Arbeitsblatt 6: Programmierung geometrischer Figuren

Die Karten, auf denen die Lärmmessungen dargestellt werden, bestehen aus einer Vielzahl geometrischer Formen. Diese geometrischen Formen ergeben zusammen die Karte. In den folgenden Aufgaben lernst du, wie solche Elemente auf dem Computer gezeichnet werden können.

Aufgabe 1: Bilder aus Einzelteilen erzeugen

➔ Überlege dir, wie die folgenden Bilder erstellt werden können. Aus welchen geometrischen Elementen bestehen sie? Benenne alle vorkommenden geometrischen Elemente in den Bildern unten (schreibe direkt in die Bilder). Du darfst dabei nur die folgenden geometrischen Elemente verwenden:

- Punkte
- Linien
- Rechtecke
- Ellipsen (Kreise)

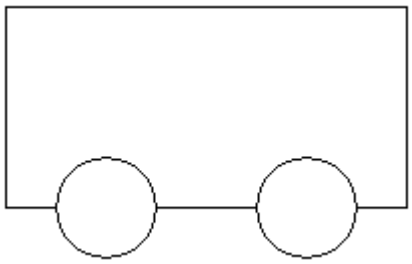


Bild 1

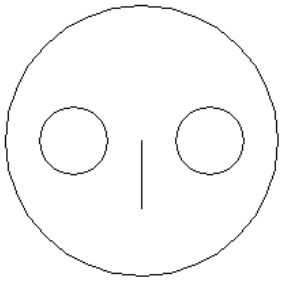


Bild 2

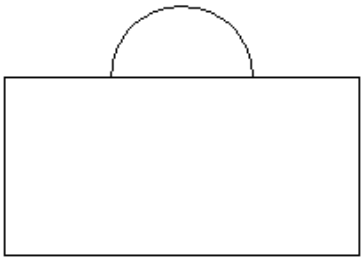


Bild 3

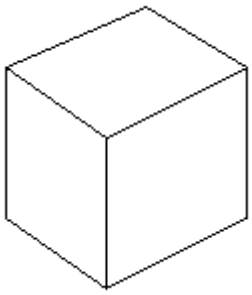
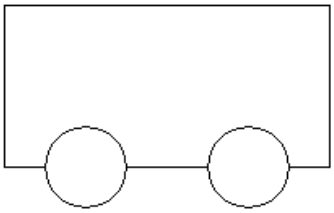
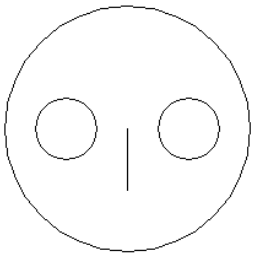
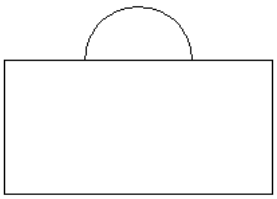
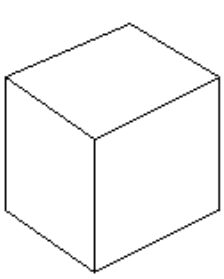


