Weitere Aufgaben

Aufgabe 1:

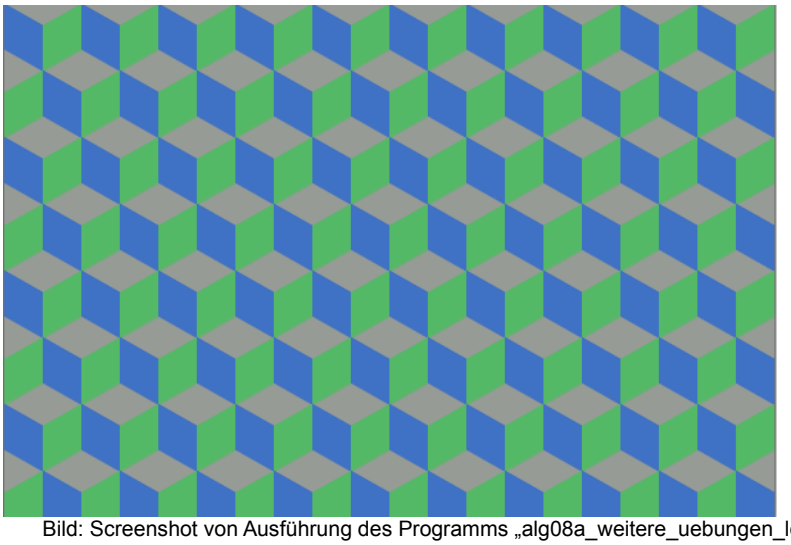


Bild: Screenshot von Ausführung des Programms „alg08a_weitere_uebungen_loesungen“ (Eisenmann)

1. *Überlege dir zunächst, aus welchen Figuren diese optische Täuschung besteht.*
2. *Schreibe eine Methode* `viereck()`*, der die Füllfarbe und die Koordinaten der vier Eckpunkte übergeben werden können. Nutze dazu die Methode* `quad()`[1]*:*
3. *Das Zeichnen der Rauten einer Farbe kann man sich sparen, wenn man zu Beginn den Hintergrund in dieser Farbe füllt. Überlege dir, welche Rauten dazu am geeignetsten sind und schreibe die* `setup()`*-Methode mit Einstellen der Fenstergröße und Färben des Hintergrunds (in der Vorlage: grau).*
4. *Programmiere dann das Zeichnen der Rauten der beiden anderen Farben (in der Vorlage: blau und grün im Wechsel).*
5. *Verändere ggf. so, dass das ganze Fenster gefüllt wird.*

1 Siehe Blatt „Methoden zum Zeichnen“

Mögliche Lösung zu Aufgabe 1:

1. Es handelt sich um Rauten.

2. Methode viereck():

```
void viereck(int farbe, int x1, int y1, int x2, int y2, int x3, int y3, int x4, int y4){
  fill(farbe);
  stroke(farbe);
  quad(x1,y1,x2,y2,x3,y3,x4,y4);
}
```

3. Die drei Farben wurden als globale Variablen deklariert und initialisiert:

```
int farbe1 = #3F71C4;  // blau
int farbe2 = #54B966;  // grün
int farbe3 = #969B96; // grau
```

In der setup()-Methode wird dann die Größe des Fensters und die Hintergrundfarbe eingestellt:

```
void setup() {
  size(600, 400);
  background(farbe3);
}
```

4. Es sind immer eine blaue und eine grüne Raute aneinandergelegt. Im Wechsel beginnt eine Reihe entweder mit blau oder mit grün. In der Lösung unten wurde eine Variable `ungerade` vom Typ `boolean` genutzt, die von Reihe zu Reihe ihren Wert zwischen `true` und `false` wechselt. In Reihe 1, 3, 5,… gilt `ungerade=true`, in Reihe 2, 4, 6, … gilt `ungerade=false`.

Weiter wurden zwei Variable x und y für die Position deklariert und jeweils auf den Wert 0 initialisiert.

Solange y noch nicht größer als die Höhe und x noch nicht größer als die Breite des Fensters ist, wird gezeichnet.

Nach jedem Paar von zwei Rauten wird x verändert bis die Reihe fertig ist.

Dann wird x entweder auf 0 oder auf -e/2 zurückgesetzt. Je nachdem, ob es sich um eine ungerade oder gerade Reihe handelt. Außerdem wird y erhöht.

