Fachhochschule Nordwestschweiz

Schnupperkurs Programmieren

Studiengang Informatik inkl. Profilierung iCompetence

Sibylle Peter, *Dozentin Software Engineering*

Prof. Dr. Barbara Scheuner, *Co-Studiengangleiterin Informatik (iCompetence)*

Information über das Studium

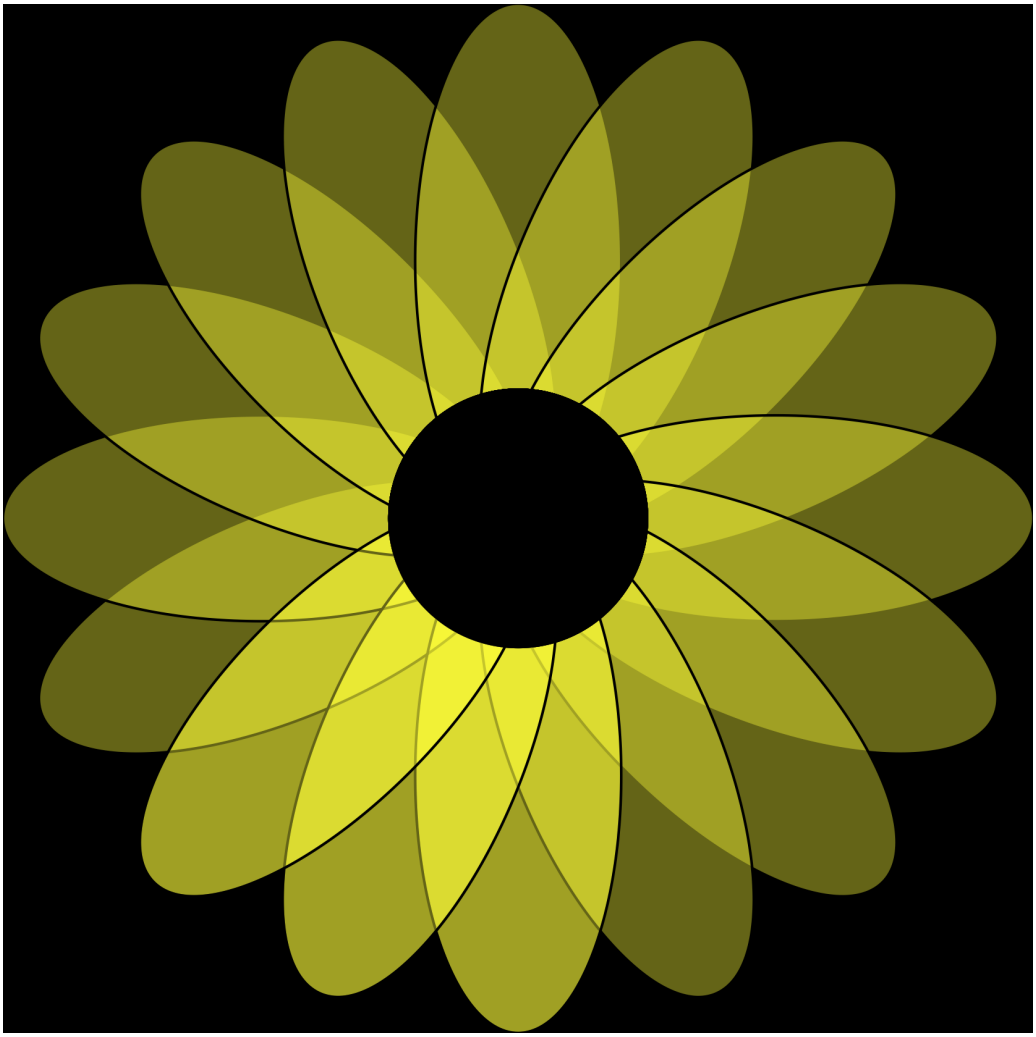
Kontaktinformationen:

- 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 Studierenden, Künstler:innen, Designern, Forscher:innen und Begeisterte 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

