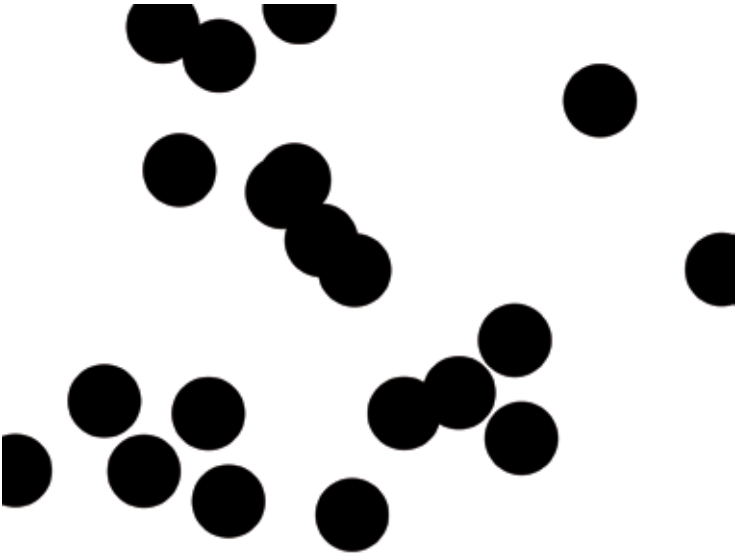


Hands-on

Adriana Mikolaskova Nautsch

Schwarze Kreisflächen werden auf einer Bildfläche verteilt. Jeder Aufruf erzeugt eine neue Konstellation. Vermutlich erkennen Sie im nachfolgenden

```
<!DOCTYPE html>
<html>

<body>

    <canvas id="bildflaeche" width="800" height="600" style="border:1px solid
#d3d3d3;">
        Dein Browser unterstützt den HTML5 Canvas-Tag nicht (erscheint als
Alternative auf dem Bildschirm</canvas>
    <script>

        var bildflaeche = document.getElementById("bildflaeche");


        var form = bildflaeche.getContext("2d");

        var formfarbe = "black";

        var anzahl = 20;

        var radius = 40;


        for (i = 1; i <= anzahl; i = i + 1) {

            var position_x = Math.floor((Math.random() * 800) + 1);
            var position_y = Math.floor((Math.random() * 800) + 1);


            //hier wird das Bildelement gezeichnet und mit der Farbe gefüllt
            form.beginPath();
            form.arc(position_x, position_y, radius, 0, 2 * Math.PI);
            form.fillStyle = formfarbe;
            form.fill();
        }

    </script>
</body>

</html>
```

Code.

Code auch ohne Erklärungen, wie Sie die Farbe der Kreise, ihren Radius sowie die Anzahl der dargestellten Elemente beeinflussen können.[1]

Das nachfolgende Kapitel brauchen Sie für Ihre eigenen Experimente mit dem oben aufgeführten Beispiel nicht zwingend notwendig. Für ein besseres Verständnis und weiterführende Experimente lohnt es sich aber, die für die meisten Programmiersprachen gültigen Begriffe und Konzepte zu kennen.

Grundlegende Begriffe und Konzepte

Ein Programm besteht aus *Anweisungen,* die der Reihe nach ausgeführt werden. Die Anweisungen verarbeiten eingegebene *Werte.* Werte können in sogenannten *Variablen* gespeichert werden. In unserem Beispiel wird die Farbe der Kreisfläche in der Variablen *farbe* abgelegt, die Anzahl der Kreisflächen in der Variablen *anzahl.*

Höhere Programmiersprachen arbeiten mit Funktionen. Den meisten Funktionen können *Werte* mitgegeben werden. Im nachfolgenden Beispiel wird der Funktion *fill* die Farbe *black* übergeben, sodass das in der Variablen *form* abgelegte Objekt, unsere Kreisfläche, mit der Farbe Rot gefüllt wird. Diese *Übergabewerte* nennt man *Parameter.*

Einige Funktionen

Mit der Funktion *fillStyle* wird die Füllfarbe für ein Objekt definiert:

```
form.fillStyle(„red“);
```

Einige Farbwerte sind in Farbbezeichnungen, also *rot, gelb, blau* etc., vordefiniert. Eine beliebige Farbe können Sie über eine Dreiergruppe von Zahlenwerten, die für je eine der additiven Grundfarben stehen, angeben: entweder hexadezimal *#ff0000* oder *rgb (255, 0, 0),* wobei im ersten Fall *hexadezimal ff* der grösste Wert, im zweiten Fall *dezimal 255* der grösste Wert ist, also viel Rot, kein Grün, kein Blau.

```
form.fillStyle="#ff0000";
form.fillStyle="rgb(255,0,0)";
```