Aufgabe 5. ist schon in der Lösung durch die verschachtelte while-Schleife integriert.

Gesamte Lösung siehe nächste Seite.

Mögliche Lösung zu Aufgabe 1:

```
// globale Variablen
int e = 60; // Länge der waagrechten Diagonale der grauen Raute
int f = 35; // Länge der senkrechten Diagonale der grauen Raute

int farbe1 = #3F71C4;  // blau
int farbe2 = #54B966;  // grün
int farbe3 = #969B96; // grau

// setup-Methode
void setup() {
  size(600, 400);
  background(farbe3);
  drei_d();
}

void viereck(int farbe, int x1, int y1, int x2, int y2, int x3, int y3, int x4, int y4){
  fill(farbe);
  stroke(farbe);
  quad(x1,y1,x2,y2,x3,y3,x4,y4);
}

void drei_d() {
  int x = 0;
  int y = 0;
  boolean ungerade = true; // es beginnt mit Farbe 1 - jede ungerade Reihe
  while (y < height) {
    while (x < width) {
      viereck(farbe1, x,y,x+ e/2,y+f/2,x+e/2,y+3*f/2,x,y+f);
      viereck(farbe2, x+ e/2,y+f/2,x+e/2,y+3*f/2,x+e,y+f,x+e,y);
      x = x + e;
    }
    if (ungerade == true){
      ungerade = false;
      x = -e/2;  // erste sichtbare Farbe links ist Farbe 2
    } else {
      ungerade = true;
      x = 0;
    }
    y = y + 3*f/2;
  }
}
```

Aufgabe 2:

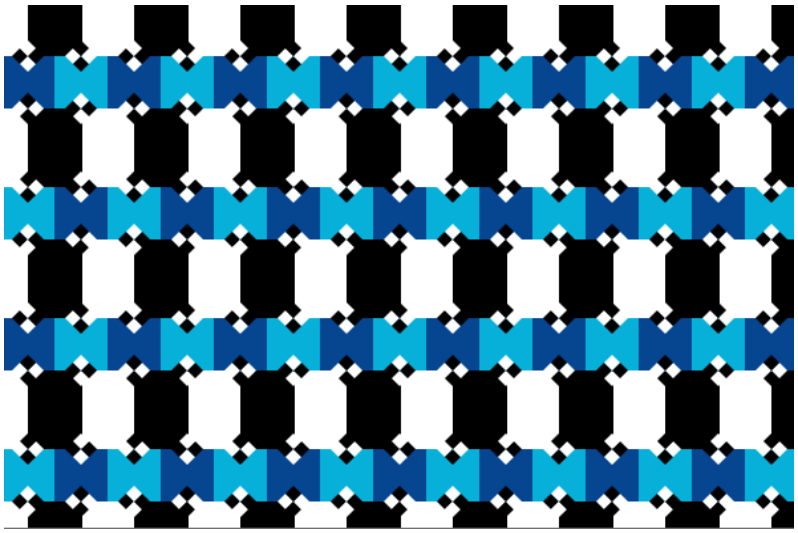


Bild: Screenshot von Ausführung des Programms „alg08b_weitere_uebungen_loesungen“ (Eisenmann)

1. *Programmiere zunächst die schwarzen, senkrechten Balken.*
2. *Programmiere die waagrechten Balken, in denen sich hell- und dunkelblaue Quadrate abwechseln.*
3. *Um die kleinen Vierecke – zusammengesetzt aus vier Rauten – zeichnen lassen zu können, brauchst du die wie in Aufgabe 1 die Methode* `quad()`*. Schreibe eine Methode* `viereck()`*, der du die Farbe und alle Koordinaten der Eckpunkte übergeben kannst und dann eine Methode* `vier_rauten()`*, der eine Farbe und die Koordinaten eines Punktes übergeben werden können. Hier bietet sich der Punkt an, in dem sich alle vier Rauten treffen.*
   (Wenn du Aufgabe 1 gelöst hast, kannst du selbstverständlich die Methode `viereck()` *von dort nutzen.)*
4. *Ergänze dein Programm so, dass die optische Täuschung gezeichnet wird.*
5. *Schreibe ggf. eine zweite Version so, dass die optische Täuschung das ganze Fenster füllt.*

Mögliche Lösung zu Aufgabe 2:

1. Man könnte hier auch zunächst eine Zählschleife verwenden, die Lösung ist aber gleich mit einer while-Schleife programmiert, um Aufgabe 5 mit zu erledigen.

