zeichneQuadrat(100, 200, 255, false);
}
```

Lösung nach Aufgabe 19:

```
void alleQuadrate() {
  noStroke();
  // erste Reihe
  int aktuellefarbe = 255; // Farbe des ersten Quadrates ist weiß
  for (int i=0; i<=7; i++) {
    zeichneQuadrat(i*s, 0, aktuellefarbe, true);
    // Farbwechsel für nächstes Quadrat
    if (aktuellefarbe == 255) {
      aktuellefarbe = 0;
    } else {
      aktuellefarbe = 255;
    }
  }
  // zweite Reihe
  aktuellefarbe = 0; // Farbe des ersten Quadrates ist schwarz
  for (int i=0; i<=7; i++) {
    zeichneQuadrat(i*s, s, aktuellefarbe, false);
    // Farbwechsel für nächstes Quadrat
    if (aktuellefarbe == 255) {
      aktuellefarbe = 0;
    } else {
      aktuellefarbe = 255;
    }
  }
    // dritte Reihe
  aktuellefarbe = 255; // Farbe des ersten Quadrates ist weiß
  for (int i=0; i<=7; i++) {
    zeichneQuadrat(i*s, 2*s, aktuellefarbe, true);
    // Farbwechsel für nächstes Quadrat
    if (aktuellefarbe == 255) {
      aktuellefarbe = 0;
    } else {
      aktuellefarbe = 255;
    }
  }
  // vierte Reihe
  aktuellefarbe = 0; // Farbe des ersten Quadrates ist schwarz
  for (int i=0; i<=7; i++) {
    zeichneQuadrat(i*s, 3*s, aktuellefarbe, false);
    // Farbwechsel für nächstes Quadrat
    if (aktuellefarbe == 255) {
      aktuellefarbe = 0;
    } else {
      aktuellefarbe = 255;
    }
  }
}
```

Lösung nach Aufgabe 26:

```
// globale Variable
int s = 40; // Seitenlänge der Quadrate
int a = 2; // Abstand Kreise - Quadratrand
int d = 20; // Abstand der optischen Täuschung zum Rand

int farbwechsel (int farbe) {
  if (farbe == 255) { // Farbe ist weiß
    farbe = 0;
  } else {  // Farbe ist nicht weiß
    farbe = 255;
  }
  return farbe;
}


void zeichneQuadrat(int x, int y, int farbe, boolean links) {
  fill(farbe); // der übergebene Wert der Variable farbe wird als Füllfarbe übergeben
  rect(x, y, s, s); // ein Quadrat mit Seitenlänge s wird an der Position (x|y) gezeichnet

  // Farbauswahl Kreise
  fill(farbwechsel(farbe));

  // Zeichnen der beiden kleinen Kreise
  ellipseMode(CORNERS);
  noStroke();
  int d = s/4;
  if (links == true) {
    ellipse(x+a, y+a, x+d+a, y+d+a);  // links oben
    ellipse(x+a, y + 3*d-a, x+d+a, y+s-a); // links unten
  } else {
    ellipse(x + 3*d-a, y+a, x+s-a, y+d+a);  // rechts oben
    ellipse(x + 3*d-a, y + 3*d-a, x+ s-a, y+s-a);  // rechts unten
  }
}
```

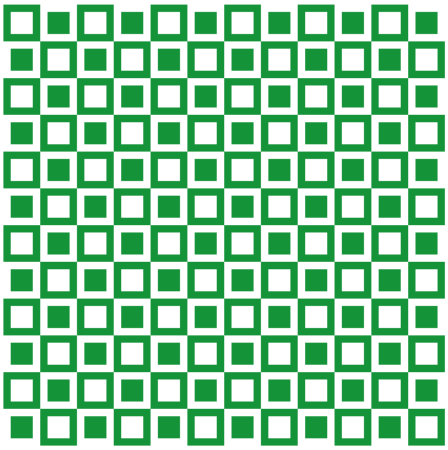
```
void alleQuadrate() {
  noStroke();
  int aktuellefarbe = 255; // Farbe des ersten Quadrates ist weiß
  boolean pos_aktuell = true; // Kreise sind zunächst links
  for (int j=0; j<=3; j++) {
    for (int i=0; i<=7; i++) {
      zeichneQuadrat(d + i*s, d + j*s, aktuellefarbe, pos_aktuell);
      // Farbwechsel für nächstes Quadrat
      aktuellefarbe = farbwechsel(aktuellefarbe);
    } // end Zeichnen einer Reihe
    // Wechsel Startfarbe
    aktuellefarbe = farbwechsel(aktuellefarbe);
    // Wechsel Position der Kreise
    if (pos_aktuell == true) {
      pos_aktuell = false;
    } else {
      pos_aktuell = true;
    }
  }

