Fachhochschule Nordwestschweiz

Schnupperkurs Programmieren

Studiengang Informatik inkl. Profilierung iCompetence

Prof. Dr. Barbara Scheuner, *Co-Studiengangleiterin Informatik iCompetence*
Sibylle Peter, *Dozentin Software Engineering*

Information über das Studium

Kontaktinformationen:

- Fragen zum Studium: barbara.scheuner@fhnw.ch
- Fragen zur Anmeldung: admin.technik@fhnw.ch

Webseiten:

- Hochschule für Technik: https://www.fhnw.ch/de/studium/technik
- Studiengang Informatik: https://www.fhnw.ch/de/studium/technik/informatik
- Profilierung iCompetence: https://www.fhnw.ch/de/studium/technik/icompetence

Erste Programmierschritte mit Processing

Inhalt

Was ist Processing?

> Processing funktioniert als ein flexibles Software-Skizzenbuch und wurde speziell als Sprache entwickelt um Programmieren zu lernen. Seit 2001 fördert Processing die Software-Kompetenz in der bildenden Kunst und die visuelle Kompetenz in der Technologie. Zehntausende von Studenten, Künstlern, Designern, Forschern und Hobbyisten nutzen Processing zum Lernen und für Prototypen. In Processing kann in den Sprachen Java, Python und JavaScript programmiert werden.
>
> — übersetzt aus: https://processing.org/

Grundlagen

Umgebung

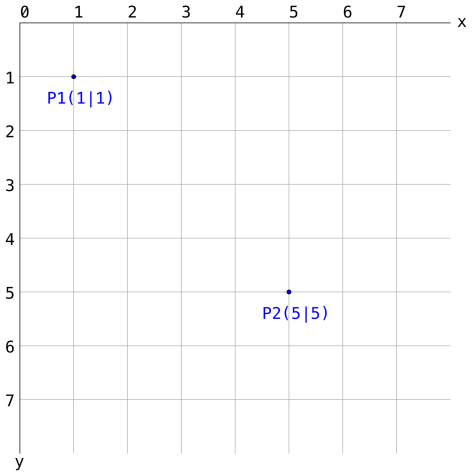
Processing kommt mit einem Editor und einer Laufzeitumgebung. Diese können lokal installiert werden. Wer nichts installieren möchte, kann den Webeditor benutzen.

Processing wurde entwickelt, möglichst einfach grafische Elemente darstellen zu können.

Koordinatensystem

Processing funktioniert mit einem Koordinatensystem, das wahrscheinlich aus der Schule bekannt ist. Es gibt eine x-Achse (horizontal) und eine y-Achse (vertikal). In einem Koordinatensystem können nun durch Bestimmung von x und y Punkte gesetzt werden.

Im Unterschied zu den bekannten Koordinatensystemen aus der Schule ist der Nullpunkt (0|0) in der **oberen** linken Ecke. Wenn `x` grösser wird, bewegt sich der Punkt nach rechts und wenn `y` grösser wird, bewegt sich der Punkt nach unten.

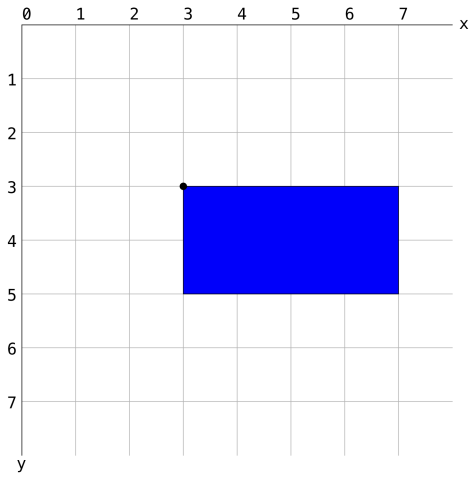


Formen

Processing bietet Methoden an, um verschiedene Formen zu zeichnen. Mit `rect(x,y,l,b)` kann z. B. ein Rechteck gezeichnet werden. Die vier Parameter sind alles Zahlen. `x` und `y` bestimmen den

Startpunkt, welcher, analog zum Koordinatensystem, an der oberen linken Ecke liegt. `l` ist die Ausdehnung in Richtung x-Achse, `b` die Ausdehnung in Richtung y-Achse.