Bild 4

Aufgabe 2: In Computersprache übersetzen

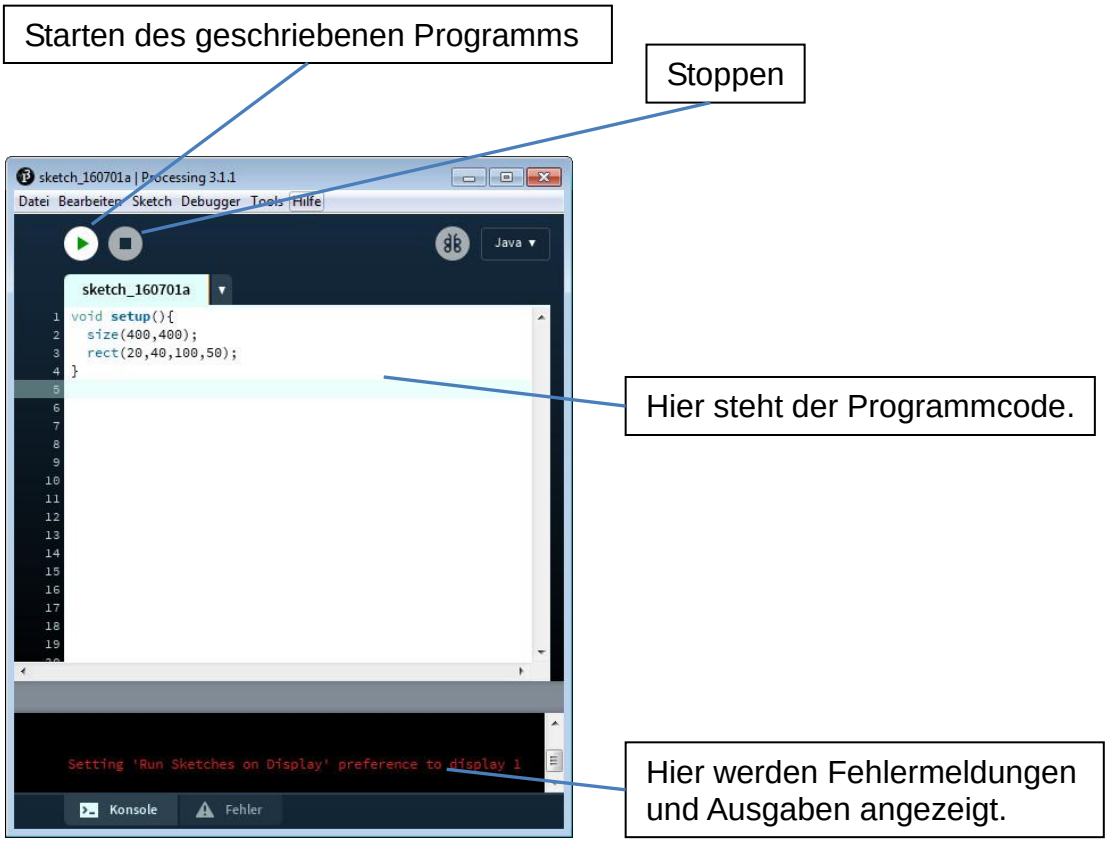
Damit der Computer verstehen kann, was du zeichnen möchtest, müssen die Befehle in «seine» Sprache übersetzt werden. Eine solche Sprache nennt man Code oder Programmcode.

➔ Überlege dir, wie du die Objekte aus Aufgabe 1 bezeichnen kannst, damit der Computer genau das versteht, was du meinst. **Benutze dabei die Informationsblätter 2 und 3.** Diese enthalten alle notwendigen Informationen und helfen dir beim «Übersetzen» in den Programmcode.

Bild	Code
	`ellipse(100,100,150,50);`
	
	
	
	

Aufgabe 3: Erstes Programm schreiben

Nun schreibst du dein erstes Programm. Dies tust du mit der Software «Processing». Wenn «Processing» nicht bereits geöffnet ist, wird es einfach durch Doppelklick auf die Anwendung «processing.exe» gestartet.



➔ Gib im Feld, in dem der Programmcode stehen soll, Folgendes ein, und starte das Programm durch Drücken des Pfeilsymbols:

```
void setup(){
  size(400,400);
  rect(20,40,100,50);
}
```

Zwischen den geschweiften Klammern **{ }** stehen die Anweisungen an den Computer, der unsere Figuren zeichnen soll. Als Erstes wird hier jeweils noch angegeben, wie gross der «Kartenausschnitt» («size») sein soll. In diesem Fall ist es eine Fläche der Grösse 400 Pixel auf 400 Pixel: `size(400,400);`

➔ Ersetze nun den Befehl `rect(20,40,100,50)` durch die Anweisungen (den Programmcode), die du in Aufgabe 2 aufgeschrieben hast. Funktioniert es? Wenn nicht, korrigiere deinen Programmcode, bis es stimmt.

Aufgabe 4: Farben

Bis hier waren unsere Bilder nur aus den «Farben» Grau, Weiss und Schwarz. Ab jetzt soll Farbe ins Spiel kommen.

Auftrag A: Die Hintergrundfarbe

Die Hintergrundfarbe kann mit dem Befehl `background()` gesetzt werden. Diese Anweisung benötigt drei Werte. Der erste Wert gibt an, wie gross der Rotanteil ist, der zweite Wert gibt an, wie gross der Grünanteil ist, und der dritte Wert gibt an, wie gross der Blauanteil ist. Diese Farben werden auch **RGB**-Farben genannt (Rot-Grün-Blau-Farben). Die Werte liegen bei allen drei Farben zwischen 0 (nicht vorhanden) und 255 (so viel wie möglich).

Nimm das folgende Programm als Ausgangslage:

```
void setup(){
  size(400,400);
  background(0,0,0);
}
```

➔ Fülle nun die folgende Tabelle aus, indem du die Zahlen unten ins obige Programm einsetzt und das Ergebnis betrachtest. Welche Hintergrundfarben ergeben sich bei den gegebenen Farben?