  //zeichne die drei Linien zwischen die Quadrate
  for (int i=0; i<=2; i++) {
    stroke(8, 250, 78);  // Linienfarbe hellgrün
    strokeWeight(2);
    line(d, d + s + s*i, d + 8*s, d + s + s*i);
  }
}

void setup() {
  size(600, 400);
  // Zeichnen der optischen Täuschung
  alleQuadrate();
}
```

Weitere Aufgaben

Beispiel 1:



Bilder: Screenshots von Ausführung des Programms „wellen_b1_lsg_farbwechsel“ (Eisenmann)

1. *Programmiere das Zeichnen der optischen Täuschung aus Beispiel 1. Verwende dazu alles, was du gelernt hast: eigene Methoden (zeichneQuadrat(…) und farbwechsel(...)), verschachtelte Zählschleife. (Hilfekarten liegen aus.)*
2. *In einer zweiten Version sollen die Quadrate das ganze Fenster füllen. Dazu brauchst du eine neue Schleifenart. Hol dir dazu die Anleitung bei deiner Lehrerin.*
3. *Suche dir eine weitere optische Täuschung aus, die du schon programmiert hast, speichere sie unter einem neuen Namen ab und verändere sie mithilfe der neuen Schleifenart so, dass sie das ganze Fenster füllt.*
4. ***** Eine Herausforderung ist Beispiel 2, wenn du noch Zeit hast. Ein paar Hinweise dazu liegen aus.*

Beispiel 2 *:**

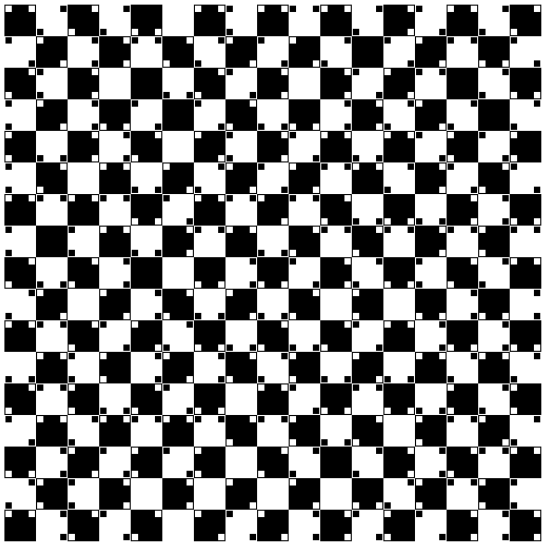


Bild: Screenshot von Ausführung des Programms „wellen2_zufall“ (Eisenmann)

HILFEKARTEN ZU BEISPIEL 1

Tipp 1

Überlege dir wieder eine Methode `zeichneQuadrat()`, mit der du ein Quadrat mit grünem Rand und weißer Füllung oder weißem Rand und grüner Füllung zeichnen lassen kannst.

Nutze zwei globale Variablen für die Farben `weiss` und `gruen`. Wähle einen Grünton aus der Farbauswahl unter Tools.

Es ist später einfacher, die Methode zu nutzen, wenn man zwei Quadrate übereinander zeichnen lässt. Z.B. ein grünes Quadrat mit Seitenlänge s und darauf ein weißes Quadrat, das überall vom Rand den Abstand d hat.

Tipp 2

Hast du es geschafft? Dann vergleiche mit der Lösung auf Tippkarte 5.

Du hast noch keine Idee? Dann überlege dir zunächst, was du der Methode mitteilen willst, also welche Übergabeparameter du brauchst.

Tipp 3

Hier genügen im Gegensatz zu der vorherigen optischen Täuschung die Position der linken oberen Ecke und die Grundfarbe. Hier kannst du entscheiden, ob du die Farbe des hinteren oder die des darüber liegenden Quadrates übergeben willst.

Tipp 4

Der Rahmen deiner Methode könnte jetzt z.B. so aussehen:

