Processing

Anleitung

Wichtige Funktionen

setup()	Die Funktion `setup()` wird zu Beginn des Programms aufgerufen. Sie eignet sich gut für Einstellungen:
	`void setup() {`
	`  size(640, 360); // setze Größe des Fensters`
	`}`

draw()	Diese Funktion wird in jeder Sekunde 60 mal ausgeführt und alles abgearbeitet, was in ihr beschrieben ist.
	Häufig entscheiden wir hier, was auf dem Bildschirm zu sehen ist:
	`void draw() {`
	`  background(51); // Hintergrund dunkel: 0=schwarz 255=weiß`
	`  background(0, 255, 0); // Hintergrund grün: background([r],[g],[b])`
	`  fill(255, 0, 0); // Alles zukünftige soll rot werden`
	`  rect(20, 20, 20, 20); // Zeichne ein kleines Viereck:`
	`  rect([x],[y],[länge],[breite])`
	`}`

Hilfreiche Funktionen

println()	Damit kann man sich Variablen anzeigen lassen.
	`void draw() {`
	`  println(mouseX);`
	`}`

size(width, height)	Setzt die Größe des Fensters (width=Breite, height=Höhe).
	`void setup() {`
	`  size(800, 600);`
	`}`

// Kommentar
 	
**/*
Kommentar
*/**
	Manchmal möchte man einfach einen Text mit in den Code schreiben, der keine Relevanz für das Programm hat. Dazu verwendet man Kommentare:
	`println(“hello”); // Hier kann man schreiben, was man will`
	`/*`
	`Häufig verwendet man Kommentare, um anderen zu erklären,`
	`was dieser Code macht.`
	`*/`

Zeichnen

`rect(x, y, width, height)`	Zeichnet ein Rechteck in das Fenster ein. x und y bestimmen die Position und widht und height die Größe des Vierecks.
`circle(x, y, radius)`	Zeichnet einen Kreis in das Fenster ein. x und y bestimmen die Position und der Radius die Größe des Kreises.

`color(r, g, b)`	Mit der Funktion color() kann man eine neue Farbe erstellen. Die Farbe besteht aus den drei Anteilen **R**ot, **G**rün und **B**lau. 0 ist der kleinste Wert und 255 der größte.
	--- 	
	`color rot = color(255, 0, 0);`
	`void draw() {`
	`  fill(rot); // alles wird Rot gezeichnet`
	`}`

`fill(color)`
`stroke(color)`
	Diese Funktionen setzen die Füllfarbe sowie die Randfarbe von Formen (zum Beispiel rect())
	--- 	
	`void draw() {`
	`  fill(rot); // setze Füllfarbe auf Rot`
	`  stroke(blau); // setze Randfarbe auf Blau`
	`  rect(10, 10, 20, 20); // Zeichne Viereck`
	`}`

`background(color)`	Diese Funktion bestimmt die Hintergrundfarbe.
	--- 	
	`void draw() {`
	`  background(51); // alles wird dunkel grau`
	`}`

`strokeWeight()`	Damit kann eingestellt werden, wie dick die Linien sind, die gezeichnet werden sollen.
	--- 	
	`void setup() {`
	`  strokeWeight(0); // keine Linien`
	`  strokeWeight(1); // dünne Linien`
	`  strokeWeight(8); // dicke Linien`
	`}`

Interaktion mit dem Benutzer

mouseX
mouseY
	Die Variablen `mouseX` und `mouseY` beinhalten die Position der Maus.
	--- 	
	`void draw() {`
	`  fill(255, 0, 0);`
	`  rect(mouseX, mouseY, 10, 10);`
	`}`
	--- 	
	Dieser Code zeichnet ein Viereck dorthin, wo sich die Maus befindet.

keyPressed()
keyReleased()
	Mit der Funktion keyPressed() kann man abfragen, ob eine Taste gedrückt wurde. Genauso kann mit der Funktion keyReleased() herausgefunden werden, ob eine Taste losgelassen wurde.
	--- 	
	`void keyPressed() {`
	`  println(key);`
	`  if (key == ‘a’) {`
	`  println(“Die Taste a wurde gedrückt”);`
	`  }`
	`}`
	
	`void keyReleased() {`
	`  println(key);`
	`  if (key == ‘a’) {`
	`  println(“Die Taste a wurde losgelassen”);`
	`  }`
	`}`

mouseClicked()	Diese Funktion wird ausgeführt, wenn der Benutzer die Maustaste drückt und wieder loslässt.
	--- 	
	`void mouseClicked() {`
	`  println(“mouse pressed :D”);`
	`}`

Programmieren

Variablen	Manchmal muss man Informationen über längere Zeit speichern. Dafür können Variablen genutzt werden:
	`color rot = color(255, 0, 0);`
	`int position = 0;`
	
