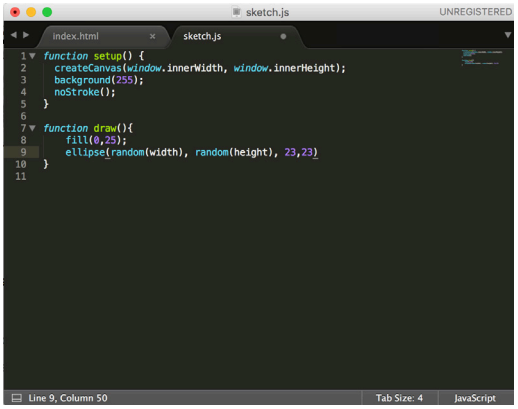
p5.js

Animiertes WEB - P5.js basics für fexibles Design

Was ist P5.js?

P5.js ist eine open source Programmiersprache für die Programmierung von Grafik, Animation und Sound im Browser. Es wurde speziell für Studenten, Künstler und Designer entwickelt und basiert auf Javascript, was Vielseitigkeit und Plattformunabhängigkeit garantiert.
P5.js wurde von Lauren McCarthy lanciert und bietet eine Alternative zu kommerzieller Software und ist eine Moderne-re Version von Processing.

Durch den einfachen Aufbau ist P5.js sehr gut geeignet, um die Grundstrukturen des Programmierens zu lernen.



Grundaufbau P5.js

P5.js Oberfläche Besteht aus einer HTML Datei und einem Javascript, dass mit jedem Editor modifiziert werden kann. Die „Applikation“ ist direkt im Browser über die HTML Datei anzuschauen.

Dateiformate

Die Programme sind als Javascript (Dateien mit der Endung .js) abgelegt. Die Javascriptdateien werden in einem Browser über einen sogenannten source Befehl in die HTML Datei verknüpft (src="../p5.min.js“) und kann dann über die HTML Datei in den meisten gängigen Browsern betrachtet werden.

Beispiele und Hilfe:
Unter „https://p5js.org/examples/“ können in P5.js vordefinierte Programme zu den meisten Standardfunktionen geladen werden.
Eine Ausführliche Programmdokumentation gibt es unter
https://p5js.org/reference/
oder auch P5.js learn
https://p5js.org/learn/

Download & Links

https://p5js.org/download/
Hier gibts die Bibliotheken zum Download - zum start nutze die Dateien im Ordner empty-example. Das HTML file zum anschauen und das sketch.js um deinen P5.js code einzufügen.

p5.js

Animiertes WEB - P5.js basics für fexibles Design

Aufbau einer *.HTML-Datei für P5.js

```
<html>
</html>
Beschreibt den Anfang und das Ende der Datei

<head>
</head>
Beschreibt den sogenannten Headder der HTML Datei.
Hier werden auch die p5.js Dateien (Sketch.js und Bi-
bliotheken) geladen.

<body>
</body>
Beschreibt den Kern der HTML Seite und ist bei einem
einfacen Test mit p5.js leer.

<script src="../p5.min.js"></script>
Dieser Befehl fügt im Headder die P5.js Bibliothek
hinzu. Dies ist die Grundvoraussetzung, dass P5.js
Befehle in HTML ausgeführt werden können.

<script src="sketch.js"></script>
Dieser Befehl fügt im Headder euer individuelles
P5.js Skript hinzu.
```

Aufbau eines P5.js-Sketches

```
function setup() {
  createCanvas(800,800);
  background(255);
}

function draw(){
     line(pmouseX,pmouseY, mouseX,mouseY)
}
```

Die Funktion setup() wird nur einmal aufgerufen, und zwar beim Start oder Reload des Browsers. Hier werden Parameter wie Fenstergrösse, Hintergrundfarbe und Zeichenstile festgelegt, aber es können auch Funktionen aufgerufen und Variablen definiert werden.

Die Funktion draw() wird in jedem Frame einmal aufgerufen, die Framerate wird in fps (frames per second) angegeben (http://de.wikipedia.org/wiki/Bildwiederholfrequenz). In dieser Phase entsteht Bewegung und Interaktion, die ganze Rechnerei, Sensorabfragen und dergleichen geschieht hier.

