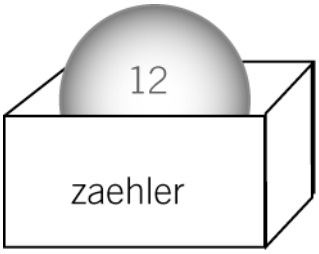
Processing – Info zu Variablen und Bedingungen

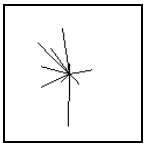
Dieses Dokument enthält

- Variablen (Variablen, Datentypen, Deklaration, Geltungsbereich, Benennung)
- Bedingungen (if, mousePressed, else)

Variablen



Eine Variable ist eine Größe, die verschiedene Werte annehmen kann. Sie ist also in ihrer Größe veränderlich oder variabel. Eine Variable kann man sich als Behälter vorstellen, der jeweils einen Wert aus einer bestimmten Wertemenge (Zahlen, Worte, etc.) aufnehmen kann. In unserem Beispiel nimmt die Variable zaehler den Wert 12 auf.
Das Arbeiten mit Variablen vereinfacht den Programmcode, und erleichtert Änderungen.



Diesen Stern würden wir ohne Variablen so zeichnen:

```
line(200,200, random(150,250), random(150,250));
line(200,200, random(150,250), random(150,250));
line(200,200, random(150,250), random(150,250));
line(200,200, random(150,250), random(150,250));
line(200,200, random(150,250), random(150,250));
```

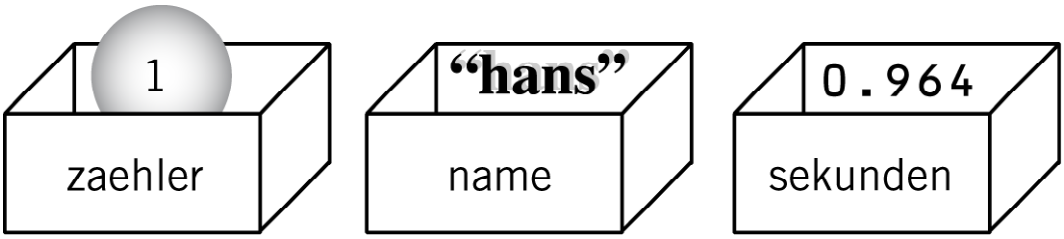
Mit den Variablen x und y ersetzen wir sich wiederholende Elemente. Das ist eine Art Abkürzung, welche die Überarbeitung des Codes flexibler macht. Um den Mittelpunkt neu zu setzen, müssen wir die Werte nur an einer Stelle ändern.

```
int x=200;
int y=200;

line(x,y, random(x-50,x+50),random(y-50,y+50));
line(x,y, random(x-50,x+50),random(y-50,y+50));
line(x,y, random(x-50,x+50),random(y-50,y+50));
line(x,y, random(x-50,x+50),random(y-50,y+50));
line(x,y, random(x-50,x+50),random(y-50,y+50));
```

Datentypen

Variablen können Werte von verschiedenen sog. Datentypen speichern:



- **`int`** Integer (ganze Zahlen, als z.B. 1, 2, 3, -17, 31008)
  Wird eine Zahl ohne Komma geschrieben, so ist sie automatisch integer
- **`String`** String (Zeichenketten, also z.B. "hans", "a", " processing ist toll ")
  Strings werden immer in Anführungszeichen geschrieben, damit sie nicht mit Variablennamen verwechselt werden.

  `name="hans";` (das Wort "hans" wird in der Variabeln name gespeichert)

- **`float`** Floating point numbers (Kommazahlen, also z.B. 3.14, 1.0, -10029.93333)

Deklaration und Verwendung von Variablen

Variablen müssen in Processing immer erst deklariert werden, bevor sie verwendet werden. Die Deklaration ist eine Art Platzreservation, sozusagen das "Anschreiben" unseres Behälters. Die Deklaration muss vor der ersten Verwendung stattfinden, und sieht beispielsweise so aus:

```
int x=0; // Deklariere eine Variable x vom Typ integer, und setze sie = 0.
int y; // Deklariere eine Variable y vom Typ integer. Diese hat vorerst noch keinen Wert.
```

`float r=random(5);` // deklariere eine Variable r. Sie wird zufällig auf einen Wert zwischen 0.0 und 5.0 gesetzt. Da die Funktion random reelle Werte annehmen kann, muss r vom Typ float sein.

`int  gr= int(random(5));` // deklariere eine Variable gr und setze sie auf einen ganzzahligen Zufallswert zwischen 0 und 4.

Nach der Deklaration kann die Variable jederzeit gesetzt oder gelesen werden.

`x=200;` // setze die Variable x auf den Wert 200

`point(x,y);` // Verwende die Werte x,y um einen Punkt zu zeichnen

`println(x);` // Gib den Wert von x im Processing Fenster aus (sehr nützlich zur Fehlersuche).

Geltungsbereich von Variablen

Je nachdem, ob eine Variable nur kurzzeitig gebraucht wird (lokal) oder ob sie generell im ganzen Programm gültig sein soll (global) spricht man von einem anderen Geltungsbereich der Variablen. Jede Variable ist genau in dem Block gültig, in dem sie deklariert wurde.

Beispiel 1: lokale Variable

```
void setup() {
  size(400,400);
  int x=200;
  int y=200;
  point(x,y)
}

void draw() {
  int x=mouseX;
  int y=mouseY;
  point(x,y);
}
```

Die Variable wird nur gerade verwendet, um einen Punkt zu zeichnen. Nach Ende der setup-Routine wird sie ungültig. In der Routine draw muss sie wieder neu deklariert werden.

