PROCESSING

STRUKTUR UND INPUT

Created by Michael Kirsch & Beat Rossmy

INHALT

RÜCKBLICK

PROCESSING BASICS

STRUCTURE / COLOR

Setup Funktion	`void setup() {...}`
Draw Funktion	`void draw() {...}`
Größe des Sketchs	`size(w,h);`
Konsolenausgabe	`println("something");`
Hintergrundfarbe	`background(r,g,b);`
Füllfarbe	`fill(r,g,b);`
Umrissfarbe	`stroke(r,g,b);`
Zufallszahl	`random(1);`

PROCESSING BASICS

2D PRIMITIVES

Rechteck	`rect(x,y,w,h);`
Ellipse	`ellipse(x,y,w,h);`
Linie	`line(x1,y1,x2,y2);`
Punkt	`point(x,y);`

1,2,3,... – INTEGER!

- Mit **`int`** bezeichnen wir Variablen die ganzzahlige Werte enthalten.
- Addition und Subtraktion mit anderen Integern funktioniert problemlos.
- Was allerdings passiert bei Multiplikation und Division?

```
int x = 4;

println(x+2); // prints "6"
println(x-2); // prints "2"

println(x/2); // prints "2"
println(x*2); // prints "8"

println(x/3); // prints ???
```

BOOLSCHE OPERATOREN

Größer	`if (a > b) {...}`
Größer gleich	`if (a >= b) {...}`
Gleich	`if (a == b) {...}`
Kleiner gleich	`if (a <= b) {...}`
Kleiner	`if (a < b) {...}`
Ungleich	`if (a != b) {...}`

BEDINGUNGEN

- Mit **`if`** können wir auf bestimmte Situationen reagieren.
- Ein bestimmter Fall tritt nur unter der Bedingung ein, dass die Aussage in den **`( )`** wahr ist.
- Manche Aussagen können stets wahr sein.

```
int x;

void setup () {
  x = 0;
}

void draw() {
  x = x+1;
  if (x == 10) {
    x = 0;
  }
  if (true) {
    println(x);
  }
}
```

THEORIE

... SONST!

Mit mehreren **`if`** kann auch auf das Gegenteil der ursprünglichen Aussage reagiert werden.

```
if (x == 100) {
  println("x ist genau 100");
}
if (x != 100) {
  println("x ist nicht 100");
}
```

Mit **`else`** lässt sich einfacher das Gegenteil einer Aussage behandeln.

```
if (x == 100) {
  println("x ist genau 100");
}
else {
  println("x ist nicht 100");
}
```

Das ist hilfreich wenn unsere Bedingungen komplexer werden.

```
if (x == 100 && y > 50) {
  println("x ist genau 100");
}
else {
  println("x!=100 und/oder y kleinergleich 50");
}
```

WENN MEHR GELTEN SOLL!

Und (sowohl als auch)	`if (x > 50 && y < 50) {}`
Oder (das eine, das andere oder beides)	`if (x > 50 || y < 50) {}`
Nicht (das Gegenteil davon)	`if (x > 50 || !(y >= 50)) {}`
Klammern halten Aussagen zusammen!	`if ((x>5 || !(y>=5)) || (x==5 && y==5)) {}`

KLEINE SCHRITTE UND GROSSE SCHRITTE.

- Manchmal reichen ganze Zahlen nicht aus um jeden Sachverhalt entsprechend darstellen zu können.
- "Ein Apfel", "Ein Haus", ...
- "Eine halbe Stunde", "Ein halber Meter", ...

http://zak-site.com/Great-American-Novel/images/44-102/ff98-one-step.jpg

KLEINE SCHRITTE UND GROSSE SCHRITTE.

- Angenommen unser Kreis soll sich nun doppelt so schnell bewegen. Was müssen wir an unserem bisherigem Sketch ändern?
- Aber wenn er sich nur halb so schnell bewegen soll...

```
int x;

void setup () {
   size(100,100);
   x = 0;
}

void draw () {
   background(255,0,0);
   x = x+1;
   ellipse(x,50,50,50);
}
```

FLOAT

- Es gibt natürlich auch andere Datentypen als Integer.
- Mit **`float`** definieren wir Variablen, die Gleitkommazahlen enthalten.
- D.h. alle Zahlen, die bei uns üblicherweise ein Komma enthalten.
- Aber Vorsicht!

```
float x;

void setup () {
   size(100,100);
   x = 0;
}

void draw () {
   background(255,0,0);
   x = x+0.5;
   ellipse(x,50,50,50);
}
```

IMMER UND IMMER WIEDER...