Zunächst werden die Breite und die beiden Blautöne als globale Variablen deklariert und initialisiert.

```
// globale Variablen
int breite = 40; // Breite der schwarzen Streifen
int hellblau = #07B0D8; // r=7, g=176, b=216
int dunkelblau = #064590; // r=6, g=69, b=144
```

```
void schwarze_streifen() {
  fill(0); // Füllfarbe schwarz
  int pos = breite/2; // Abstand erster Streifen zum linken Rand
  while (pos < width) {
    rect(pos, 0, breite, height);
    pos = pos + 2*breite;
  }
}
```

2. Zunächst wurde eine Methode zum Farbwechsel geschrieben:

```
int farbwechsel(int farbe0) { // ändert zwischen hell- und dunkelblau
  int neuefarbe;
  if (farbe0 == hellblau) {
    neuefarbe = dunkelblau;
  } else {
    neuefarbe = hellblau;
  }
  return neuefarbe;
}
```

In der Methode für die blauen Streifen werden zunächst zwei lokale Variablen deklariert und initialisiert. Einmal die Startfarbe (jeweils für die erste Farbe der Reihe) und dann die Farbe, die für das aktuelle Rechteck gilt.

Außerdem brauchen wir zwei Variablen, die sich in der verschachtelten while-Schleife ändern: Der Abstand nach oben (von Reihe zu Reihe verändert) und der Abstand nach links (von Quadrat zu Quadrat verändert in jeder Reihe).

```
void blaue_streifen() {
  int startfarbe = dunkelblau;
  int farbe = startfarbe;
  int d_nach_oben = breite;  // Abstand zum oberen Rand
  int d_nach_links = 0; // Abstand nach links

  noStroke();
  while (d_nach_oben < height) {
    while (d_nach_links < width) {
      fill(farbe);
      rect(d_nach_links, d_nach_oben, breite, breite);
      d_nach_links = d_nach_links + breite;
      farbe = farbwechsel(farbe);
    }
    d_nach_oben = d_nach_oben + breite + breite + breite/2;
    d_nach_links = 0;
    startfarbe = farbwechsel(startfarbe);
  }
}
```

Innere Schleife: Solange der Abstand nach links noch kleiner ist als die Breite des Fensters, wird ein Rechteck gezeichnet, der Abstand verändert und die Farbe des Quadrats gewechselt.

Nach Ende der inneren Schleife, wird der Abstand nach oben verändert, der Abstand nach links wieder auf 0 gesetzt und die Startfarbe gewechselt.

Das Gesamte wird so lange wiederholt bis der Abstand nach oben größer als die Höhe des Fensters ist.

3. Methode zum Zeichnen eines Vierecks:

```
void viereck(int farbe, int x1, int y1, int x2, int y2, int x3, int y3, int x4, int y4) {
  fill(farbe);
  quad(x1,y1,x2,y2,x3,y3,x4,y4);
}
```

Für die Überlegung, wie die vier kleinen Rauten gezeichnet werden müssen, macht es Sinn, sich eine Skizze zu zeichnen. Für die Länge der Diagonale einer der Rauten ist eine Variable a eingesetzt worden.

Der Farbwechsel zwischen schwarz und weiß wird durch `255 - farbe` ausgeführt. Ist der Wert von farbe = 0, so erhalten wir 255 – 0 = 255, sonst 255 – 255 = 0.

```
void vier_rauten(int farbe, int x_m, int y_m){
  int a = 12;
  noStroke();
  //Raute links
  viereck(farbe,x_m-a, y_m, x_m - a/2, y_m-a/2, x_m, y_m, x_m - a/2, y_m+a/2);
  //Raute unten
  viereck(255-farbe, x_m, y_m, x_m - a/2, y_m + a/2, x_m, y_m + a, x_m + a/2, y_m + a/2);
  //Raute rechts
  viereck(farbe, x_m, y_m, x_m + a/2, y_m + a/2, x_m + a, y_m, x_m + a/2, y_m - a/2);
  //Raute oben
  viereck(255-farbe, x_m, y_m, x_m - a/2, y_m - a/2, x_m, y_m - a, x_m + a/2, y_m - a/2);
}
```