Die folgenden Codezeilen haben den selben Effekt, es wird ein Kreis mit Durchmesser 50 am Punkt 100/100 (von oben

Links 100 Pixel nach rechts und 100 Pixel nach unten) gezeichnet.
Und doch unterscheiden sie sich:

```
function setup(){
  ellipse(100, 100, 50, 50);
}
```

>> der Kreis wird zu Beginn des Programms ein Mal gezeichnet. wird ein anderes Objekt an die gleiche Stelle gesetzt, überdeckt es den Kreis.

```
function draw() {
  ellipse(100, 100, 50, 50);
}
```

>> der Kreis wird in jedem Frame ein Mal gezeichnet. Wird ein anderes Objekt an die gleiche Stelle gesetzt, wird er immer wieder darüber gesetzt.

Das nächste Beispiel soll den Effekt illustrieren (mit „mouseX“ und „mouseY“ werden die Koordinaten des Mouszeigers angesprochen:

```
function setup(){
  ellipse(mouseX, mouseY, 50, 50);
}
```

>> Ein weisser Kreis wird an die Mausposition gezeichnet. Da diese aber im ersten Frame noch nicht definiert ist, wird der Kreis oben links (Koordinaten 0, 0) gezeichnet, und weil dies nur einmal geschieht, passiert sonst nichts.

```
function draw() {
 ellipse(mouseX, mouseY, 50, 50);
}
```

>> In diesem Beispiel wird auch im ersten Frame der Kreis an die Stelle (0, 0) gezeichnet. Weil aber die Mausposition ständig aktualisiert wird, kann man mit der Maus nun Kreise zeichnen, in dem man über die Bildfläche fährt.

mouseX und mouseY sind Systemvariablen, also Begriffe, welche in P5.js eine klare Aufgabe haben. Als nächstes folgt ein kurzer Überblick über die wichtigsten Begriffe, die wir in diesem Kurs verwenden werden:

Code-Referenz

Ausführliche Refernz ist unter https://p5js.org/ zu finden

Spezielle Zeichen:

p5.js

Animiertes WEB - P5.js basics für fexibles Design

`//` Kommentare

`/* */` Mehrzeiliger Kommentar

`;` schliesst eine Code-Zeile (statement) ab, muss am Ende jeder Zeile stehen

`{}` Begrenzung eines Anweisungsblocks, z.B. einer Prozedur

`[]` Kennzeichnung und Zugriff auf Arrays/Listen (Klammer wird zur Indexierung von einelnen Elementen aus Arrays/Listen genutzt)

Initialisierung:

`createCanvas(w,h);`
Bühnengrösse, w: Breite, h: Höhe

`background(r,g,b);`
Hintergrundfarbe in rot, grün, blau -Werten
Werten im Bereich 0-255

`smooth();`
alle Formen werden anti-aliased dargestellt

Zeichnen:

`point(x,y);`
färbt einen Pixel an Position x, y

`line(x1,y1,x2,y2);`
Linie von Punkt x1,y1 zu Punkt x2,y2

`rect(x, y, w, h);`
Rechteck an der Stelle x,y, mit Dimensionen (Breite Höhe) w, h

`ellipse(x, y, w, h);`
Ellipse an Stelle x,y mit Dimensionen w, h

`ellipseMode(CENTER);`
x und y der Ellipse sind von der Mitte aus definiert

`triangle(x1,y1, x2,y2, x3,y3);`
Dreieck zwischen Punkten x1,y1 / x2,y2 / x3,y3

Eigenschaften:

`stroke(r,g,b);`
Linienfarbe in rgb / Graustufen mit oder ohne alpha (Transparenz)

`noStroke();`
Formen sollen ohne Linien gezeichnet werden

`fill(r,g,b);`
Füllfarbe in rgb / Graustufen mit oder ohne alpha

`noFill();`
Formen sollen ohne Füllung gezeichnet werden

`strokeWeight(s);`
Liniendicke entspricht der Zahl „s“ pixel

Maus-Interaktion:

`mouseX, mouseY`
Variablen, welche die Maus-Position enthalten

`pmouseX, pmouseY`
Variablen, welche die letzte Mausposition enthalten

`mouseIsPressed`
: boolsche Variable, true, wenn ein Mausbutton gedrückt, andernfalls false

`mousePressed()`
: Funktion, welche bei jedem Drücken des Buttons einmal ausgelöst wird

`mouseReleased()`
: Funktion, welche bei jedem Loslassen des Buttons einmal ausgelöst wird

`mouseButton`
: Eine String-Variable, die bei jedem Mausklick einen der Werte LEFT, RIGHT oder CENTER annimmt, je nachdem welcher Button gedrückt wird

`mouseMoved()`
: Funktion, welche jedesmal aufgerufen wird, wenn die Maus bewegt wird ohne dass ein Button gedrückt ist

`mouseDragged()`
: Funktion, welche jedesmal aufgerufen wird, wenn die Maus bei gedrücktem Button bewegt wird.