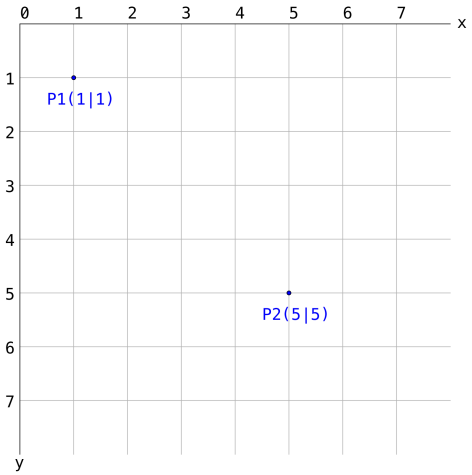
Umgebung

Processing kommt mit einem Editor und einer Laufzeitumgebung. Diese können lokal installiert werden. Damit wir sofort starten können, benutzen wir einen Webeditor. Processing wurde entwickelt, möglichst einfach grafische Elemente darstellen zu können. Am Ende dieses Workshops werden wir ein Programm haben, mit dem wir dynamisch eine Blume zeichnen.

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.

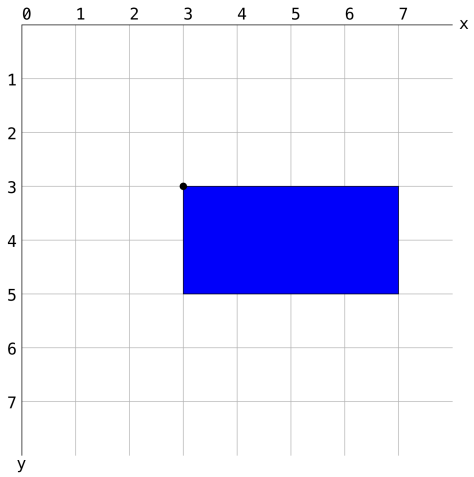


Formen

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.

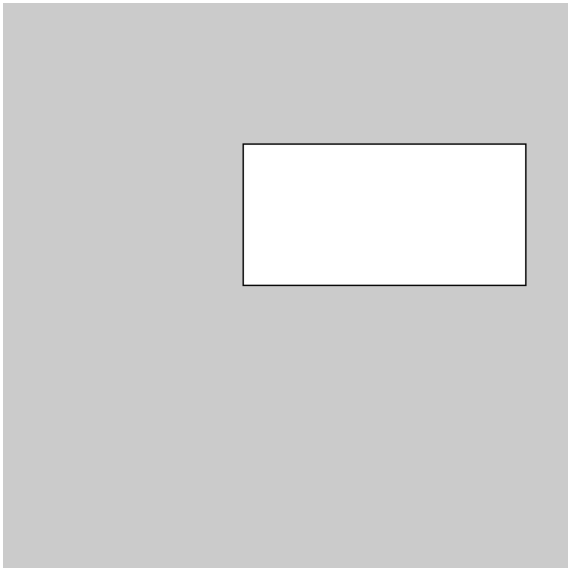
`rect(3,3,4,2)` dargestellt im (vergrösserten) Koordinatensystem: Es beginnt bei Punkt (3|3) und hat in x-Richtung eine Ausdehnung von 4 und in y-Richtung eine Ausdehnung von 2.



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);
```

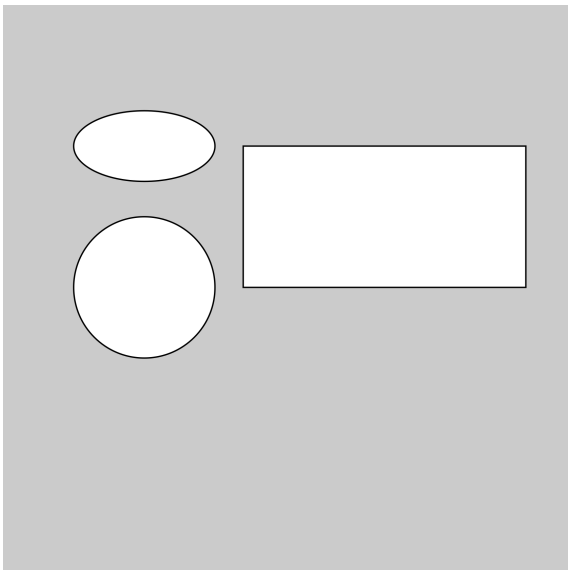


Für die Darstellung der dynamischen Blume brauchen wir allerdings Kreise und Ellipsen. Auch für diese Formen gibt es Methoden:

`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);
```



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.