4. Jetzt fehlt nur noch der Aufruf der Methode aus Aufgabe 3 in der Methode, die die blauen Streifen zeichnet.

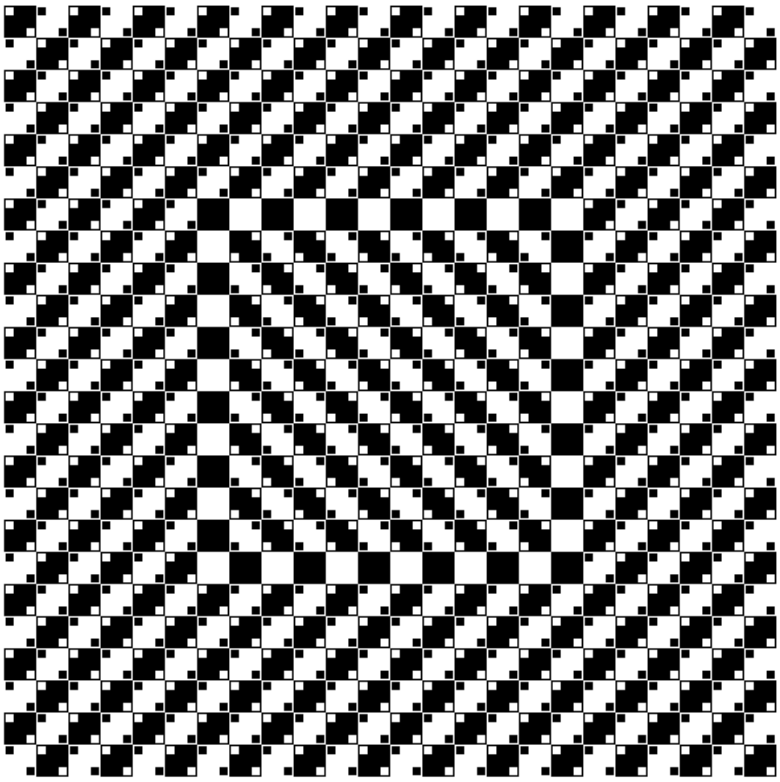
```
void blaue_streifen() {
  int startfarbe = dunkelblau;
  int farbe = startfarbe;
  int d_nach_oben = breite;  // Abstand zum oberen Rand
  int d_nach_links = 0; // Abstand nach links
  int farbe_raute = 0;
  noStroke();
  while (d_nach_oben < height) {
    while (d_nach_links < width) {
      fill(farbe);
      rect(d_nach_links, d_nach_oben, breite, breite);
      vier_rauten(farbe_raute,d_nach_links + breite/2 ,d_nach_oben); // obere Rauten
      vier_rauten(255-farbe_raute,d_nach_links + breite/2,d_nach_oben + breite); // untere
      d_nach_links = d_nach_links + breite;
      farbe = farbwechsel(farbe);
      farbe_raute = 255 - farbe_raute;
    }
    d_nach_oben = d_nach_oben + breite + breite + breite/2;
    d_nach_links = 0;
    startfarbe = farbwechsel(startfarbe);
  }
}
```