Übungen P5.js – Session 1

Formen

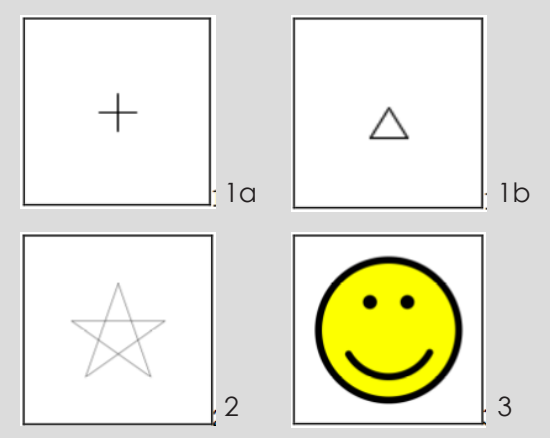
Als Input:
https://p5js.org/examples/

Zeichenbefehle

1. * Schreibe je ein P5.js-Programm, das folgende vorgegebenen Figuren zeichnet:
a. Ein Kreuz
b. Ein Dreieck (ohne die Triangle-Funktion)

2. ** Schreibe ein P5.js-Programm, das einen Stern zeichnet.

3. ** Schreibe ein P5.js-Programm, das einen Smiley zeichnet. Passe für den Mund folgende Zeile an `arc(100, 100, 80, 80, 0, PI);`

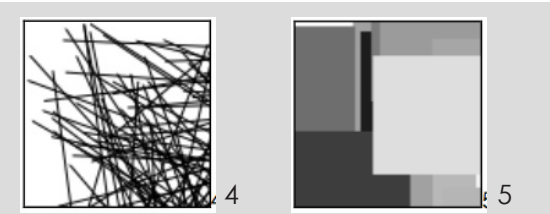


Setup, draw, random

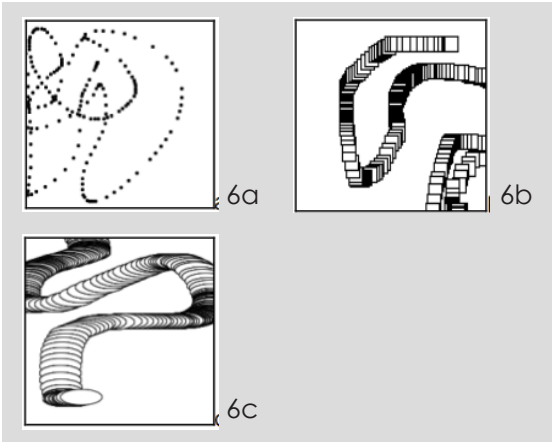
Der Ausdruck `random(x);` generiert eine Zufallszahl zwischen 0 und x, wobei die Zahl x für eine beliebige (fliesskomma-) Zahl steht (auch negativ). mit dem Ausdruck `random(x, y);` werden Ober- und Untergrenze der Zufallszahl definiert, `random(-42, 79.3);` liefert einen Wert zwischen -42 und 79.3. Die kleinere Zahl muss zuerst stehen.

4. * random: Schreibe ein Programm, das Linien mit zufällig gesetzten Anfangs- und Endpunkten zeichnet.

5. * Schreibe ein Programm, das Quadrate von 100x100



Pixel zufällig über dem Bildschirm verteilt. Der Grauwert der Quadrate soll ebenfalls zufällig zwischen 0 und 255 gesetzt werden.