Die Funktion *arc* zeichnet einen Bogen. Als Parameter nimmt sie die Position, den Radius, den Startwinkel und die Bogengrösse, in unserem Fall ein voller Kreis, entgegen:

```
form.arc(position_x, position_y, radius, 0, 2 * Math.PI);
```

Mit einer Kombination von mathematischen Funktionen kann ein Zufallswert[2] in einem definierten Bereich generiert und verwendet werden:

```
position_x = Math.floor((Math.random() * 800) + 1);
position_y = Math.floor((Math.random() * 600) + 1);
form.arc(position_x, position_y, radius, 0, 2 * Math.PI);
```

In unserem Beispiel wird die Zufallsfunktion für die Position der Kreise erzeugt – die Werte liegen jedoch immer im Rahmen der definierten Bildfläche, also nicht grösser als 800 bzw. 600 Pixel.

Auf die gleiche Art und Weise können Sie auch einen Wert für den Radius, die Farbe der Kreise oder der Hintergrundfarbe definieren (vgl. Codesnipplets auf der Website).

Kontrollstrukturen

Um den Programmfluss zu steuern, benutzt man sogenannte *Kontrollstrukturen,* zum Beispiel die *for-Schleife,* um bestimmte Stellen zu wiederholen. Dabei wird definiert, wie oft und mit welchen Werten die Schleife durchlaufen wird. In unserem Beispiel wird die Anzahl an Wiederholungen auf *5* festgelegt und der Anfangswert *1* in der Variablen *i* abgelegt. In jedem Durchlauf wird der Wert von i um eins erhöht, *i=i+1,* und zwar so lange, bis *i<=5* ist.

```
x=5;
for (i = 0; i < x; i++) {
form.beginPath();
form.arc(95+i*100,50,30,0,2*Math.PI);
form.stroke();
form.fillStyle = „black“;
form.fill();
}
```

Der Programmiercode kann und sollte für eine bessere Lesbarkeit mit Kommentaren versehen werden. *Kommentare* werden mit speziellen Zeichen gesetzt, was eine Ausführung verhindert – sie sind nur für den Programmierer lesbar.

```
// hier wird die Anzahl Kreise definiert, die auf der
Bildfläche dargestellt werden
var anzahl=20;
```

Kommentarzeichen können auch dazu verwendet werden, um die Ausführung von bestimmten Codestellen zu verhindern – man spricht von ‹auskommentieren›.

```
var farbe=“red”;
// var farbe=“#ff0000;
```

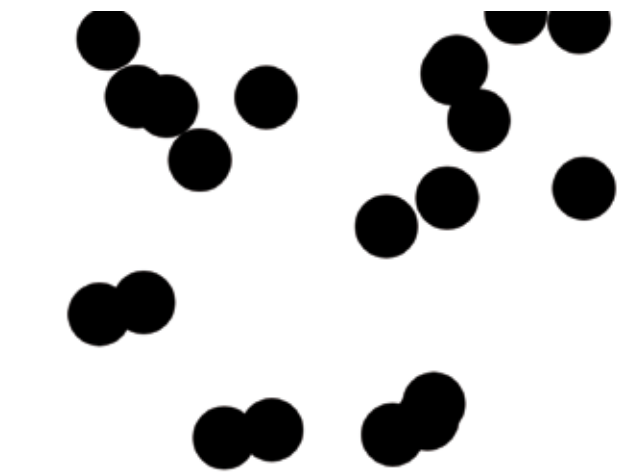
```
<!DOCTYPE html>
<html>

<body>

    <canvas id="bildflaeche" width="800" height="600" style="border:1px solid #d3d3d3;">
        Dein Browser unterstützt den HTML5 Canvas-Tag nicht (erscheint als Alternative
auf dem Bildschirm</canvas>
    <script>
        //Hier wird die Bildfläche erzeugt
        var bildflaeche = document.getElementById("bildflaeche");

        //Hier wird eine Form angelegt
        var form = bildflaeche.getContext("2d");

        var formfarbe = "black";

        var position_x = 100;
        var position_y = 100;

        var radius = 40;


        //Hier wird das Bildelement gezeichnet und mit der Farbe gefüllt
        form.beginPath();
        form.arc(position_x, position_y, radius, 0, 2 * Math.PI);
        form.fillStyle = formfarbe;
        form.fill();
    </script>
</body>

</html>
```