Verwendungszweck: Kurzzeitiges Zwischenspeichern von Werten.

Beispiel 2: globale Variable

```
int x=100;

void setup() {
 size(200,200);
}

void draw() {
 fill(255,200,0);
 ellipse(x,200,10,10);

 x = x + 3;
}
```

In diesem Beispiel wird mit jedem Ausführen der draw-Routine die Variable x hochgezählt, und damit die Position der Ellipse neu gesetzt. Der Wert x muss nach dem Ausführen der draw() Routine erhalten bleiben, damit die Position beim nächsten Mal neu aus der alten berechnet werden kann.
Die Variable x wurde am Anfang, (d.h. ausserhalb der Klammern) deklariert, und ist somit übergreifend gültig.

Verwendungszweck: Variablen, die zwischen setup und draw, oder zwischen zweimaligem Ausführen der draw Routine ihren Wert behalten sollen.

Variablen benennen

Ein Variablenname kann aus Buchstaben, Unterstrichen und Zahlen bestehen. Er muss mit einem Buchstaben beginnen. Zulässige Namen sind also:

`Name, a5, a_5, meine_kleine_variable`

Unzulässig sind z.B.

`5_a`, oder **`hokus pokus`** (weil der Name einen Leerschlag enthält)

Processing nimmt Gross- und Kleinschreibung sehr genau:

`MeineVariable meinevariable MEINEVARIABLE mEiNeVaRiAbLe`

Sind vier verschiedene Variablen.

Benenne Variablen immer mit möglichst sinnvollen Bezeichnungen. Also z.B. **`zaehler`** statt **`i`** für eine Zählervariable, **`xpos, ypos`** für Variablen, die Punktkoordinaten bezeichnen.

Bedingungen mit if

If ist eine Konstruktion, die es erlaubt, Teile des Codes nur unter einer gegebenen Bedingung auszuführen. Ein klassisches Beispiel dafür ist die Abwandlung unseres Zeichnungsprogrammes. Es soll nur gezeichnet werden, wenn die Maus gedrückt ist.

```
if (mousePressed) {
   line(pmouseX, pmouseY, mouseX, mouseY);
 }
```

Die Bedingung in Klammern muss wahr sein, damit die Anweisungen in den geschweiften Klammern ausgeführt werden. Die Systemvariable mousePressed ist eine vordefinierte Bedingung. Im Normalfall müssen wir die Bedingungen selber formulieren.

Vergleiche

Dieses Skript bewegt eine Ellipse über den Bildschirm (hierzu wird die Variable x in jedem Durchgang erhöht).

```
 x = x + xspeed;
 if (x >= width) {
   x=0;
 }
ellipse(x,200,10,10);
```

In der zweiten Zeile wird überprüft, ob die horizontale Koordinate x den rechten Rand überschritten hat. Wenn dies erfüllt ist, wird x=0 gesetzt.

Vergleichsoperatoren

==	if (x==1)	überprüft, ob x gleich 1 ist (Achtung 2 Gleichheitszeichen)
<=	if (x<=y)	überprüft, ob x kleiner oder gleich ist als y
!=	if (x!=0)	überprüft ob x ungleich 0 ist

Kombination von Vergleichen

Mit den Zeichen
`||` (Alt-7, "oder")
`&&` (Shift-6, "und")
können Bedingungen auch mit einander verknüpft werden- Jede einzelne Bedingung muss dabei in Klammern stehen, und die gesamte Kombination muss ihrerseits umklammert sein.

```
if ((x > width) || (x < 0)) {
    xspeed = xspeed * -1;
  }
```

Dieser Code dreht die Bewegungsrichtung xspeed um, wenn die horizontale Koordinate x entweder rechts über die Zeichenfläche hinausragt (x > width), oder links (x<0).

Negation

```
 if !(mousePressed) {
...
}
```

Das Ausrufezeichen negiert die Bedingung. Der Befehl wird also nur ausgeführt, wenn die Maus nicht gedrückt wurde.

Alternativen mit else

Die Konstruktion if kann mit dem Zusatz else versehen werden. Damit wird gleichzeitig eine Alternative angeboten, die ausgeführt wird, falls die Bedingung nicht wahr ist.

```
if (mousePressed) {
   line(pmouseX, pmouseY, mouseX, mouseY);
 }
 else
 {
   point(mouseX, mouseY);
 }
```

Zeichnet Linien, falls die Maus gedrückt ist, und Punkte, falls sie nicht gedrückt ist.

Code-Referenz zu Variablen und Bedingungen

Datentypen, Variablendeklaration

`int`	Ganzzahl
`String`	Zeichenkette
`float`	Kommazahl

Systemvariablen

`width, height`	Dimension der Zeichenfläche

Strukturen

`if`	Bedingung

Vergleichsoperatoren (für if)

`<=`	kleiner oder gleich
`>=`	grösser oder gleich
`==`	gleich (! 2 Gleichheitszeichen)
`!=`	ungleich

logische Operatoren

`||`	logisches "oder"
`&&`	logisches "und"
`!`	loschies "nicht"

Maus-Interaktion

`MousePressed`	ist wahr, wenn Maus gedrückt, sonst falsch

Anweisungen

`println();`	gibt eine Meldung aus, nützlich zum Debuggen

Nimm und lies

Zu Variablen

Reas/Fry "Processing": S. 37, Data 1: Variables, S. 101, Data 2: Text

Zu Bedingungen

Reas/Fry "Processing": S. 51, Control 1: Decisions