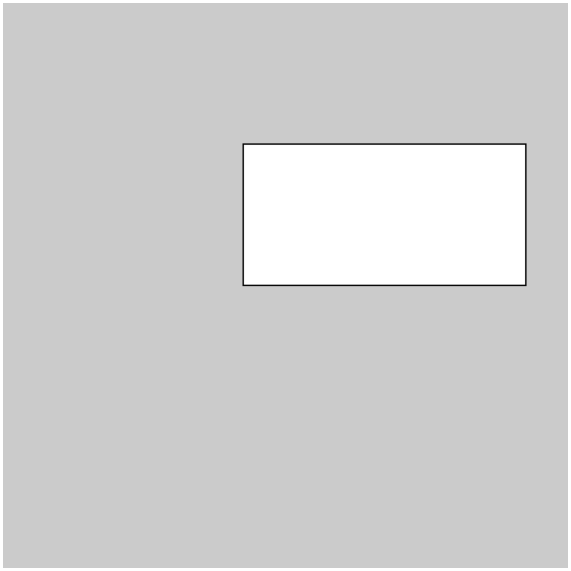
`rect(3,3,4,2)` dargestellt im (vergrösserten) Koordinatensystem: Es beginnt bei Punkt (3|3) und hat in x-Richtung eine Ausdehnung von 4 und in y-Richtung eine Ausdehnung von 2.



In Realität sind die Pixel viel kleiner, der folgende Code zeigt ein weisses Rechteck ausgehend von Punkt (170|100) mit einer Länge von 200 in x-Richtung und einer Breite in y-Richtung von 100. Standardmässig wird es mit einem schwarzen Rand dargestellt.

Damit das Zeichenfenster nicht zu klein ist, wird die Grösse des Zeichenfensters mit der Methode `size(400,400);` auf 400x400 Pixel gesetzt.

```
1 size(400,400);
2 fill(255); // weiss
3 rect(170,100,200,100);
```

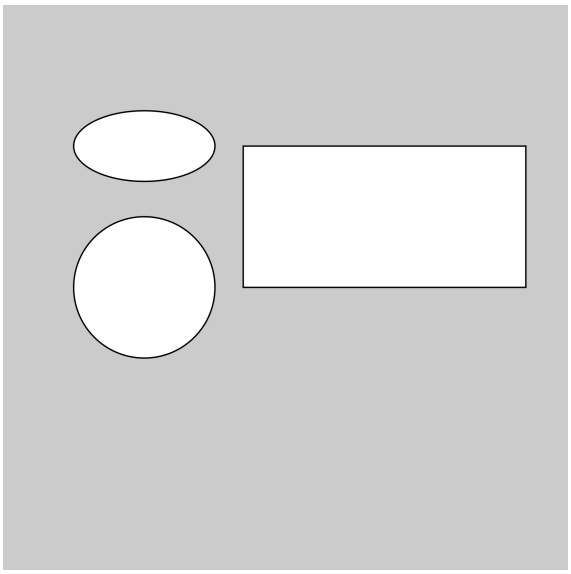


Ein weiteres Element sich Kreise bzw. Ellipsen:

`ellipse(x, y, laenge, breite);` zeichnet eine Ellipse der gewünschten Länge und Breite von dem Startpunkt (x|y) aus. Der Startpunkt liegt hier allerdings in der Mitte der Ellipse, wie auf dem nächsten Bild ersichtlich wird.

Ein Kreis ist eine Ellipse mit gleicher Länge und Breite. Wir ergänzen also unseren Code mit einer Ellipse der gleichen Länge und Breite wie das Rechteck und einem Kreis.

```
1 size(400,400);
2 fill(255);
3 rect(170,100,200,100);
4 ellipse(100,100,100,50);
5 ellipse(100,200,100,100);
```



Natürlich können bei einem auf Graphik spezialisierten Programm noch mehr Elemente gezeichnet werden.