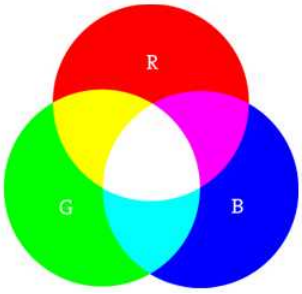
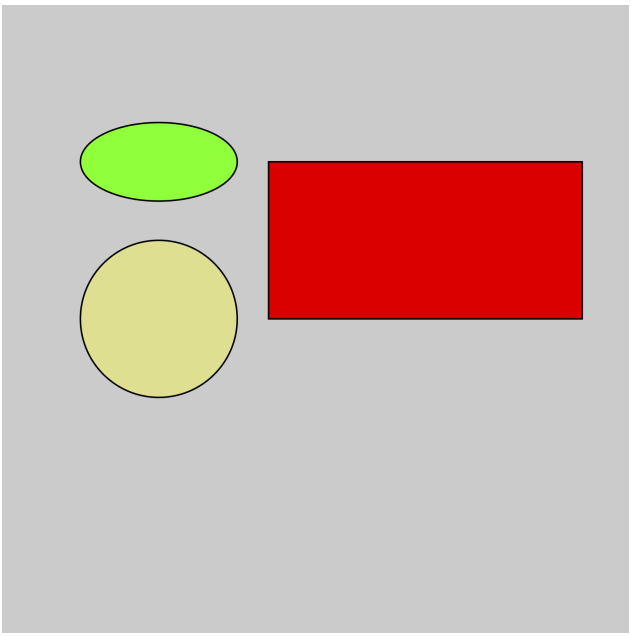


Bild von https://processing.org/tutorials/color

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. Hier gibt es eine Deckkraft von 100 - so wie wir das nachher auch für die Blütenblätter wollen.

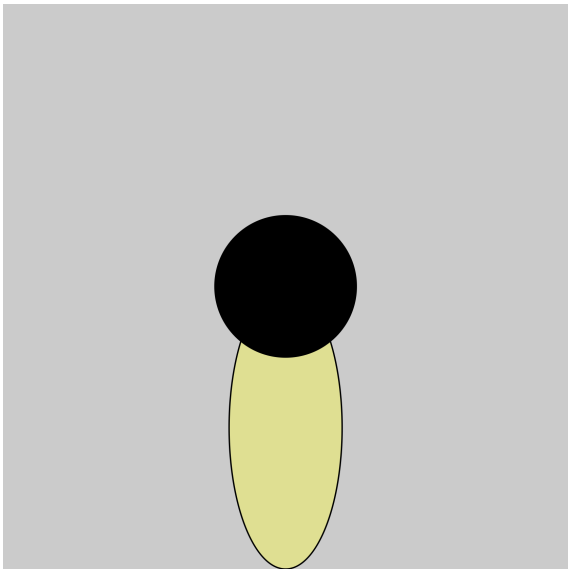


Animierte Blume

Versuchen Sie nun mit den bisherigen Informationen einen Teil der Blume wie auf dem Bild rechts zu zeichnen.

+ Hinweise:

- Die Grösse des Zeichenfelds ist nach wie vor 400x400 Pixel.
- Das gelbe Blütenblatt ist eine Ellipse mit einer Länge von 80 und einer Höhe von 200 Pixel.
- Der schwarze Kreis der Blüte hat einen Radius von 100 und liegt genau in der Mitte.
- Das gelbe Blütenblatt hat eine Deckkraft von 100.