Rot	Grün	Blau	Farbe?
0	0	0	
255	255	255	
255	0	0	
0	255	0	
0	0	255	
255	255	0	
0	255	255	
255	0	255	
120	0	0	
100	255	255	
255	150	150	

Natürlich können auch geometrische Formen farbig gezeichnet werden:

Mit der Anweisung `fill()` kann die Farbe geändert werden, mit der die gezeichneten Objekte ausgefüllt werden. Diese Anweisung muss immer vor der Anweisung zur geometrischen Figur stehen! Auch hier werden wieder die drei Werte für Rot, Grün und Blau angegeben.

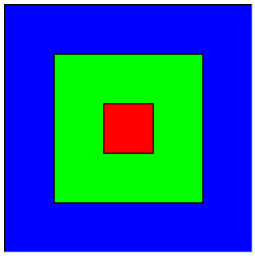
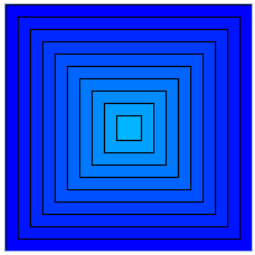
Beispiel:

```
void setup(){
  size(400,400);
  fill(0,0,0);
  rect(100,100,200,200);
}
```

➔ Welche Farbe hat das gezeichnete Quadrat?

Auftrag B: Farbige Figuren Zeichnen

Zeichne die folgenden zwei Figuren in «Processing», und schreibe den Code in die Tabelle:

HINWEIS (freiwilliger Zusatzauftrag): Mit dem Befehl `noStroke()` werden die schwarzen Ränder der Rechtecke nicht mehr gezeichnet.

Aufgabe 5: Variablen

In der Programmierung von beweglichen Objekten (z. B. in Spielen) können sich die Zustände (Werte) des Hintergrunds oder der Objekte (z. B. Spielfiguren) verändern. Diese Zustände müssen irgendwo gespeichert werden. Man möchte sich merken, wo sich beispielsweise in einem Spiel der Gegner oder das Ziel gerade befindet. Dazu werden Variablen benötigt.

Werte in «Processing»

In «Processing» gibt es einige Schlüsselwörter (sogenannte Befehle), die für Werte stehen. Einige Beispiele dafür sind:

`height`	➔ Höhe des Fensters
`width`	➔ Breite des Fensters
`mouseX`	➔ x-Koordinate der Maus
`mouseY`	➔ y-Koordinate der Maus

Soll eine Linie von links oben nach rechts unten gezeichnet werden, kann das in dieser Schreibweise geschrieben werden:

```
line(0,0,width,height);
```

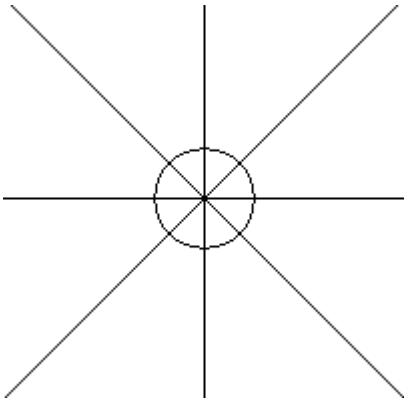
Der grosse Vorteil dieser Schreibweise ist, dass mit diesen Werten auch gerechnet werden kann.

Code	**Bedeutung**
`+`	Addition
`-`	Subtraktion
`/`	Division (ganzzahlig)
`*`	Multiplikation

Soll also ein Kreis in der Mitte des Fensters gezeichnet werden, sieht das so aus:

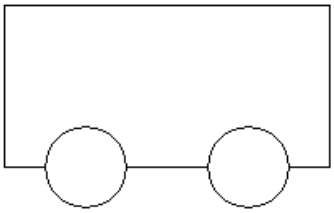
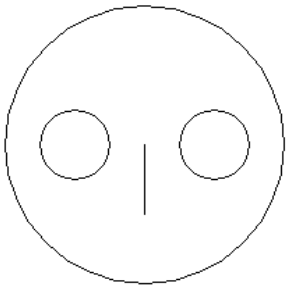
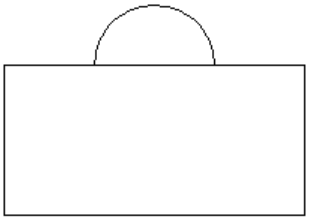
```
ellipse(width/2,height/2,50,50);
```

➔ Zeichne mithilfe von Rechenoperationen und den genannten Schlüsselwörtern das folgende Bild.