```
void zeichneQuadrat(int x, int y, int rahmenfarbe) { … }
```

Schreibe jetzt die Methode so, dass das Quadrat an der Position mit der Farbe gezeichnet wird.

Tipp 5

```
void zeichneQuadrat(int x, int y, int rahmenfarbe) {

  fill(rahmenfarbe); // Linienfarbe einstellen
  noStroke();
  rect(x, y, s, s); // Zeichnen des hinteren Quadrates

  // Bestimmen der Farbe des inneren Quadrates
  if (rahmenfarbe == weiss) {
    fill(gruen);
  } else {
    fill(weiss);
  }

  rect(x+d, y+d, s-2*d, s-2*d); // Zeichnen des inneren Quadrates
}
```

Teste die Methode, indem du sie zweimal aufrufst. Nutze zwei unterschiedliche Positionen und übergib einmal die Farbe weiss, einmal gruen.

Tipp 6

Dein Test könnte etwa so aussehen:

```
// globale Variable
int s=50; // Seitenlänge der Quadrate
int d=5; // Abstand inneres Quadrat zum Rand
int weiss = 255;
int gruen = #1CAF38;


// setup-Methode
void setup() {
  size(700, 400);
  background(255); // weißer Hintergrund

  // Test
  zeichneQuadrat(30, 30, gruen);
  zeichneQuadrat(30+s, 30, weiss);
}
```

Tipp 7

Überlege dir jetzt, wie du die erste Reihe mit einer Zählschleife programmiere kannst und in einem zweiten Schritt mit einer weiteren Zählschleife alle Reihen.

Tipp 8

Hast du es allein geschafft? Dann vergleiche mit der Lösung.

Du brauchst Hilfe? Dann rufe zunächst deine Methode für die ersten drei Quadrate auf und überlege dir, was sich verändert.

Schreibe jetzt die Zählschleife für die erste Reihe.

Tipp 9

Es ändert sich wieder nur die x-Koordinate der Startposition der Quadrate. Es kommen jedes Mal s Pixel dazu.

Der Code der Zählschleife könnte so aussehen:

```
void alleQuadrate(){
  int farbe = gruen; // Festlegen der ersten Rahmenfarbe
  // erste Reihe
  for (int i=0; i<12; i++) {
    zeichneQuadrat(30+i*s, 30, farbe);
    if (farbe == gruen) {  // du könntest hier wieder eine Methode Farbwechsel schreiben
      farbe = weiss;
    } else {
      farbe = gruen;
    }
  }
}
```

Es ist hier ein Abstand zum Fensterrand mit 30 Pixel angegeben. Das Bild dazu sieht so aus:

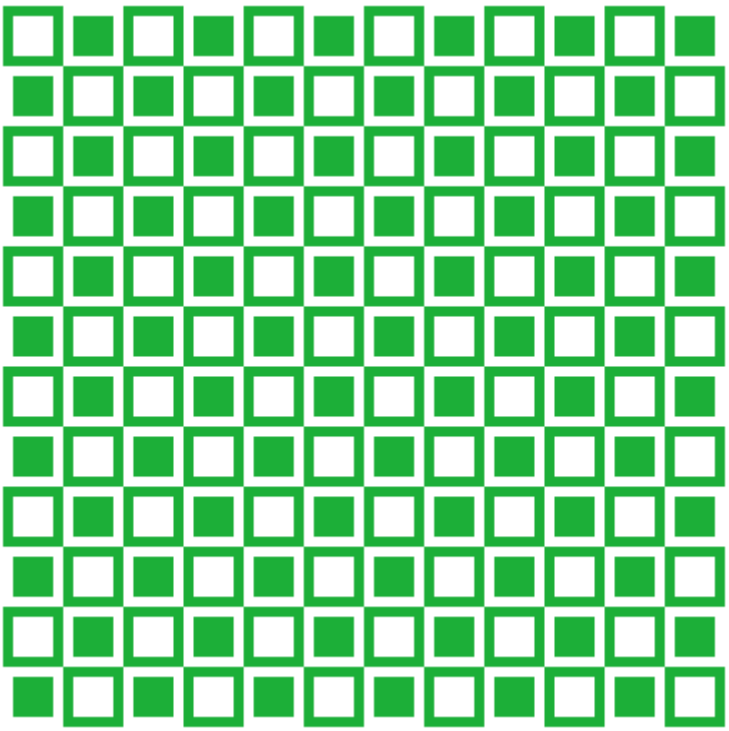
Wenn du noch keine Idee hast, schreibe den Code für die zweite Reihe,… Überlege dir dann, was sich von Reihe zu Reihe verändert und schreibe die zweite Zählschleife für die Reihen.

Vergleiche dann mit der Lösung.

Lösung