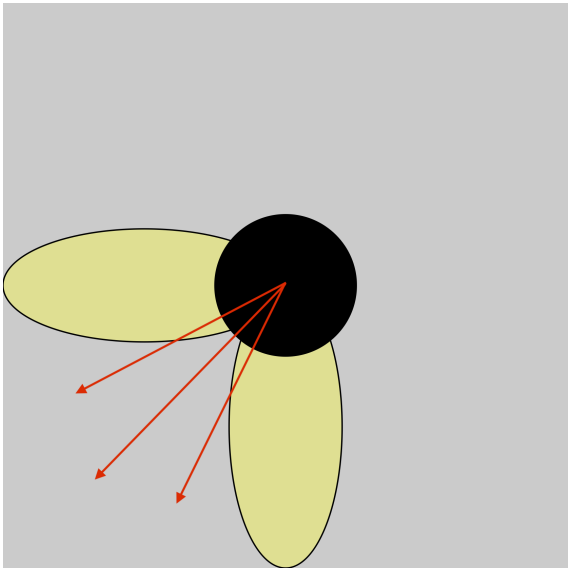
Es kommt darauf an, in welcher Reihenfolge, die Elemente gezeichnet werden.

▼ *Lösungshinweis*

```
1 size(400,400);
2 fill(255,255,0,100);
3 ellipse(200,300, 80, 200);
4 fill(0);
5 ellipse(200,200,100,100);
```

Transformationen

Damit ist der Anfang der Blume vorhanden. Es braucht allerdings noch weitere Blütenblätter. Die Blütenblätter für die anderen 3 Himmelsrichtungen wären wahrscheinlich relativ einfach zu bestimmen. Durch Hinzufügen von `ellipse(100,200, 200, 80);` entsteht z. B. das Blütenblatt nach Westen.



Aber wie zeichnen wir die Blütenblätter dazwischen, z. B. das Blatt, das nach Südsüdwest zeigt?

Rotation

Die Blütenblätter zwischen den 4 Richtungen können nicht einfach so gezeichnet werden, sondern müssen rotiert werden.

Dazu stellt uns Processing die Funktion `rotate(radians)` zur Verfügung. Wir sind uns vielleicht eher gewohnt, Winkel in Grad und nicht in Radiant anzugeben. Es gibt Umrechnungsmethoden in Processing, es ist aber einfach Radiant im Verhältnis zu PI anzugeben. Ein Radiant entspricht einem Kreis, also 360°. Das bedeutet, wir können Radianten auch mit Pi ausdrücken.

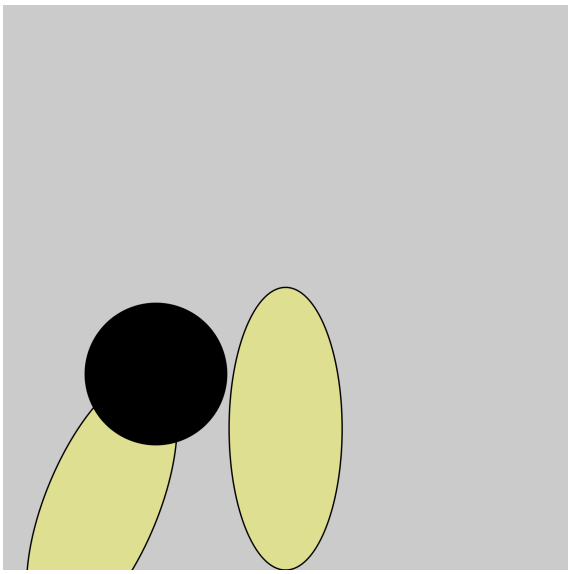
`1 rad = 2*PI`.

Für 16 Blütenblätter bedeutet das, dass sie jeweils um einen Winkel von 360°/16 rotiert werden. Ausgedrückt in Radiant ist das `2PI/16` resp. `PI/8`.