Mögliche Lösung zu Aufgabe 2:

```
// globale Variablen
int breite = 40; // Breite der schwarzen Streifen
int hellblau = #07B0D8; // r=7, g=176, b=216
int dunkelblau = #064590; // r=6, g=69, b=144

void schwarze_streifen() {
  fill(0); // Füllfarbe schwarz
  int pos = breite/2; // Abstand erster Streifen zum linken Rand
  while (pos < width) {
    rect(pos, 0, breite, height);
    pos = pos + 2*breite;
  }
}

void blaue_streifen() {
  int startfarbe = dunkelblau;
  int farbe = startfarbe;
  int d_nach_oben = breite;  // Abstand zum oberen Rand
  int d_nach_links = 0; // Abstand nach links
  int farbe_raute = 0;
  noStroke();
  while (d_nach_oben < height) {
    while (d_nach_links < width) {
      fill(farbe);
      rect(d_nach_links, d_nach_oben, breite, breite);
      vier_rauten(farbe_raute, d_nach_links + breite/2, d_nach_oben); // obere Rauten
      vier_rauten(255-farbe_raute, d_nach_links + breite/2, d_nach_oben + breite); // unter
      d_nach_links = d_nach_links + breite;
      farbe = farbwechsel(farbe);
      farbe_raute = 255 - farbe_raute;
    }
    d_nach_oben = d_nach_oben + breite + breite + breite/2;
    d_nach_links = 0;
    startfarbe = farbwechsel(startfarbe);
  }
}

int farbwechsel(int farbe0) { // ändert zwischen hell- und dunkelblau
  int neuefarbe;
  if (farbe0 == hellblau) {
    neuefarbe = dunkelblau;
  } else {
    neuefarbe = hellblau;
  }
  return neuefarbe;
}
```

```
void viereck(int farbe, int x1, int y1, int x2, int y2, int x3, int y3, int x4, int y4) {
  fill(farbe);
  quad(x1,y1,x2,y2,x3,y3,x4,y4);
}

void vier_rauten(int farbe, int x_m, int y_m) {
  int a = 12;
  noStroke();
  //Raute links
  viereck(farbe, x_m-a, y_m, x_m - a/2, y_m-a/2, x_m, y_m, x_m - a/2, y_m+a/2);
  //Raute unten
  viereck(255-farbe, x_m, y_m, x_m - a/2, y_m + a/2, x_m, y_m + a, x_m + a/2, y_m + a/2);
  //Raute rechts
  viereck(farbe, x_m, y_m, x_m + a/2, y_m + a/2, x_m + a, y_m, x_m + a/2, y_m - a/2);
  //Raute oben
  viereck(255-farbe, x_m, y_m, x_m - a/2, y_m - a/2, x_m, y_m - a, x_m + a/2, y_m - a/2);
}

void setup() {
  size(600, 400);
  background(255);  // weißer Hintergrund
  schwarze_streifen();
  blaue_streifen();
}
```

Aufgabe 3:



Bilder: Screenshots von Ausführung des Programms „alg08c_weitere_uebungen_loesungen“ (Eisenmann)

1. *Schreibe eine Methode `zeichneQuadrat()`, der du neben der Startposition, die Grundfarbe und die Richtung der beiden kleinen Quadrate übergeben kannst.*
2. *Programmiere die optische Täuschung zuerst so, dass das ganze Feld so gefüllt wird, wie in den ersten Reihen.*
3. *Überlege dir anschließend, wie du die Mitte zeichnen lassen kannst und probiere deine Idee aus.*

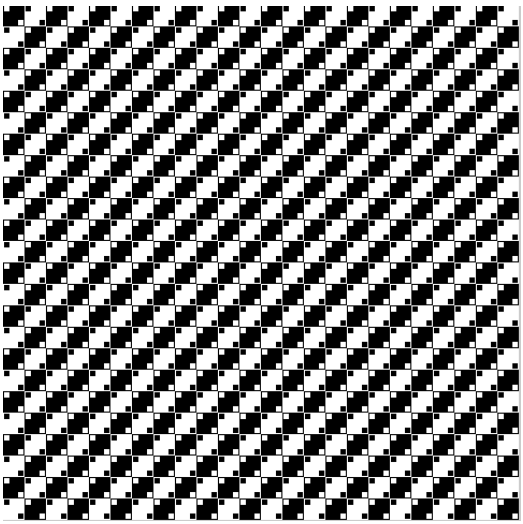
Mögliche Lösung zu Aufgabe 3:

1. Methode zum Zeichnen eines Quadrats mit zwei kleinen Quadraten in der jeweils anderen Farbe:

```
int s=20; // Seitenlaenge Quadrat
int k=5; // Seitenlaenge kleines Quadrat
int d=1; // Abstand kleine Quadrate zum Rand

void zeichneQuadrat(int x, int y, int farbe, boolean minus) {
  fill(farbe);
  noStroke();
  rect(x, y, s, s);
  fill(255-farbe); // Farbwechsel zwischen weiß und schwarz
  if (minus == true) { // links oben, rechts unten
    rect(x+d, y+d, k, k);
    rect(x+s-d-k, y+s-d-k, k, k);
  } else { // links unten, rechts oben
    rect(x+d, y+s-d-k, k, k);
    rect(x+s-d-k, y+d, k, k);
  }
}
```