- Von Zeit zu Zeit passiert es, dass man manche Sachen immer und immer wieder zu erledigen hat...
- Selbst wenn es einfache Aufgaben sind, wie "zeichne 5 zufällige Punkte" kann sich dadurch der Programmcode immens verlängern.

http://img.thesun.co.uk/multimedia/archive/01647/LOW-RES-GETTY-2694_1647748a.jpg

IMMER UND IMMER WIEDER...

- Aber wir erkennen ein Muster!
- Es entstehen Blöcke von drei Zeilen, die sich wiederholen.
- Könnte man diese Blöcke automatisch wiederholen...

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

  float x = random(width);
  float y = random(height);
  ellipse(x,y,5,5);

  x = random(width);
  y = random(height);
  ellipse(x,y,5,5);

  x = random(width);
  y = random(height);
  ellipse(x,y,5,5);

  x = random(width);
  y = random(height);
  ellipse(x,y,5,5);

  x = random(width);
  y = random(height);
  ellipse(x,y,5,5);
}
```

FOR

- **`for`** leitet unsere Schleife ein.
- In den **`()`** beschreiben wir die Wiederholungen: "Zähle von 0 bis 4, bei dem 5. Durchgang breche ab."
- **`{}`** enthalten die auszuführenden Befehle und wird Schleifenrumpf genannt.

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

  for (int n=0; n<5; n++) {
    float x = random(width);
    float y = random(height);
    ellipse(x,y,5,5);
  }
}
```

WIE KANN ICH EINFLUSS NEHMEN?

- Alles läuft so wie ich es geschrieben habe!
- Aber wie kann ich mein Programm während der Laufzeit beeinflussen?
- Wie funktioniert das mit der Maus und dem Keyboard?

http://dougengelbart.org/images/pix/img0030.jpg

WIE KANN ICH EINFLUSS NEHMEN?

- **`mouseX`** und **`mouseY`** sind Variablen, die immer die aktuelle Position der Maus enthalten!
- **`mousePressed`** ist **`true`** wenn die Linke Maustaste gedrückt ist, ansonsten **`false`**.
- Wie wir auf Tasten des Keyboards reagieren können, lernen wir in der nächsten Sitzung.

```
void setup () {
    size(800,800);
}

void draw () {
    background(0);
    ellipse(mouseX,mouseY,100,100);
}
```

PROCESSING BASICS

Einzeiliger Kommentar

```
int x = 0; // x ist 0
x = x+1; // x ist 1
```

Mehrzeiliger Kommentar

```
int x = 0;
/*
x ist 0
gleich ist x 1
*/
x = x+1;
```

Inkrementieren

```
int x = 0;
x = x+1;
x += 1;
x++; // x ist 3!
```

Dekrementieren

```
int x = 0;
x = x-1;
x -= 1;
x--; // x ist -3!
```

PROCESSING BASICS

Breite des Fensters

```
int w;

void setup () {
    size(700,350);

    w = width;
    // w ist 700
}
```

Höhe des Fensters

```
int h;

void setup () {
    size(700,350);

    h = height;
    // h ist 350
}
```

ANWENDUNG

RANDOM

```
void setup () {
    size(800,800);
}

void draw () {
    background(0);
    // zufälliger RGB-Hintergrund?
}
```

ELSE

```
int counter;

void setup () {
    size(800,800);
    // counter initialisieren
}

void draw () {
    background(0);
    // counter inkrementieren

    // counter zurücksetzen

    // Kreis oder Rechteck
    // abhängig von counter
}
```

FLOAT

```
float x1,y1,x2,y2;

void setup () {
    size(800,800);
    // Koordinaten initialisieren
}

void draw () {
    background(0);
    // x1, x2 minimal inkrementieren

    // x1 zu int konvertieren

    // wandernde Kreise zeichnen
}
```

FOR

```
void setup () {
    size(800,800);
}

void draw () {
    background(0);
    // zeichne mehrere Rechtecke mit zufälligen Parametern in Schleife
}
```

MOUSEX & MOUSEY

```
void setup () {
    size(800,800);
}

void draw () {
    // zeichne Hintergrund abhängig ob Maus gedrückt ist oder nicht

    // zeichne Kreis an der aktuellen Mausposition
}
```