Ein einzelner Kreis – Code und Bild.

```
<!DOCTYPE html>
<html>
<body>
    <canvas id="bildflaeche" width="800" height="600" style="border:1px solid
#d3d3d3;">Dein Browser unterstützt den HTML5 Canvas-Tag nicht (erscheint als
Alternative auf dem Bildschirm</canvas>
    <script>
        //hier wird die Bildfläche erzeugt
        var bildflaeche = document.getElementById("bildflaeche");

//hier wird eine Form angelegt
        var form = bildflaeche.getContext("2d");

        var formfarbe = "black";

        var anzahl = 5;

        var position_x = 100;
        var position_y = 100;

        var radius = 40;

        //Folgender Code  gemäss der oben definierten Anzahl - z.b. 20x wiederholt
        for (i = 1; i <= anzahl; i = i + 1) {
            //die x-Position wird bei jedem Durchlauf um 100 Pixel verschoben
            var position_x = position_x + 100;
            //hier wird das Bildelement gezeichnet und mit der Farbe gefüllt
            form.beginPath();
            form.arc(position_x, position_y, radius, 0, 2 * Math.PI);
            form.fillStyle = formfarbe;
            form.fill();
        }
    </script>
</body>
</html>
```

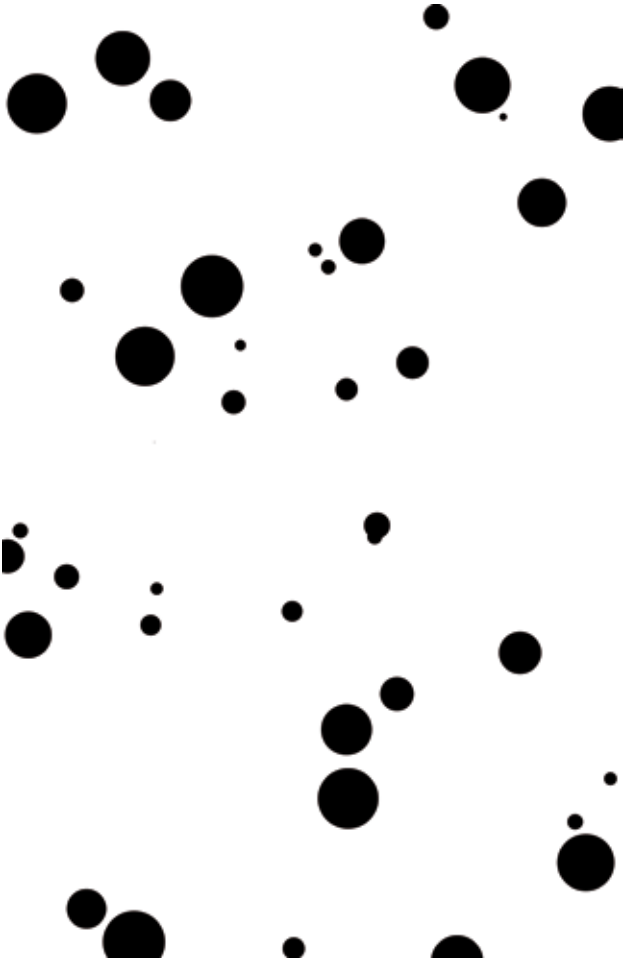
Reihe – Code und Bild.

```
<!DOCTYPE html>
<html>
<body>
    <canvas id="bildflaeche" width="800" height="600" style="border:1px solid
#d3d3d3;">
        Dein Browser unterstützt den HTML5 Canvas-Tag nicht (erscheint als Alternative
auf dem Bildschirm</canvas>
    <script>

        var bildflaeche = document.getElementById("bildflaeche");


        var form = bildflaeche.getContext("2d");

        var formfarbe = "black";

        var anzahl = 20;

        var radius = 40;


        for (i = 1; i <= anzahl; i = i + 1) {

            var position_x = Math.floor((Math.random() * 800) + 1);
            var position_y = Math.floor((Math.random() * 600) + 1);

            //hier wird das Bildelement gezeichnet und mit der Farbe gefüllt
            form.beginPath();
            form.arc(position_x, position_y, radius, 0, 2 * Math.PI);
            form.fillStyle = formfarbe;
            form.fill();
        }
    </script>
</body>
</html>
```