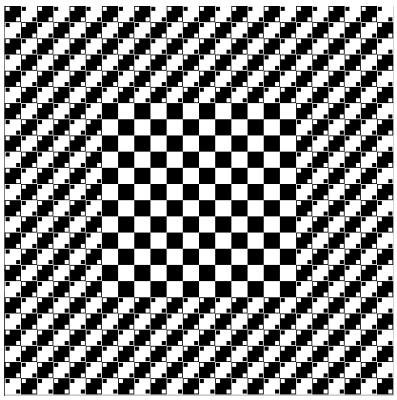
2. Zunächst werden 24 Reihen mit 24 Quadraten gefüllt, ohne auf die Mitte zu achten:

```
void optischeTaeuschung() {
  int farbe_akt = 0;
  for (int j=0; j<24; j++) {
    for (int i=0; i<24; i++) {
      zeichneQuadrat(i*s, j*s, farbe_akt, true);
      farbe_akt = 255 - farbe_akt;
    }
    farbe_akt = 255 - farbe_akt;
  }
}

// setup-Methode
void setup() {
  size(500, 500);
  optischeTaeuschung();
}
```



3. Die Mitte ist zweigeteilt: Der Rand mit schwarzen und weißen Quadraten und das Feld mit den Quadraten, bei denen die Richtung der kleinen Innenquadrate genau umgekehrt ist.

```
// Mitte gefüllt mit schwarz - weiß...
for (int j=0; j<12; j++){
  for (int i=0; i<12; i++){
    fill(farbe_akt);
    rect(6*s +i*s, 6*s+ j*s, s, s);
    farbe_akt = 255 - farbe_akt;
  }
  farbe_akt = 255 - farbe_akt;
}
```



```
// innere Mitte
for (int j=0; j<10; j++) {
  for (int i=0; i<10; i++) {
    zeichneQuadrat(7*s+i*s, 7*s+ j*s, farbe_akt, false);
    farbe_akt = 255 - farbe_akt;
  }
  farbe_akt = 255 - farbe_akt;
}
```

Gesamte Methode:

```
void optischeTaeuschung() {
  int farbe_akt = 0;
  // kompletter Hintergrund
  for (int j=0; j<24; j++) {
    for (int i=0; i<24; i++) {
      zeichneQuadrat(i*s, j*s, farbe_akt, true);
      farbe_akt = 255 - farbe_akt;
    }
    farbe_akt = 255 - farbe_akt;
  }
  // Mitte gefüllt mit schwarz - weiß...
  for (int j=0; j<12; j++){
    for (int i=0; i<12; i++){
      fill(farbe_akt);
      rect(6*s +i*s, 6*s+ j*s, s, s);
      farbe_akt = 255 - farbe_akt;
    }
    farbe_akt = 255 - farbe_akt;
  }
  // innere Mitte
  for (int j=0; j<10; j++) {
    for (int i=0; i<10; i++) {
      zeichneQuadrat(7*s+i*s, 7*s+ j*s, farbe_akt, false);
      farbe_akt = 255 - farbe_akt;
    }
    farbe_akt = 255 - farbe_akt;
  }

}

// setup-Methode
void setup() {
  size(500, 500);
  optischeTaeuschung();
}
```

Processing bietet noch viele weitere Möglichkeiten. :)

Aufgabe 4:

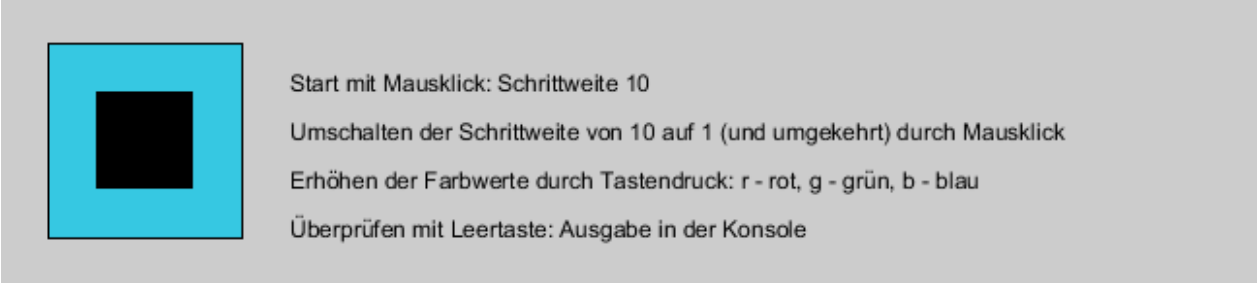


Bild: Screenshot von Ausführung des Programms „alg08d_weitere_uebungen_loesungen“ (Eisenmann)