Die Methode `rotate();` wird vor dem Zeichnen des nächsten Elements aufgerufen und betrifft alles, was nachher gezeichnet wird.

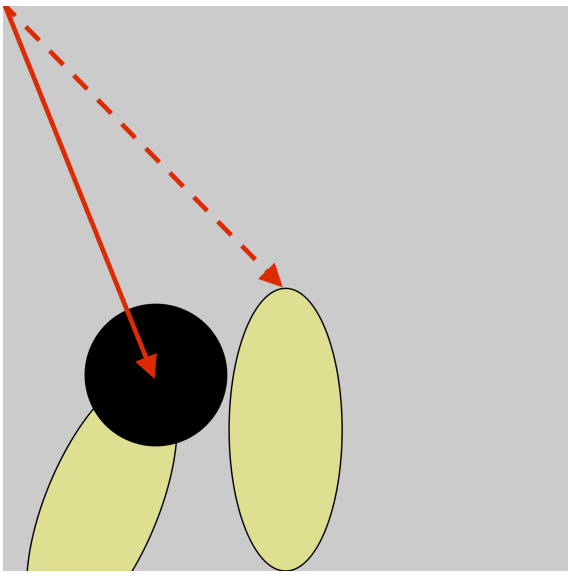
Wenn wir vor dem nächsten Blütenblatt `rotate(PI/8);` aufrufen, gibt es das folgende Resultat:

```
1 size(400,400);
2 fill(255,255,0,100);
3 ellipse(200,300, 80, 200); // Süden
4 rotate(PI/8);
5 ellipse(200,300, 80, 200);
  //SüdSüdWest
6 fill(0);
7 ellipse(200,200,100,100);
```



Das ist allerdings nicht das gewünschte Resutat.

Der Grund ist, dass in Processing nicht die Elemente rotiert werden, sondern das Koordinatensystem und zwar immer um seinen Nullpunkt.



Translation (Verschiebung)

Wenn immer um den Nullpunkt des Koordinatensystems rotiert wird, muss der Nullpunkt an den gewünschten Ort verschoben werden. Genau das macht die Methode `translate(x,y);`.

Die Verschiebung des Nullpunkts hat Auswirkungen auf die Koordinaten der Elemente, d. h. die Ellipse des Blütenblatts beginnt nun bei (0,100);

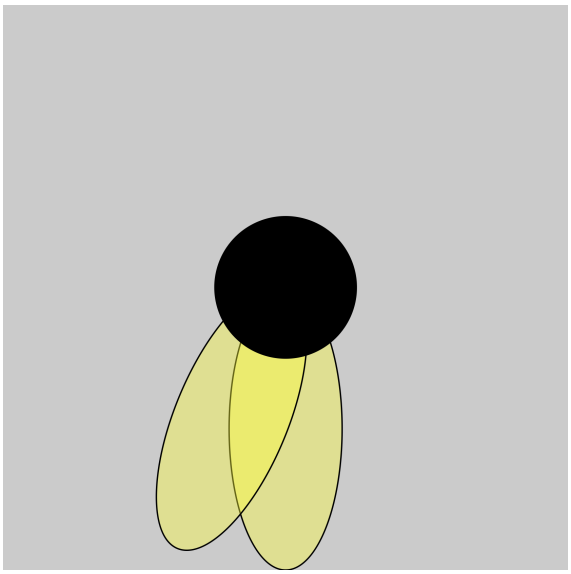
```
1 size(400,400);
2 fill(255,255,0,100);
3 translate(200,200); ①
4 ellipse(0,100,80,200); ②
5 rotate(PI/8);
6 ellipse(0,100,80,200); ③
7 fill(0);
8 ellipse(0,0,100,100); ④
```

① verschiebt

② 1. Ellipse hat neu den Mittelpunkt (0|100)

③ Nach der Rotation wird die gleiche Ellipse gezeichnet

④ Der schwarze Kreis hat neu den Mittelpunkt (0|0)



Animation

Um eine vollständige Blume zu erhalten, könnten die Zeilen 5 und 6 vom vorherigen Source Block 14 mal wiederholt werden. Auch wenn wir dazu eine Schleife benutzen um Wiederholungen programmieren, so erhalten wir keine animierte Blume.