	`void draw() {`
	`  fill(rot);`
	`  rect(position, position, 10, 10);`
	`  position += 5; // Position wird verändert`
	`}`
	Eine Variable kann immer nur eine bestimmte Art von Dingen speichern. Im folgenden Abschnitt werden einige Typen behandelt.

int	Integer (kurz: int) werden verwendet, um ganze Zahlen darzustellen, wie z.B. 1, 0, 5, -10, oder 42. Sie können beispielsweise für Positionen oder Farbanteile (z.B. der Rotton einer Farbe) verwendet werden. Siehe Beispiel bei Variablen.

float	Manchmal möchte man auch rationale Zahlen (das heißt Kommazahlen) verwenden. Dann sollte man float verwenden. Diese kann man auch für Positionen, Geschwindigkeiten und viele andere Dinge verwenden.
	`float speed = 10.0f; // Geschwindigkeit des Vierecks`
	`float pos = 100.f; // x-Position des Vierecks`
	
	`void setup() {`
	`  size(800, 600);`
	`}`
	
	`void draw() {`
	`  background(200); // Hintergrund hellgrau`
	`  fill(20); // Viereck schwarz`
	`  rect(pos, 100, 20, 20); // Viereck zeichnen`
	`  pos = pos + speed; // Wir bewegen das Viereck`
	`  speed = speed * 0.98; // und werden langsamer`
	`}`

boolean	Manches ist entweder wahr oder falsch. Um die Wahrheitswerte "true” oder “false” (“wahr” oder “falsch”) zu speichern, wird der Datentyp boolean verwendet. Sie werden häufig in Verbindung zu Verzweigungen genutzt (siehe folgender Abschnitt).

Verzweigungen

Verzweigungen können verwendet werden, um nur unter bestimmten Bedingungen etwas zu tun.

```
color rot = color(255, 0, 0);
int position = 0;

void draw() {
  fill(rot);
  rect(position, position, 10, 10);
  position += 5; // Position wird verändert
  if (position > 100) { // <- Verzweigung
    position = 0;
  }
}
```

In diesem Beispiel wird die Position auf 0 zurückgesetzt, wenn sie größer als 100 ist.

String / char

Wörter und Sätze lassen sich in Strings speichern.

```
String title = “Mein Spiel”;

void setup() {
  windowTitle(title);
}
```

Möchte man nur einzelne Buchstaben speichern, kann auch der Datentyp char verwendet werden:

```
char my_fav_char = ‘g’;

void setup() {
  println(“Mein Lieblingscharakter: “ +
my_fav_char);
}
```

Jetzt etwas verrücktes: Probier mal, was passiert, wenn du `my_fav_char = 97;` setzt. Was wird auf der Konsole angezeigt? Wie könnte man wohl andere Buchstaben anzeigen lassen?

Rechen-Operatoren

Mit vielen Zahlen kann man rechnen:

```
int counter = 0;

void draw() {
  counter = counter + 1;
  println(“counter: ” + counter);
  int square = counter * counter; // Das Quadrat
  println(“square: “ + square);
  float root = sqrt(counter); // Die Wurzel
  println(“root: “ + root);
```

	`}`
	
	Es gibt sehr viele Rechenoperationen. Hier sind einige:
	`+` : Addition, das heißt die Summe zweier Zahlen
	`-` : Subtraktion, die Differenz zweier Zahlen
	`*` : Multiplikation, das Produkt zweier Zahlen
	`/` : Division, der Quotient zweier Zahlen
	`%` : Modulo, der Rest bei der Division zweier Zahlen
	
	Zusätzlich kann man das Ergebnis einer Rechnung auch gleich einer neuen Variablen zuweisen. Die folgenden drei Zeilen haben alle denselben Effekt. Die Variable counter wird um 1 erhöht.
	
	`void draw() {`
	`  counter += 1;`
	`  counter = counter + 1;`
	`  counter++;`
	`}`

`Vergleichsoperatoren`	Manchmal möchte man zwei Zahlen vergleichen. Dazu können Vergleichsoperatoren verwendet werden
	
	`int counter = 0;`
	`void draw() {`
	`  counter++;`
	`  if (counter > 200) {`
	`  println(“counter ist über 200”);`
	`  }`
	`}`
	
	Es gibt viele Vergleichsoperatoren:
	`a < b` : Ist a kleiner als b
	`a <= b`: ist a kleiner oder gleich b
	`a == b`: ist a gleich b
	`a != b`: ist a ungleich b

`Weitere Quellen`	Die Processing-Reference ist ein guter Ort, um noch weitere Möglichkeiten zu erfahren. Ihr könnt sie unter dem Link https://processing.org/reference/ finden.