Hier geht es um ein Farbenspiel. Das große Quadrat wird mit einer zufälligen Farbe gefärbt. Die einzelnen Farbwerte (rot, grün und blau) des inneren Quadrates kann man durch Tastaturdruck verändern (siehe Anleitung auf dem Screenshot). Mit der Leertaste (ASCII-Code: 32) wird überprüft und man bekommt die Information, ob die Farben zu viel, zu wenig oder genau richtig gewählt sind.

1. *Schreibe eine Methode `vorbereitung()`, in der du die beiden Quadrate zeichnen und den Text ausgeben lässt. Rufe diese Methode in der `setup()`-Methode auf. Nutze für die Farben jeweils zwei Variablen (z.B. `rot` und `rot_hintergrund`).*
2. *Für das Spiel brauchst du Maus- und Tastaturereignisse. Lies dir die Information unten auf diesem Blatt durch und programmiere dann Schritt für Schritt die nötigen Ereignismethoden. Denke dabei ans Testen! (Lasse dir z.B. die Farbwerte in der Konsole ausgeben: `println(…)`.)*

Möchte man in Processing **Maus- und Tastaturereignisse** nutzen, so braucht man die Methode `draw()`. Diese Methode wird ständig neu aufgerufen. Die Häufigkeit der Aufrufe gibt man in der `setup()`-Methode mit `frameRate(a)` vor. Der Wert a steht für die Anzahl der Aufrufe pro Sekunde.

Braucht man keinen ständigen Aufruf wie in unserem Programm hier in der Aufgabe, so nutzt man `noLoop()` in der `setup()`-Methode. Durch `redraw()` kann man dann aktiv die `draw()`-Methode aufrufen. In der Aufgabe bietet sich das bei jedem Tastendruck an.

Als Mausereignis brauchen wir `mousePressed()`, um zwischen den beiden Schrittweiten umzuschalten und als Tastaturereignis `keyPressed()`. Hier bekommt man bei jedem Tastendruck einen Wert in der Variable `key` übergeben. Zur Überprüfung der Buchstabentasten kann man direkt mit den Buchstaben arbeiten (z.B. `if (key == 'b')` …). Nutzt man Sondertasten, wie hier die Leertaste, so braucht man den zugehörigen ASCII-Code.

Die Mausposition bei den Mausereignissen erhält man mit `mouseX` und `mouseY`.

Weitere Informationen findest du in der Processing-Referenz[2] unter Input.

3. *Ergänze dein Programm nach Belieben (Versuche zählen; Möglichkeit, auch den Farbwert kleiner zu machen; …).*

2 *https://processing.org/reference/* (abgerufen am 26.04.2019)