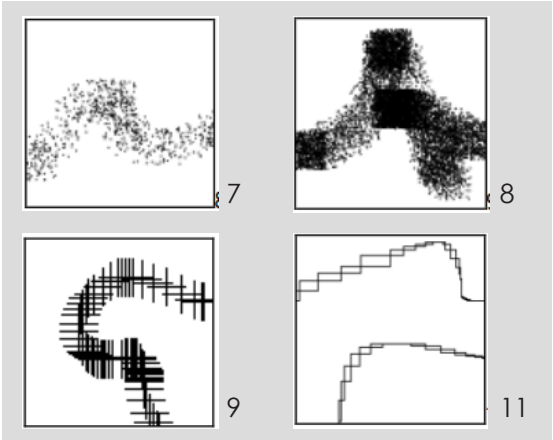


Zeichnen mit der Maus

6. * Wandle das vorgegebene Zeichenprogramm ab: Zeichne statt Linien
a. Punkte
b. Rechtecke
c. Ellipsen

```
function setup() {
}

function draw() {
  line(pmouseX, pmouseY, mouseX, mouseY);
}
```



Zeichnen mit Berechnungen

7. ** Schreibe ein Zeichnungsprogramm, das zufällige Pixel um die Mausposition zeichnet (in einem Abstand von 10 Pixeln)

8. ** Wandle das Zeichenprogramm so ab, dass die Streuung dichter wird (aber immer noch in einem Abstand von 10 Pixeln erfolgt.

9. ** Schreibe ein Zeichenprogramm mit einem Kreuz als Pinsel.

Form zeichnen

10. * Erfinde dein eigenes Zeichenprogramm

Extra

11. *** Wandle das Rechteckprogramm so ab, dass die Breite jedes Rechtecks aus der Differenz der alten und der neuen Mausposition berechnet werden.

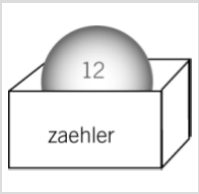
p5.js

Animiertes WEB - P5.js basics für fexibles Design

Variablen, Schleifen, Bedingungen

In P5.js werden wie in allen Programmiersprachen gewisse Strukturelemente zur Verfügung gestellt. Der geschickte Umgang mit diesen Elementen bringt die Vorteile der abstrakten, codebasierten Gestaltung zur Geltung (wie der bereits bekannte random-Begriff).

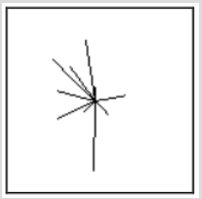
Variablen



Eine Variable ist wie in der Mathematik ein Element, welches verschiedene Werte annehmen kann. Sie ist in ihrer Größe veränderlich, also variabel.
Eine Variable kann man sich als Behälter vorstellen, der eine eindeutige Bezeichnung hat und jeweils einen Wert aus einer bestimmten Wertemenge (Zahlen, Worte, etc.) aufnehmen kann. In diesem Beispiel nimmt die Variable „zaehler" den Wert 12 auf.

Das Arbeiten mit Variablen vereinfacht den Programmcode, und macht es wesentlich einfacher, ihn nachträglich abzuändern.

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));
```

Wir definieren nun die Variablen x und y und ersetzen mit ihnen Elemente, die sich wiederholen. Das ist eine Art Abkürzung, welche die Überarbeitung des Codes flexibler macht. Um den Stern an einen anderen Punkt zu setzen, müssen wir die Werte nur noch an einer Stelle (bei der Variablendeklaration) ändern und nicht mehr in jeder Zeile.

```
var x=200;
var 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));
```

Um den Stern auch in seiner Grösse einfach verändern zu können, wird noch eine Variable `max` für die maximale Länge eingesetzt

```
var x=200;
var y=200;
var max=50;

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

Durch diesen Aufbau kann an dem Stern nun über drei Variablen jeder Wert verändert werden.

Datentypen

Variablen können Werte von verschiedenen Datentypen speichern, welche unterschiedliche Eigenschaften aufweisen. Javascript selbst unterscheidet jedoch keine Datentypen, daher müssen wir selbst aufpassen wie wir die erzeugte Variable verwenden.

Deklaration und Verwendung von Variablen

Variablen müssen in p5.js immer erst deklariert werden, bevor sie verwendet werden können. Die Deklaration ist eine Art Platzreservation, sozusagen das Bereitstellen unseres Behälters. Die Deklaration muss vor der ersten Verwendung stattfinden, und sieht beispielsweise so aus:

```
var x=0;
```
Deklariere eine Variable x, und setze sie = 0.

```
var y;
```
Deklariere eine Variable y. Diese hat vorerst noch keinen Wert.

```
var r=random(5);
```
Deklariere eine Variable r. Sie wird zufällig auf einen Wert zwischen 0.0 und 5.0 gesetzt.