Form	Methode	Beschreibung
Linie	`line(x1,y1,x2,y2)`	Zeichnet eine Line vom Punkt (x1|y1) zum Punkt (x2|y2).
Punkt	`point(x,y)`	Zeichnet einen Punkt an der Position (x|y).
Text	`text(buchstaben, x,y)`	Hier wird zuerst der Text angegeben und dann die Position (unten links des Textfeldes).

Farbe

Wir haben bereits im vorherigen Beispiel gesehen, dass wenn wir die Methode `fill(255);` vor dem Zeichnen der Formen aufrufen, werden diese weiss gefüllt und nicht schwarz (Standardwert).

Für Grautöne haben wir eine Farbtiefe von acht Bit zur Verfügung. Das bedeutet, wir haben acht Stellen, die jeweils 0 oder 1 sein können. Das ergibt insgesamt 256 (2^8) Möglichkeiten. Die Grautöne beginnen bei 0 (= Schwarz) und geht bis zu 255 (= Weiss). Wenn wir `fill();` mit nur einem Parameter aufrufen, wird ein Grauton erzeugt.

Für Farben haben wir dreimal acht Bit zur Verfügung. einmal für Rot, einmal für Grün und einmal für Blau (RGB). Das ergibt insgesamt eine Farbtiefe von 24Bit. D. h. wir brauchen auch mind. drei Parameter (rot, grün, blau) für `fill()`.

`fill(255,0,0);` ergibt daher ein reines, leuchtendes Rot, während `fill(0,255,0)` ein Grün darstellt.

Im RGB Spektrum funktioniert Mischen anders als auf Papier, Rot und Grün gibt Gelb und für Weiss werden alle Farben auf 255 gesetzt, wie auf diesem Bild ersichtlich wird.

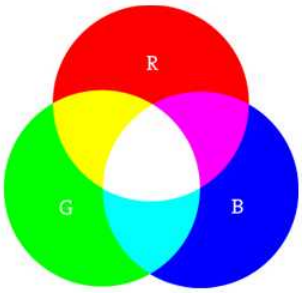
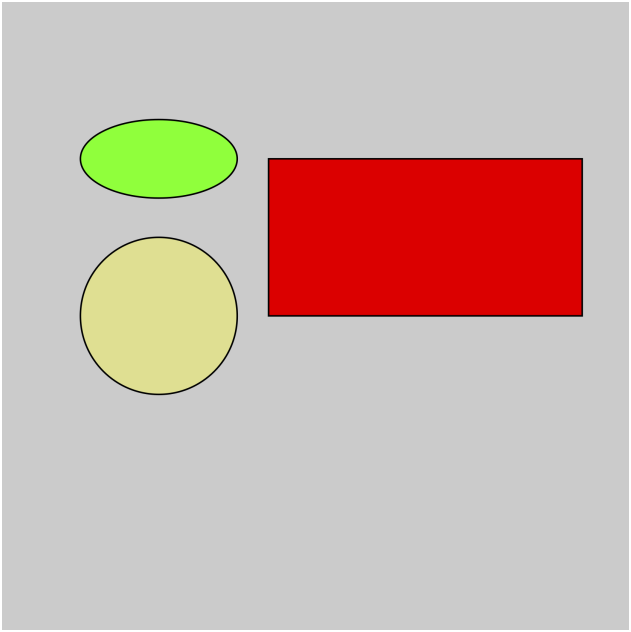


Bild von https://processing.org/tutorials/color

Wenn wir unsere Formen von vorher mit Farbe füllen, sieht das dann so aus.

```
1 size(400,400);
2 fill(255,0,0); //rot
3 rect(170,100,200, 100);
4 fill(0,255,0); //grün
5 ellipse(100,100,100,50);
6 fill(255,255,0, 100); //gelb ①
7 ellipse(100,200,100, 100);
```

① Es gibt noch einen vierten Parameter, den sogenannten *Alpha-Kanal*. Damit können wir die Deckkraft einer Farbe bestimmen. 0 ist vollständig transparent (0% der Farbe) und 255 vollständig deckend und die Default-Einstellung. Aktuell gewählt wurde eine Deckkraft von 100.