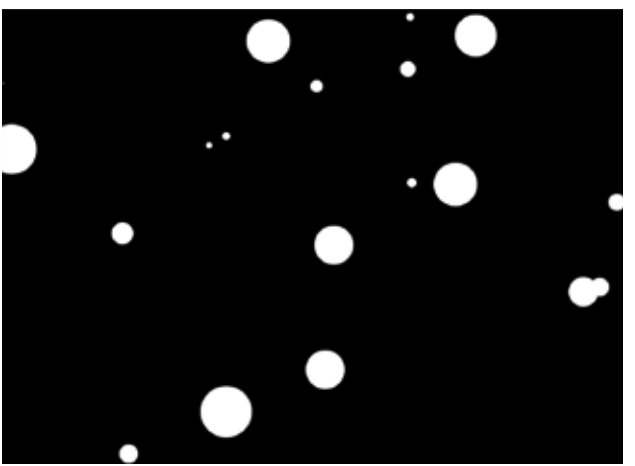
Zufällige Anordnung – Code und generiertes Bild.

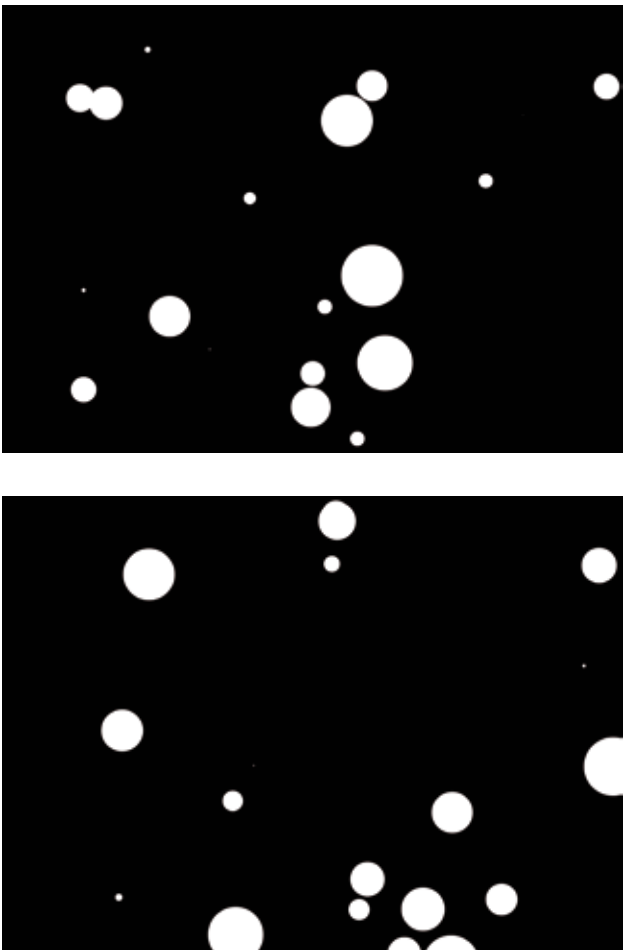
```
<!DOCTYPE html>
<html>
<body>
    <!--in dieses Element wird das weiter unten generierte Bild eingefügt-->
    <canvas id="bildflaeche" width="800" height="600" style="border:1px solid #d3d3d3;">
        Dein Browser unterstuetzt den HTML5 Canvas-Tag nicht (erscheint als Alternative
auf dem Bildschirm</canvas>
    <script>

        var bildflaeche = document.getElementById("bildflaeche");
        var form = bildflaeche.getContext("2d");

        var formfarbe = "black";

        bildflaeche.style.backgroundColor = "white";

        var anzahl = 20;

        for (i = 1; i <= anzahl; i = i + 1) {

            var position_x = Math.floor((Math.random() * 800) + 1);
            var position_y = Math.floor((Math.random() * 600) + 1);
            var radius = Math.floor((Math.random() * 60) + 1);

            form.beginPath();
            form.arc(position_x, position_y, radius, 0, 2 * Math.PI);
            form.fillStyle = formfarbe;
            form.fill();
        }

    </script>
</body>

</html>
```



Zufällige Anordnung, zufällige Grösse des Radius, Schwarz auf Weiss.

```
<!DOCTYPE html>
<html>
<body>
    <!--in dieses Element wird das weiter unten generierte Bild eingefügt-->
    <canvas id="bildflaeche" width="800" height="600" style="border:1px solid #d3d3d3;">
        Dein Browser unterstuetzt den HTML5 Canvas-Tag nicht (erscheint als Alternative
auf dem Bildschirm</canvas>
    <script>

        var bildflaeche = document.getElementById("bildflaeche");
        var form = bildflaeche.getContext("2d");

        var formfarbe = "white";

        bildflaeche.style.backgroundColor = "black";

        var anzahl = 20;

        for (i = 1; i <= anzahl; i = i + 1) {

            var position_x = Math.floor((Math.random() * 800) + 1);
            var position_y = Math.floor((Math.random() * 600) + 1);
            var radius = Math.floor((Math.random() * 40) + 1);

            form.beginPath();
            form.arc(position_x, position_y, radius, 0, 2 * Math.PI);
            form.fillStyle = formfarbe;
            form.fill();
        }

    </script>
</body>

</html>
```





Zufällige Anordnung, zufällige Grösse, Weiss auf Schwarz.