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

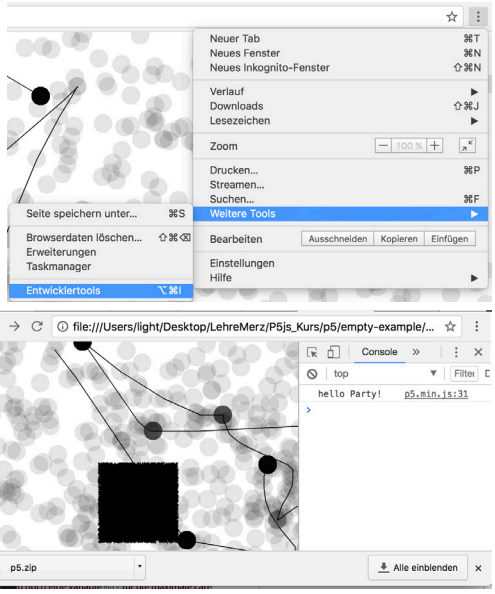
```
x=200;
```
Setze die Variable x auf den Wert 200

p5.js

Animiertes WEB - P5.js basics für fexibles Design

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

`print(x);`
Gib den Wert von x in der Console eures Web Browsers 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 Geltungs-bereich der Variablen. Jede Variable ist genau in dem Block gültig, in dem sie deklariert wurde.

Beispiel 1: lokale Variable

```
function setup() {
  createCanvas(400,400);
  var x=200;
  var y=200;
  point(x,y);
}
function draw() {
  var x=mouseX;
  var 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

```
var x=100;

function setup() {
 createCanvas(200,200);
}
function 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 und muss mit einem Buchstaben beginnen.
Zulässige Namen sind also:
`Name, a5, a_5, meine_kleine_variable`

Unzulässig sind z.B.
`5_a` (kein Buchstabe am Anfang), `hokus pokus` (Name enthält einen Leerschlag), `faktor1.4` (punkt im Namen), `länge` (Umlaut).
P5.js nimmt auch Gross- und Kleinschreibung sehr genau: `MeineVariable`, `meinevariable`, `MEINEVARIABLE` und `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 usw. andererseits sind auch zu genaue Bezeichnungen wie `linke_obere_ecke_vom_zweiten_roten_quadrat` nicht sinvoll, da empfehlen sich Abkürzungen wie z.B. `q2_rot_ol`

Schleifen (Loops)

Schleifen sind ein Mittel um repetitive bzw. iterative Vorgänge abgekürzt zu schreiben. Unser Linienstern braucht in der ausführlichen Fassung zehn Mal dieselbe Zeile:

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

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

Mit einer Schleife lässt sich dies nun zusammenfassen. In dieser Schreibweise wird i als Zählervariable verwendet, um alle Anweisungen zwischen den geschweiften Klammern {} zehn Mal zu wiederholen:

```
for (var i=1 ; i<=10 ; i=i+1 ) {
 line(x,y, random(x-50,x+50),random(y-50,y+50));
}
```

Dabei ist

`var i`

die **Deklaration**: die Lauf-/Zählervariable wird normal deklariert und ist innerhalb der geschweiften Klammern {} gültig

`i<=10`

die **Abbruchbedingung**: Die Anweisungen werden so lange wiederholt, wie die Bedingung wahr ist (d.h. solange der Zähler i kleiner oder gleich 10 ist).

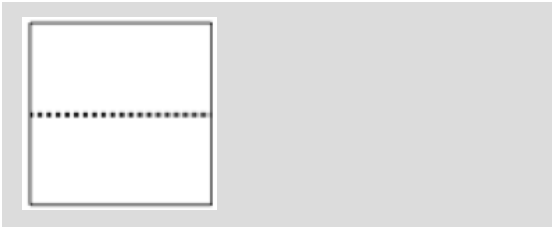
`i=i+1`

die **Inkrementierung**: i wird in jedem Durchgang um 1 erhöht.

Rechnen mit der Zählervariable

Schleifen können aber mehr als nur wiederholen: sie können wie andere Variablen auch zum Rechnen verwendet werden. Damit kann in jedem Durchgang der Schleife ein Wert verändert werden, z.B. die Position eines Elements:

```
for(var i=0; i<=20; i=i+1) {  // zähle von 1 bis 20
  point(10*i, 100);  // zeichne je einen Punkt
}
```

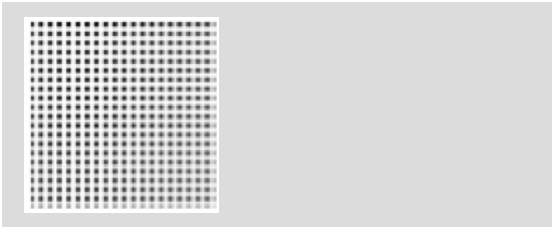


Die Position des Punktes beträgt 0,10, 20 etc. also für jedes i einen anderen Wert.