```
void alleQuadrate(){
  int farbe = gruen; // Festlegen der ersten Rahmenfarbe
  for (int j=0; j<12; j++) { // j zählt die Reihen

    for (int i=0; i<12; i++) {  // i zählt die Quadrate in einer Reihe
      zeichneQuadrat(30+i*s, 30+j*s, farbe);
      if (farbe == gruen) {  // du könntest hier wieder eine Methode Farbwechsel schreiben
        farbe = weiss;
      } else {
        farbe = gruen;
      }
    }
    // Wechsel der Farbe des ersten Quadrates für die nächste Reihe
    if (farbe == gruen) {  // du könntest hier wieder eine Methode Farbwechsel schreiben
      farbe = weiss;
    } else {
      farbe = gruen;
    }
  }
}
```



Schreibe jetzt noch eine Methode Farbwechsel und nutze sie geeignet, wenn du es noch nicht gemacht hast.

Lösung mit Farbwechsel

```
// globale Variable
int s=30; // Seitenlänge der Quadrate
int d=5; // Abstand inneres Quadrat zum Rand
int weiss = 255;
int gruen = #1CAF38;


// setup-Methode
void setup() {
  size(700, 700);
  background(255); // weißer Hintergrund
  alleQuadrate();
}

int farbwechsel(int farbe) {
  if (farbe == weiss) {  // Wechsel von weiß auf grün
    return gruen;
  } else {  // Wechsel von grün auf weiß
    return weiss;
  }
}

void zeichneQuadrat(int x, int y, int rahmenfarbe) {

  fill(rahmenfarbe); // Linienfarbe einstellen
  noStroke();
  rect(x, y, s, s); // Zeichnen des hinteren Quadrates

  // Bestimmen der Farbe des inneren Quadrates
  fill(farbwechsel(rahmenfarbe));

  rect(x+d, y+d, s-2*d, s-2*d); // Zeichnen des inneren Quadrates
}

void alleQuadrate(){
  int farbe = gruen; // Festlegen der ersten Rahmenfarbe
  for (int j=0; j<12; j++) { // j zählt die Reihen

    for (int i=0; i<12; i++) {  // i zählt die Quadrate in einer Reihe
      zeichneQuadrat(30+i*s, 30+j*s, farbe);
      farbe = farbwechsel(farbe);
    }
    // Wechsel der Farbe des ersten Quadrates für die nächste Reihe
    farbe = farbwechsel(farbe);
  }
}
```

Lösung mit while-Schleife:

(Rest siehe Lösung mit Farbwechsel)

```
void alleQuadrate(){
  int farbe = gruen; // Festlegen der ersten Rahmenfarbe
  int pos_x = 0;
  int pos_y = 0;
  while (pos_y < 700) { // unterer Rand ist noch nicht erreicht