Für interaktive Programme und Animationen werden sogenannte **Frames** verwendet. Frames werden in regelmässigen Abständen immer wieder aufgerufen. Processing kennt dafür die eingebaute Methode `draw()`. Code in dieser Methode wird in jedem Frame ausgeführt, dabei werden vorhandene Elemente überzeichnet.

Weil nicht aller Code mehrmals ausgeführt werden muss, gibt es die Methode `setup()`, die zu Beginn des Programms genau einmal ausgeführt wird.

Für unsere animierte Blume nutzen wir diese Möglichkeiten und schreiben unseren Code etwas um.

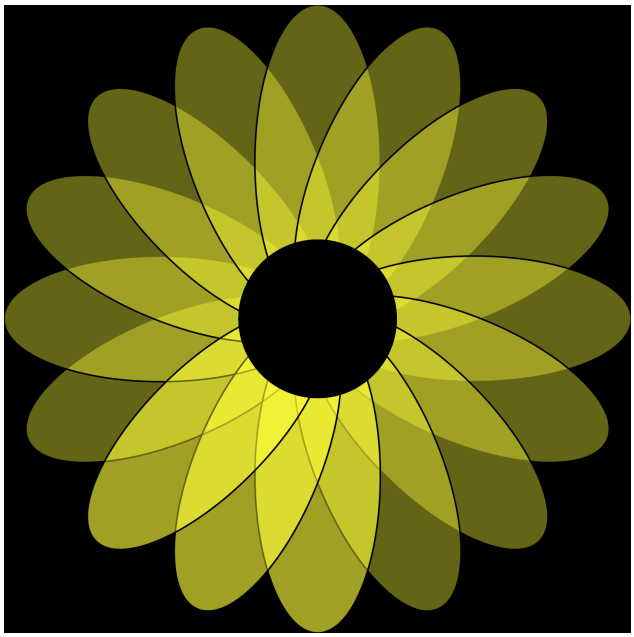
In der `setup()` Methode setzen wir die Grösse des Zeichenfensters und die Hintegrundfarbe auf Schwarz mit `background(0);`. Ausserdem verlangsamen wir die mit der Methode `frameRate(5)`.

Die Blume zeichnen wir in der Methode `draw()`. Es wird nur ein Blütenblatt gezeichnet, allerdings jedes Mal in einem anderen Winkel. Die Grösse des Winkels können wir verändern, indem wir den Basiswinkel (`PI/8`) mit einer Zählvariable `i` multiplizieren. So wird ein Blütenblatt nach dem anderen gezeichnet. Am Schluss zeichnen wir noch die schwarze Blütenmitte.

Die Animation läuft endlos, durch das mehrmalige Zeichnen der Blütenblätter wird das Gelb immer weniger transparent.

Die Variable muss ausserhalb der Methode initialisiert werden, damit das Hochzählen funktioniert.

```
1 void setup(){
2   size(400,400);
3   frameRate(5);
4   background(0);
5 }
6 int i = 0;
7 void draw(){
8   fill(255,255,0,100);
9   translate(200,200);
10  rotate(i*PI/8);
11  ellipse(0,100, 80, 200);
12  fill(0);
13  ellipse(0,0,100,100);
14  ①
15  translate(-200,-200);
16  i++;
17 }
```



① Die Rücksetzung des Koordinatensystems geschieht eigentlich automatisch. Die Web-IDE hat aber einen Fehler, darum müssen wir diese Zeile einfügen.

Selbständig ausprobieren

Die Anzahl Blütenblätter kann durch Ändern des Basiswinkels erreicht werden, z. B. `PI/4` oder `PI/16`.

Herzliche Gratulation

Das erste Programm mit Processing ist geschafft.

Ich hoffe, dieser Ausflug in die Welt des Processings hat Spass gemacht und wir können Sie im Herbst bei uns im Studium begrüssen.

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