Methoden von Processing

Damit das Programm nicht nur einmalig zeichnet, gibt es in Processing eine Methode, welche sich immer wieder wiederholt und im Prinzip jedes Mal etwas neues zeichnet. Das wirkt dann wie ein Daumenkino. Will man diese Methode verwenden muss auch der Start des Programms, der aktuell nur die Definition der Fenstergrösse beinhaltet, in eine Methode geschrieben werden.

```
1 void setup(){  ①
2     size(400,400);
3 }
4 void draw(){ ②
5     ellipse(200,200,100,100);
6 }
```

① In dieser Methode werden die Startwerte gesetzt.

② Diese Methode wird so lange wiederholt bis das Programm geschlossen wird.

Variablen

Ein Grundkonzept der Programmierung ist das Speichern und Verarbeiten von Daten. In unserem kleinen Programm werden wir die Daten in Variablen vom Typ `int` speichern. Es gibt aber noch andere Typen.

```
1 int x=1; ①
2 void setup(){
3     size(400,400);
4 }
5 void draw(){
6     x=x+2; ②
7     ellipse(200,200,x,x); ③
8 }
```

① Deklaration der Variablen x und Setzen des Startwerts auf 1.

② Erhöhen des Wertes von x um 2.

③ Verwenden des Wertes von x.

Verzweigungen (Tests)

Meistens muss in einem Programm ein Test durchgeführt werden. z.B. ob eine bestimmte Taste gedrückt wurde. Je nach Tastendruck wird dann etwas anderes ausgeführt. In Processing kann abgefragt werden, ob eine Taste gedrückt wurde (Wahr oder Falsch) und welche Taste gedrückt wurde.

Testen ob eine Taste gedrückt ist

Mit dem Wort `if` (wenn) kann ein Test eingeführt werden. Nach dem `if` folgen dann `()`-Klammern. In diesen Klammern steht der Ausdruck, der wahr (true) oder falsch (false) ist.

```
1 int x=1;
2 void setup(){
3     size(400,400);
4 }
5 void draw(){
6     x=x+2;
7     if (keyPressed){ ①
8       x=1;
9     }
10    ellipse(200,200,x,x);
11 }
```

① keyPressed gibt an, ob eine Taste gedrückt wurde. Wenn ja, dann wird x wieder auf den Wert 1 gesetzt.

Testen welche Taste gedrückt ist

Die Variable key beinhaltet den Wert der aktuell gedrückten oder als letztes gedrückten Taste.

```
1  int x=1;
2  void setup(){
3      size(400,400);
4  }
5  void draw(){
6      x=x+2;
7      if (keyPressed){
8          if (key == '1') { ①
9              x=1;
10         }
11         if (key == 'r') { ②
12             fill(255,0,0);
13         }
14     }
15     ellipse(200,200,x,x);
16 }
```

① Wenn key den Wert '1' hat, dann wird x auf 1 gesetzt.

② Wenn key den Wert 'r' hat, dann wird die Füllfarbe auf Rot gesetzt.

Schleife

Auch wenn die Methode draw endlos ausgeführt wird, kann es sein, dass man noch eine weitere Schleife braucht. Diese wird dann aber nur eine bestimmte Anzahl oft ausgeführt. Eine dieser Schleifen ist die for-Schleife.

```
1  void setup(){
2      size(400,400);
3  }
4  void draw(){
5      background(0);
6      if (key=='1'){
7        for (int i=0; i<100; i=i+40){ ①
8          rect(i,40,40,40);
9        }
10     }
11     if (key=='2'){
12       for (int i=0; i<200; i=i+20){ ②
13         rect(i,40,40,40);
14       }
15     }
16     if (key=='3'){
17       for (int i=0; i<300; i=i+10){ ③
18         rect(i,40,40,40);
```

```
19        }
20      }
21      if (key=='4'){
22        for (int i=200; i<400; i=i+10){ ④
23          rect(i,40,40,40);
24        }
25      }
26  }
```

① Die erste Schleife geht von 0 bis 100 in 40er Schritten.

② Die zweite Schleife geht von 0 bis 200 in 20er Schritten.

③ Die dritte Schleife geht von 0 bis 300 in 10er Schritten.

④ Die vierte Schleife geht von 200 bis 400 in 10er Schritten.