VERKNÜPFUNG

DIE PROCESSING REFERENCE

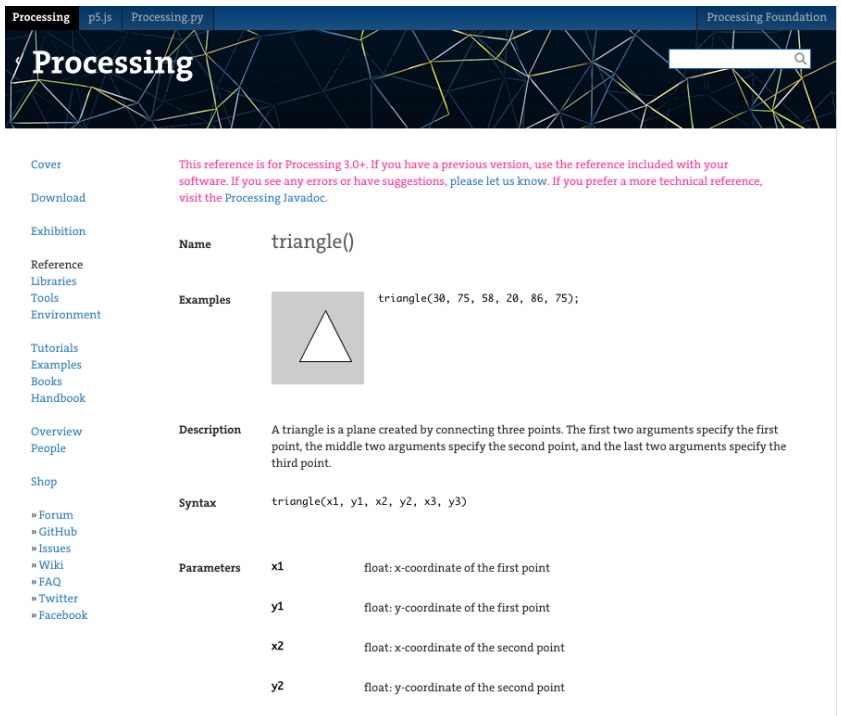
WIR KÖNNEN HIER LEIDER NICHT ALLES LERNEN...

- https://processing.org/reference/
- Inhalte sind übersichtlich nach Themen sortiert: "2D-Primitives", "Image", "Math", ...
- Wir könne auch finden was wir suchen, ohne genau zu wissen wie es heißt!

https://processing.org/reference/

DIE PROCESSING REFERENCE

- Die Dokumentation enthält immer Beispiel-code, beschreibende Texte und weiter-führenden Links zu verwandten Themen.
- Mit etwas Übung findet man sehr schnell was man sucht.
- Tipp: für Suchanfragen verwende zuerst die Processing interne Suche!



https://processing.org/reference/triangle_.html

EIN KREIS IST NICHT GENUG!

Wir lassen nun zwei Kreise sich in unterschiedliche Richtungen bewegen.

EIN KREIS IST NICHT GENUG!

- Wir deklarieren vier **`float`** Variablen für die Koordinaten der Kreise.
- Diese werden mit Werten innerhalb der Fensterdimensionen initialisiert.
- Durch inkrementieren / dekrementieren der X-Koordinaten bewegen sich die Kreise.

```
float x1,y1,x2,y2;

void setup () {
  size(500,500);
  x1 = 50;
  y1 = 50;
  x2 = 100;
  y2 = 100;
}

void draw () {
  background(0);
  x1++;
  x2--;

  ellipse(x1,y1,20,20);
  ellipse(x2,y2,20,20);

  if (x1>width) {x1 = 0;}
  if (x2<0) {x2 = width;}
}
```

BRINGEN WIR ETWAS ZUFALL INS SPIEL.

- Bei jedem neuen Start des Programms erscheinen nun die Kreise an anderen Orten innerhalb des Fensters.
- Auch die Bewegungs-geschwindigkeit der Kreise ist nun abhängig vom Zufall.

```
float x1,y1,x2,y2;

void setup () {
  size(500,500);
  x1 = random(width);
  y1 = random(height);
  x2 = random(width);
  y2 = random(height);
}

void draw () {
  background(0);
  x1 += random(10);
  x2 -= random(10);

  ellipse(x1,y1,20,20);
  ellipse(x2,y2,20,20);

  if (x1>width) {x1 = 0;}
  if (x2<0) {x2 = width;}
}
```

AUSBLICK

NÄCHSTE SITZUNG

- Ordnung in unseren Variablen
- Strukturierung unserer Befehle
- **`void`** und andere Zauberworte

ÜBUNG

...

QUELLEN

- http://zak-site.com/Great-American-Novel/images/44-102/ff98-one-step.jpg
- http://img.thesun.co.uk/multimedia/archive/01647/LOW-RES-GETTY-2694_1647748a.jpg
- http://dougengelbart.org/images/pix/img0030.jpg
- https://processing.org/reference/
- https://processing.org/reference/triangle_.html