Verschachtelte Schleifen

Man kann Schleifen auch hierarchisch verschachteln. Ein häufiges Beispiel dafür ist die Verteilung von Elementen in einem horizontalen und vertikalen Raster.

```
for (var x=0; x<=20; x=x+1) {
//Schleife 1, Zählervariable x, Zeile
  for(var y=0; y<=20; y=y+1) {
  //Schleife 2, Zählervariable y, Spalte
    point(10*x, 10*y);  // je ein Punkt
  }
}
```



p5.js

Animiertes WEB - P5.js basics für fexibles Design

Bedingungen mit if

If ist eine Konstruktion, die es erlaubt, Teile des Codes nur unter einer gegebenen Bedingung auszuführen. (Nur wenn der Ausdruck gleich „true“ oder 1 ist wird die jeweilige Bedingung ausgeführt). Ein klassisches Beispiel dafür ist die Abwandlung unseres Zeichnungsprogrammes. Es soll nur gezeichnet werden, wenn die Maus gedrückt ist.

```
if (mouseIsPressed) {
   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 variable vom Typ „boolean“ welche beim Drücken der Maustaste den Wert true annimmt. Im Normalfall müssen wir die Bedingungen selber formulieren.

Vergleiche

Dieses Skript bewegt eine Ellipse von links nach rechts über den Bildschirm (hierzu wird die Variable x in jedem Durchgang erhöht). (bouncing_ball_x2 script)

```
 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 diese Bedingung erfüllt ist, wird x=0 gesetzt und die Ellipse wird wieder an den linken Rand gesetzt.

Vergleichsoperatoren

==

`if (x==1)` überprüft, ob x gleich 1 ist (Achtung 2 Gleichheitszeichen, einzelne werden für Zuweisungen wie `int x=1` benutzt)

<=

`if (x<=y)` überprüft, ob x kleiner oder gleich y ist

!=

`if (x!=0)` überprüft ob x ungleich 0 ist

Kombination von Vergleichen

Mit den Zeichen

|| (Shift-7, logisches „oder“)
&& (Shift-6, logisches „und“)

können Bedingungen auch mit einander verknüpft werden- Jede einzelne Bedingung muss dabei in Klammern stehen und die gesamte Kombination muss ihrerseits umklammert sein. (bouncing_ball_x2 script)

```
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 (!mouseIsPressed) {
...
}
```

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

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

p5.js

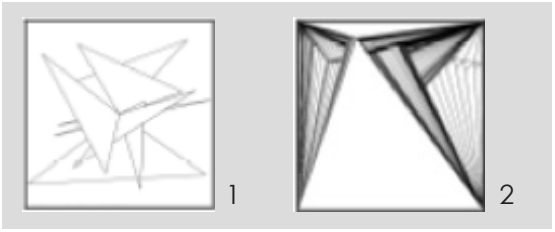
Animiertes WEB - P5.js basics für fexibles Design

Übungen P5.js – Session 2

Variablen

1. * Zeichne eine (statische) Figur aus zehn Dreiecken. Die Spitzen sollen alle in der Mitte der Zeichenfläche liegen. Die anderen Ecken sollen zufällig verteilt sein. Verwende Variablen für die Koordinaten des Mittelpunktes.

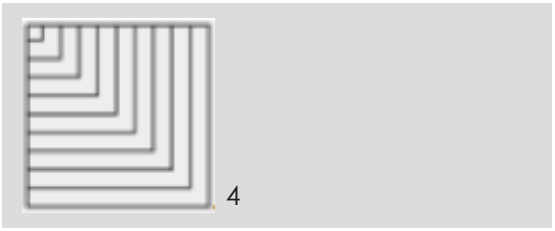
2. ** Schreibe ein Programm, welches jeweils zwei Dreiecke zeichnet. Die Spitzen der Dreiecke sollen auf der Mausposition liegen. Verwende dazu Beispiel 1, und ersetze die Variablen durch die Mauswerte. Die anderen Ecken sollen an den Bühnen-Ecken jeweils links-oben, rechts-oben, links-unten und rechts-unten liegen. Das ergibt eine Art Zeichnungsprogramm.