    while (pos_x < 700) {  // rechter Rand ist noch nicht erreicht
      zeichneQuadrat(pos_x, pos_y, farbe);
      farbe = farbwechsel(farbe);
      pos_x = pos_x + s;  // verändern der x-Position um s
    }
    // Wechsel der Farbe des ersten Quadrates für die nächste Reihe
    farbe = farbwechsel(farbe);
    // x-Position wieder an linken Rand setzen
    pos_x = 0;
    pos_y = pos_y + s; // verändern der y-Position um s
  }
}
```

HINWEISE UND TIPPS ZU BEISPIEL 2

Information

Diese optische Täuschung gibt es in vielen Varianten. Es gibt solche, bei denen bestimmte Quadrate eine feste Position haben und solche, bei denen die verschiedenen Quadrate durch Zufall bestimmt werden.

Beispiel mit fester Position:

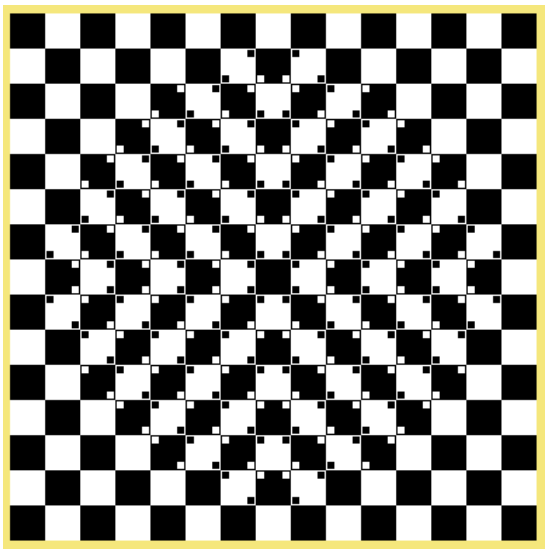


Bild: Screenshot von Ausführung des Programms „wellen2_fest“ (Eisenmann)

Du kannst selbst wählen, welche Art du programmieren möchtest.

Ganzzahlige **Zufallszahlen** zwischen a und b werden durch die Methode `random(a,b)` erzeugt. Allerdings gibt die Methode nicht den Typ `int`, sondern den Typ `float` zurück. Der steht grob für Kommazahlen.

Um einen Wert vom Typ int zu erhalten, kann man die Methode `round()` nutzen, die den übergebenen Wert auf eine ganze Zahl rundet.

Für festgelegte Positionen könntest du wieder ein Feld (`array`) nutzen, in dem du die Art der Quadrate vorab festlegst.

Wenn du keine Idee hast, wie man die **Position der kleinen Quadrate** übergeben kann, hier zwei Möglichkeiten:

1. Nutze vier Variablen vom Typ `boolean`. Ist das kleine Quadrat sichtbar, besitzt es den Wert `true`, wenn nicht, den Wert `false`.

2. Du könntest die Position der kleinen Quadrate durch Binärzahlen darstellen. Reihenfolge z.B. im Uhrzeigersinn von linker oberer Ecke bis zu linker unterer Ecke. Ist ein Quadrat sichtbar, hat es den Wert 1, sonst den Wert 0.

Kein Quadrat wäre die 0000b, alle vier Quadrate wäre die 1111b, z.B. Quadrat links oben und rechts unten wäre 1010b, usw.

Eine neue Schleifenart

Bei unseren bisherigen Beispielen wussten wir vorab immer, wie oft etwas wiederholt werden sollte. Wir konnten deshalb Zählschleifen nutzen.

Zur Erinnerung:

```
for (int i=0; i<10; i++) {
  // i läuft von 0 bis 9, es wird also 10-mal wiederholt
}
```

Wir könnten für zehn Wiederholungen auch i von 1 bis 10, von 2003 bis 2012, etc. laufen lassen. Welchen Bereich wir wählen, entscheiden wir je nachdem, wie wir die Zählvariable i in der Schleife nutzen wollen. Beispiele dazu kennst du ja schon einige.

Wissen wir vorab nicht, wie viele Schleifendurchläufe nötig sind, brauchen wir eine **Schleife mit Bedingung**.

Beispiel: „Solange die x-Koordinate kleiner ist als die Breite des Fensters, wiederhole das Zeichnen des Quadrates und erhöhe den Wert der x-Koordinate um die Seitenlänge des Quadrates."

Achtung:

In der Zählschleife wurde die Zählvariable automatisch erhöht. Hier in der neuen Schleifenart müssen wir uns selbst um die Veränderung der Variable aus der Bedingung kümmern.

Beispiel:

```
while (x < width) {
  zeichneQuadrat(x,y,farbe);
  x=x+s; // der Wert der x-Koordinate wird um den Wert von s erhöht
}
```

In Worten haben wir diese Schleife oben schon beschrieben.

Würden wir die Erhöhung von x weglassen, wäre die Bedingung immer erfüllt und die Schleife würde zu einer sogenannten **Endlosschleife**. Das Programm stürzt dabei ab.

Auch while-Schleifen können ineinander verschachtelt werden.

```
while (y < height) {
  while (x < width) {
    zeichneQuadrat(x, y, farbe);
    x=x+s;
  }
  y=y+s;
}
```

„Solange die y-Koordinate kleiner ist als die Höhe des Fensters, **zeichne eine Reihe Quadrate bis zum Rand** und erhöhe dann y um die Seitenlänge des Quadrates."

Der fettgedruckte Teil wird durch die innere while-Schleife ausgeführt und entspricht dem Text von oben.