Bei der `for`-Schleife kann bestimmt werden, bei welcher Zahl gestartet wird, und bis zu welcher Zahl die Schleife laufen soll. Auch kann angegeben werden, wie gross die Schritte dazwischen sind.

`for (int i=` *Startwert* `; i<` *Endwert* `; i=i+` *Schrittgroesse* `) {}`

Das Spiel

Das Nimm-Spiel ist ein einfaches 2-Personen-Spiel. Zu Beginn liegen Streichhölzer auf dem Tisch. Die Streichhölzer werden nun abwechselnd vom Tisch genommen. Man kann dabei wählen, ob man 1, 2 oder 3 Streichhölzer nimmt. Das Ziel ist, **nicht** das letzte Streichholz vom Tisch nehmen zu müssen.

Beispiel für einen Ablauf

Es liegen zu Beginn 10 Hölzer auf dem Tisch.

1. Person 1 nimmt 2 Hölzer → Es sind nun noch 8 Hölzer
2. Person 2 nimmt 1 Holz → Es sind nun noch 7 Hölzer
3. Person 1 nimmt 3 Hölzer → Es sind nun noch 4 Hölzer
4. Person 2 nimmt 3 Hölzer → Es hat noch 1 Holz
5. Person 1 nimmt das letzte Holz → Person 1 hat verloren.

Startprogramm

```
1  int anzahl=31; ①
2  int runde=0; ②
3
4  void setup(){
5    size(1400,600);
6    textSize(100); ③
7    background(0);
8    frameRate(3); ④
9  }
10
11 void draw(){
12   background(0);
13   fill(255);
14   text("Person "+(runde%2+1), 20, 150); ⑤
15   text(anzahl, 20, 300); ⑥
16 }
```

① In der Variablen `anzahl` wird gespeichert, wie viele Zündhölzer noch da sind.

② In der Variablen `runde` wird gespeichert in welcher Runde das Spiel ist.

③ Die Methode `textSize()` setzt die Grösse des Textes (Font-Grösse)

④ Die Methode `frameRate()` gibt an wie viele Bilder pro Sekunde gezeichnet werden sollen. Wenn der Wert auf 0 gesetzt ist, ist es das gleiche wie wenn das Programm stoppt.

⑤ Über den Wert von Runde wird berechnet, wer an der Reihe ist.

Weiterführende Links

Processing

Der beste Einstieg in die Dokumentation zu Processing ist die Webseite, allerdings in Englisch:

- https://processing.org/
  - Tutorial zu Farben
  - Tutorial zum Koordinatensystem, Formen und Transformationen
  - Übersicht Bücher zu Processing

Ein Buch auf Deutsch: (ohne Gewähr)

- Wolf, Matthias.2022. *Einführung ins Programmieren mit Processing. Komplett überarbeitet für Processing 4*. Lulu.com

User Experience, Usability und Interaction Design.

Sowohl Usability wie auch Interaction Design sind Aspekte der User Experience, also der Erfahrungen die Benutzende mit einem Produkt erleben.

Bei der **Usability**, die laut Nielsen neben der Nützlichkeit (oder dem Wert) die zweithöchste Priorität hat, geht es darum **wie gut** Benutzer:innen ein Produkt bedienen können.

Beim **Interaction Design** geht darum **WIE** Benutzer:innen ganz konkret mit einem Produkt interagieren.

Ein gutes Interaction Design erhöht also die Usability, ist aber nicht der einzige Faktor, der zu guter Usability führt.

- https://www.nngroup.com/articles/usability-101-introduction-to-usability/
- https://www.interaction-design.org/literature/article/what-is-interaction-design

Grafik und Farben

Das Grafikdesign schliesslich bestimmt wie die Produkte konkret gestaltet werden. Layout, Schriften, Farben etc. haben einen grossen Einfluss, ob wir etwas ansprechend oder schön finden, oder eben nicht.

Grafik hat ebenfalls einen Einfluss auf die Usability, allerdings nützt die schönste Grafik nichts, wenn das Produkt nicht gut bedienbar ist aufgrund schlechter Interaktionen.

- https://programmingdesignsystems.com
- Eine Auswahl von Programmen zu Farbpaletten
  - https://colorhunt.co
  - https://paletton.com
  - http://colormind.io