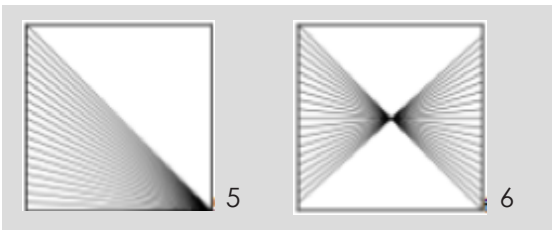
Schleifen (Loops)

3. * vereinfache Aufgabe 1 mithilfe einer for-Schleife.

4. ** Zeichne die vorgegebene Form aus 10 Quadraten, welche die linke obere Ecke gemeinsam haben und unterschiedliche Seitenlängen von 10 bis 100 aufweisen

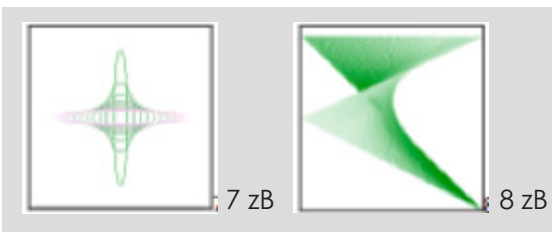


5./6. * Schreibe einen Sketch, welcher diese Form zeichnet:



7. ** Erzeuge eine eigene Komposition mit 10 Ellipsen. Die Anordnung soll nach Deinen Angaben aus der Zählervariablen berechnet werden.

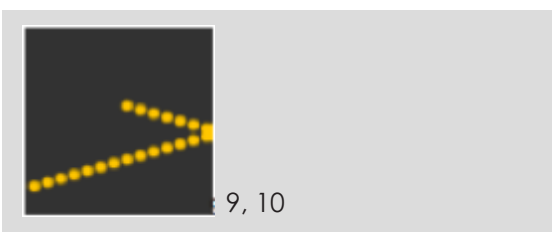
8. ** Erzeuge eine eigene Komposition mit 100 Linien. Die Anordnung soll nach deinen Angaben aus der Zählervariablen berechnet werden.



Bedingungen und globale Variablen

9. ** Wandle das Beispiel bouncing_ball_x2 (In der Sektion Bedingungen findet ihr die Kernelemente des bouncing_ball_x2 script) so ab, dass der Ball eine horizontale und eine vertikale Geschwindigkeitskomponente hat, und an allen vier Rändern abprallen kann.

10. ** Der Ball soll zu Anfang eine zufällige Geschwindigkeit haben.



p5.js

Animiertes WEB - P5.js basics für fexibles Design

11. * Verwende eine globale Variable „helligkeit“, um ein schwarzes Quadrat langsam heller werder zu lassen.

12. ** erweitere das Beispiel so, dass das Quadrat von schwarz zu weiss auf- und abblendet, dh. wieder dunkler wird, sobald es ganz hell war.

13. ** erweitere das Beispiel so, dass gleichzeitig der Hintergrund entgegengesetzt von weiss zu schwarz und wieder zurück blendet.

14. ** erweitere das Beispiel so, dass sich das Quadrat nur dann verändert, wenn die Maus gedrückt ist.

15. ** ändere das Beispiel so ab, dass man mit dem Mausklick das Quadrat ein- und ausschalten kann.

11, 12, 13, 14, 15

16. ** Entwerfe ein Zeichenprogramm, das etwas anderes zeichnet, je nachdem, ob die Maus gedrückt ist oder nicht. (z.B. Punkte statt Linien)

17. *** Entwerfe ein Zeichnungsprogramm, das beim Drücken der Maus in der linken Bildschirmhälfte mit rot zeichnet, in der rechten mit blau.

16

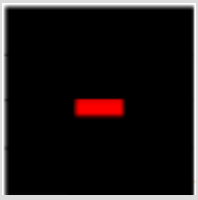
17

18. ** Entwerfe einen rechteckigen Button. Im Normalzustand ist er grau. Wenn die Maus drüberrollt wird er weiss.

19. *** Wenn die Maus drauf klickt, wird er rot (wenn die Maus daneben klickt, soll natürlich gar nichts passieren).

20. *** Wandle den Button so ab, dass er rot bleibt, wenn er einmal gedrückt wurde.

18



19,20