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()	In jeder Sekunde wird 60 mal gemacht, was wir in der `draw()` Funktion schreiben. Häufig entscheiden wir hier, was auf dem Bildschirm zu sehen ist:
	--- 	
	`void draw() {`
	`  background(51); // Hintergrund dunkel (0=schwarz 255=weiß)`
	`  fill(255, 0, 0); // Alles soll rot werden`
	`  rect(20, 20, 20, 20); // zeichne kleines Viereck`
	`}`

Hilfreiche Funktionen

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

size(width, height)	Setzt die Größe des Fensters. “width” ist die Breite und “height” die Höhe.
	--- 	
	`void setup() {`
	`  size(800, 600);`
	`}`

frameRate()	Setzt, wie oft in der Sekunde die draw() Funktion ausgeführt wird.
	--- 	
	`void setup() {`
	`  framerate(60);`
	`  println(frameRate);`
	`}`

Zeichnen

`rect(x, y, width, height)`	Zeichnet ein Rechteck in das Fenster ein. x und y bestimmen die Position und width 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”);`
	`  }`
	`}`

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. Die Position vom Typ int kann beispielsweise nur Zahlen speichern, der Typ color nur Farben. Wenn du mehr Variablentypen kennenlernen möchtest, frag uns gerne.

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.

Audio

Audio abspielen	Beispiel code Audio abzuspielen.
	--- 	
	`import processing.sound.*;`
	`SoundFile file;`
	
	`void setup() {`
	`  size(640, 360);`
	`  background(255);`
	
	`  // Lädt eine Sounddatei aus dem /data Verzeichnis des Sketches und spielt sie ab`
	`  file = new SoundFile(this, "sample.mp3");`
	`  file.play();`
	`}`

Syntax	Audio abspielen und verschiedene Parameter übergeben.
	--- 	
	`file.play()`
	`file.play(rate)`
	`file.play(rate, amp)`
	`file.play(rate, pos, amp)`
	`file.play(rate, pos, amp, add)`
	`file.play(rate, pos, amp, add, cue)`

Parameter	Die Bedeutung der einzelnen Parameter: --- • `rate(float)` Relative Wiedergabegeschwindigkeit, die verwendet werden soll. 1 ist die Originalgeschwindigkeit. 0,5 ist die halbe Geschwindigkeit und eine Oktave tiefer. 2 ist die doppelte Geschwindigkeit und eine Oktave höher. • `pos(float)` die Panoramaposition dieser Soundeinheit von -1.0 (links) bis 1.0 (rechts). Funktioniert nur bei Mono-Soundfiles! • `amp(float)` die gewünschte Wiedergabeamplitude des Audiosamples als Wert von 0.0 (völlige Stille) bis 1.0 (volle Lautstärke)

	• `add(float)` Offset der Ausgabe des Generators um den angegebenen Wert • `cue(float)` Position im Audiosample, an der die Wiedergabe beginnen soll, in Sekunden.

`Amplitude`	`Abruf der Amplitude des Audios`
	--- 	
	float amplitude = file.amplitude();

`Stichprobenrate (sample rate)`	Die sampleRate()-Methode gibt die Samplerate der Audiodatei zurück, die die Anzahl der Samples pro Sekunde darstellt. Zum Beispiel: --- float sampleRate = file.sampleRate();
Dauer	Dauer: Die Methode duration() gibt die Dauer der Audiodatei in Sekunden zurück. Zum Beispiel: --- float duration = file.duration();
Anzahl der Kanäle	int numChannels = file.channels();
Spitzenwert Amplitude	float peakAmplitude = file.peak();
Aktuelle Zeit des abgespielten Audios	float currentTime = file.currentTime();