Das Bild soll immer gleich gezeichnet werden, auch wenn die Werte in `size()` (also der Grösse des Fensters) verändert werden.

➔ Zeichne mindestens eines der Bilder der Aufgabe 2 so, dass es immer in der Mitte des Fensters ist, egal was für eine Grösse für das Fenster am Anfang gewählt wird.

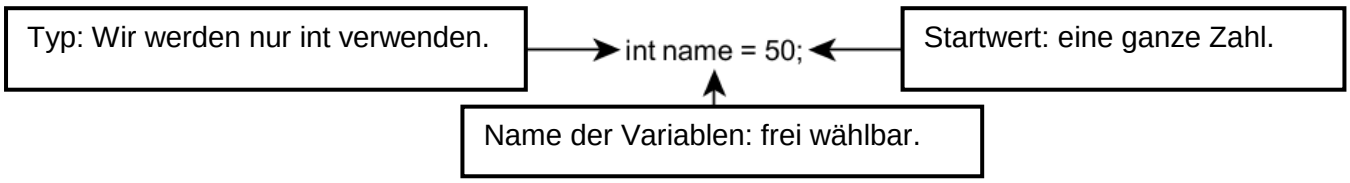
Bild	**Code**
	
	
	

Nun kannst du deinen Code testen, indem du den obigen Code im folgenden Programm an der markierten Stelle einfügst:

```
void setup(){
  size(400,400);
// bedeutet, dass das Fenster eine veränderbare Grösse hat.
  frame.setResizable(true);
}
void draw(){
  // Zeichencode
}
```

Aufgabe 6: Eigene Variablen (für die etwas Schnelleren)

Bezeichnung einer eigenen Variablen:



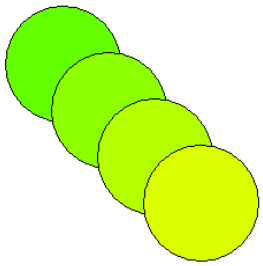
In dieser Aufgabe werden wir eine Variable definieren und deren Wert anschliessend zum Zeichnen verwenden. Die Definition der Variablen muss ganz am Anfang des Programms stehen. Die Variable mit dem Namen `pos` wird auf den Startwert 100 gesetzt. Dann wird dieser Wert an vier Stellen im Programm verwendet. Bei der Verwendung kann dem Wert auch etwas hinzugezählt oder abgezogen werden.

```
int pos=100;
void setup(){
  size(400,400);
  background(255,255,255);
  ellipse(pos,100,100,100);
  ellipse(pos+10,100,100,100);
  ellipse(pos+20,100,100,100);
  ellipse(pos+30,100,100,100);
}
```

➔ Kopiere den obigen Code, teste ihn, und mache anschliessend folgende Änderungen:

a) Ändere die Variable `pos` so, dass der Mittelpunkt des ersten Kreises nicht an der Position 100/100 ist, sondern an der Position 50/100.
b) Ändere das Programm so ab, dass sich die Kreise weniger überlappen.
c) Verwende die Variable `pos` auch noch für das Setzen beziehungsweise die Änderung der Farbe.
d) Der Kreis soll diagonal wandern. Ändere den Code entsprechend.

Das Resultat könnte am Ende beispielsweise so aussehen:



➔ Im vorangehenden Auftrag hast du den am Anfang in der Variablen gespeicherten Wert immer wieder verwendet, aber nicht verändert. Der folgende Code macht grafisch genau das Gleiche, nun wird der Wert von `pos` aber jeweils verändert.

```
int pos=100;
void setup(){
  size(400,400);
  background(255,255,255);
  ellipse(pos,100,100,100);
  pos=pos+10;
  ellipse(pos,100,100,100);
  pos=pos+10;
  ellipse(pos,100,100,100);
  pos=pos+10;
  ellipse(pos,100,100,100);
}
```

← Wert von `pos` = 100

← Wert von `pos` = 110

← Wert von `pos` = 120

← Wert von `pos` = 130

Kopiere den obigen Code, teste ihn, und mache anschliessend wieder folgende Änderungen:

a) Ändere die Variable `pos` so, dass der Mittelpunkt des ersten Kreises nicht an der Position 100/100 ist, sondern an der Position 50/100.
b) Ändere das Programm so ab, dass sich die Kreise weniger überlappen.
c) Verwende die Variable `pos` auch noch für das Setzen beziehungsweise die Änderung der Farbe.
d) Der Kreis soll diagonal wandern. Ändere den Code entsprechend.

Welche Variante bevorzugst du? Die Version des ersten oder des zweiten Auftrags? Warum?