Wahl der Programmiersprache und -umgebung

Für diese Einführung wurde die Skriptsprache *Javascript* gewählt. JavaScript kann in jedem Texteditor[3] bearbeitet werden, die Interpretation erfolgt über den Browser, sodass kein weiteres Programm installiert werden muss. Diese unmittelbare Verfügbarkeit auf jedem Computer war für diese minimale Programmieranleitung ausschlaggebend.

Für weiterführende Experimente sei auf eine Online-Referenz zu JavaScript[4] verwiesen oder auf eine der zahlreichen Programmierumgebungen und Sprachen, die für gestalterische Experimente, für Nichtprogrammierer entwickelt wurden. Die bekannteste Programmierumgebung und Sprache ist *Processing* (https://processing.org). Daneben existieren aber zahlreiche weitere, die sich je nach Anwendungsgebiet und Altersstufe weit besser eignen.

Scratch (https://scratch.mit.edu/ oder eine Online-Version http://snap.berkeley.edu/snapsource/snap.html) arbeitet mit visuellen Elementen, die miteinander zu einer Programmstruktur verbunden werden, in *Nodebox* (www.nodebox.net), die Umgebung, die Ricardo Lafuente (vgl. Artikel «Coding pictures») einsetzt, wird mit *Python* programmiert..

Selbstverständlich können aber auch allgemeine Programmier- und Skriptsprachen *(Java, Python, C++ usw.)* für bildnerische Experimente verwendet werden.

Anmerkungen

1 Unter http://mikolaskova.cz/nop/handson finden Sie die Skripte sowie genauere Instruktionen für eigene Experimente.

2 *Math.random()* liefert eine Pseudo-Zufallszahl, die größer gleich 0 und kleiner als 1 ist. Ein Pseudo-Zufall deshalb, weil der Wert theoretisch berechenbar ist. Für echten Zufall greifen Funktionen auf einen

tatsächlich zufälligen Wert aus der physischen Welt zurück (Temperatur, Rauschen elektronischer Bauteile usw.), was relativ aufwändig ist (vgl. z. B. https://www.random.org)

3 Je einfacher, desto besser, zum Beispiel: TextEdit auf Mac OS oder Editor bzw. Notepad auf Windows. Wichtig ist, dass der Code als reiner Text, mit der Endung .html abgespeichert wird.

4 Für den Einstieg eignet sich z. B.: https://www.w3schools.com/js.

Code

Eine elementare Einführung in die Programmierung als künstlerische Praktik

Der Ausgangspunkt für dieses Interview ist ein Buch, das Sie 2005 gemeinsam geschrieben haben: «CodeArt Eine elementare Einführung in die Programmierung als künstlerische Praktik» (Trogemann/Viehoff 2005). Das sehr umfang- und facettenreiche Buch ist – meines Wissens – das einzige, welches sich in diesem Ausmaß der Programmierung im künstlerischen Kontext widmet. Der erste Teil befasst sich mit der programmierbaren Maschine sowie einer umfassenden Auslegung der damit einhergehenden Fragestellungen und schafft eine Verbindung zum Kontext der künstlerischen und kulturellen Praxis.

Art

Georg Trogemann und Jochen Viehoff
Interview, durchgeführt von Adriana Mikolaskova Nautsch

Der zweite Teil besteht aus einer für Anfängerinnen und Anfänger sehr gut nachvollziehbaren, sorgfältig konzipierten Programmiereinführung, welcher auch das sogenannte «Codekit» – eine Sammlung von Beispielen und Modulen – angegliedert ist. Im Gegensatz zu allgemeinen Programmiereinführungen beziehen sich die Beispiele auf Bilder, Animationen und Klänge und ermöglichen so relativ rasch einen Zugang zu entsprechenden Experimenten. Die Einführung wird ergänzt durch sehr interessante und anregende Exkurse der Kultur- und Technikgeschichte und bettet die technischen Aspekte in philosophische Betrachtungen ein.