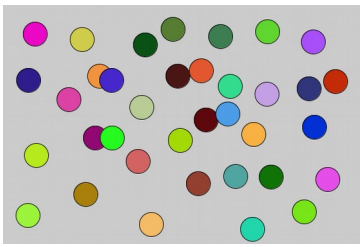
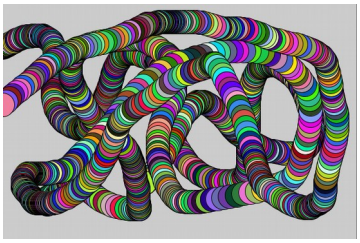
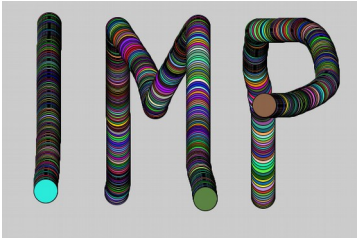
Mögliche Lösung zu Aufgabe 4:

```
int rot = 0;
int blau = 0;
int gruen = 0;
int h_rot;
int h_blau;
int h_gruen;
boolean schnell = false;

void setup() {
  size(650, 150);
  vorbereitung();
  //background(h_rot,h_gruen,h_blau); */
  noLoop();
}

void vorbereitung() {
  h_rot = int(random(0, 255));
  h_gruen = int(random(0, 255));
  h_blau= int(random(0, 255));
  fill(h_rot, h_gruen, h_blau);
  rect(25, 25, 100, 100);
  fill(0);
  text("Start mit Mausklick: Schrittweite 10", 150, 50);
  text("Umschalten der Schrittweite von 10 auf 1 (und umgekehrt) durch Mausklick", 150, 75);
  text("Erhöhen der Farbwerte durch Tastendruck: r - rot, g - grün, b - blau", 150, 100);
  text("Überprüfen mit Leertaste: Ausgabe in der Konsole", 150, 125);
}

void draw() {
  noStroke();
  fill(rot, gruen, blau);
  rect(50, 50, 50, 50);
}
```

```
void ueberpruefen() {
  String ausgabe_rot;
  String ausgabe_gruen;
  String ausgabe_blau;
  if (rot>h_rot) {
    ausgabe_rot = "zu viel rot";
  } else if (rot < h_rot) {
    ausgabe_rot = "zu wenig rot";
  } else {
    ausgabe_rot = "rot stimmt!";
  }
  if (gruen>h_gruen) {
    ausgabe_gruen = "zu viel grün";
  } else if (gruen < h_gruen) {
    ausgabe_gruen = "zu wenig grün";
  } else {
    ausgabe_gruen = "grün stimmt!";
  }
  if (blau>h_blau) {
    ausgabe_blau = "zu viel blau";
  } else if (blau < h_blau) {
    ausgabe_blau = "zu wenig blau";
  } else {
    ausgabe_blau = "blau stimmt!";
  }
  println(ausgabe_rot+", "+ausgabe_gruen+", "+ausgabe_blau);
}
 void keyPressed() {
   int d;
   if (schnell == true) {
     d=10;
   } else {
     d=1;
   }
   if (key == 32) {  // Leertaste
     ueberpruefen();
   }
   if (key == 'r' || key == 'R') {
     rot = (rot + d) % 255;
     println("rot: "+rot+" richtig: "+h_rot);
   } else if (key == 'g' || key == 'G') {
     gruen = (gruen+d) % 255;
     println("grün: "+gruen+" richtig: "+h_gruen);
   } else if (key == 'b' || key == 'B') {
     blau = (blau+d) % 255;
     println("blau: "+blau+" richtig: "+h_blau);
   }
   redraw();
 }

 void mousePressed() {
   schnell = ! schnell;
   println(schnell);
 }
```

Aufgabe 5:

Hier brauchst du die Informationen aus Aufgabe 4.

Bilder: Screenshots von Ausführung des Programms „alg08e_weitere_uebungen_loesungen“ (Eisenmann)

1. *Beim Klick auf das Fenster soll ein Kreis mit zufälliger Füllfarbe an der Mausposition gezeichnet werden. (Abbildung 1)*
2. *Wird die Maus über das Fenster bewegt (`mouseMoved()`), soll an der Mausposition ein Kreis mit zufälliger Füllfarbe gezeichnet werden. (Abbildung 2)*
3. *Nur bei gedrückter Maustaste wird gezeichnet (`mouseDragged()`). Gezeichnet wird dann wie in Aufgabe 2. (Abbildung 3)*
4. ****** Stelle Processing auf den Android-Modus um. Ändere die Größenangabe in der `setup()`-Methode auf `size(displayWidth,displayHeight)` und teste dann Nr. 3 auf dem Smartphone.*

Mögliche Lösung zu Aufgabe 5:

Der Inhalt der Mausereignismethoden ist immer derselbe. Der Unterschied ist das Ereignis, das das Zeichnen auslöst.

Hier sind die Ereignisse aus 1. und 3. dargestellt:

```
void setup() {
  size(600, 400);
}

void draw() {
}

void mousePressed() {
  fill(int(random(0, 255)), int(random(0, 255)), int(random(0, 255)));
  ellipse(mouseX, mouseY, 40, 40);
}

void mouseDragged() {
  fill(int(random(0, 255)), int(random(0, 255)), int(random(0, 255)));
  ellipse(mouseX, mouseY, 40, 40);
}
```

Änderung für Android-Version:

```
void setup() {
  size(displayWidth, displayHeight);
}
```