Interaktion und weitere Grundlagen

Erste Interaktion

Bisher haben wir Programme ausgeführt mit denen wir lediglich per Code interagieren konnten. Das ist in den meisten Fällen eher langweilig. Daher werden wir uns nun darum kümmern auch Benutzereingaben zu behandeln.

Einführungsbeispiel Mausposition

Einführungsbeispiel Mausposition

Folgender Code zeichnet ein Linienmuster von der linken oberen Ecke zum Mauszeiger.

```
void setup(){
  size(200,200);
  background(200);
}

void draw(){
  line(0, 0, mouseX, mouseY);
}
```

l0-processing-mouse-2.mp4

Folgender Code sorgt dafür, dass eine Linie zwischen der linken oberen Ecke und dem Mauszeiger gezeichnet wird:

```
void setup(){
  size(200,200);
```

```
}

void draw(){
  background(200);
  line(0, 0, mouseX, mouseY);
}
```

l0-processing-mouse-1.mp4

An dieser Unterscheidung zeigt sich, wie wichtig das Setzen des Hintergrundes sein kann.

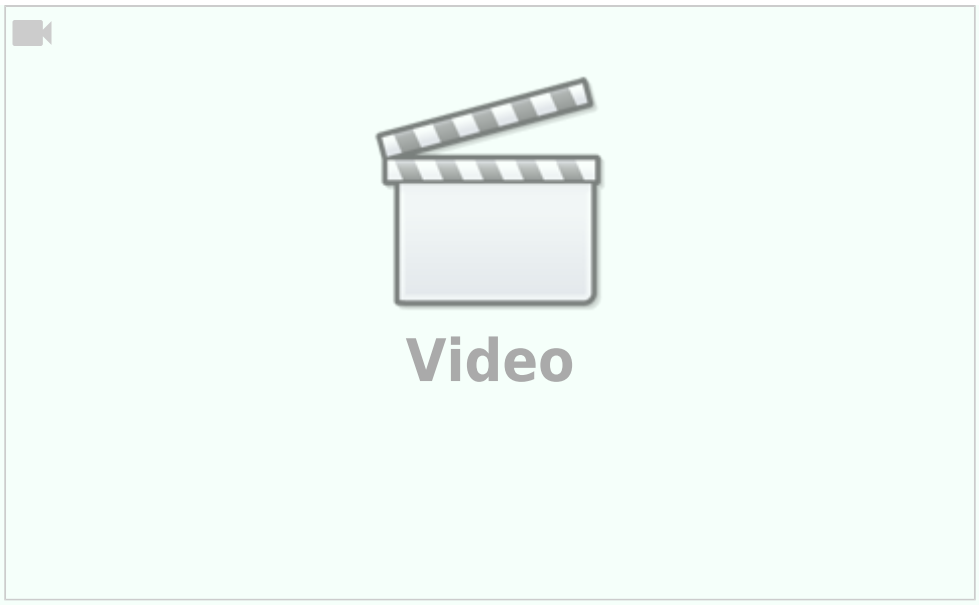
Aufgabe 1 - Erste eigene Gehversuche[1)]

1. Erzeuge ein Quadrat, das sich mit Hilfe des Mauszeigers bewegen lässt.
2. Nutze die Position des Mauszeigers, um das Quadrat in unterschiedliche Graustufen zu füllen. Hierbei hilft dir die `map()`-Funktion. Diese bildet einen Wert aus einem Intervall auf ein anderes ab. In unserem Beispiel möchten wir die Werte zwischen 0 und der Breite des Feldes auf die Werte zwischen 0 und 255 abbilden, da das unsere Graustufen sind. Somit ergibt sich für die Füllung des Rechtecks folgender Befehl: `map(mouseX, 0, width, 0, 255)`

Lösung zu Aufgabe 1)

```
void setup() {
  size(200, 200);
}

void draw() {
  background(200);
  fill(map(mouseX, 0, width, 0, 255));
  rect(mouseX, mouseY, 25, 25);
}
```



Maus- und Tastaturevents

Nun fehlt uns noch die Möglichkeit, mit einem Mausklick oder einem Tastendruck Einfluss auf unser Programm zu nehmen. Zu diesem Zweck müssen wir uns klar machen, was diese Aktionen bedeuten. Während wir mit setup() die statischen Eigenschaften unseres Programms festlegen, aktualisiert draw() ständig unsere Graphik. Ein Mausklick oder ein Tastendruck sind dagegen **Ereignisse**. Sie sind Teil des Programmablaufs, die damit verbundenen Aktionen treten aber nur auf, wenn das Ereignis ausgelöst wird. Deshalb brauchen wir weitere Codeblöcke um das erweiterte Programm umzusetzen. Processing stellt uns unter anderem die Methoden mousePressed() und keyPressed() zur Verfügung. Folgendes Beispiel soll die Arbeitweise verdeutlichen:

```
void setup() {
  size(200, 200);
  background(255);
}

void draw() {
  // Bei Maus- oder Tastatur-Interaktionen wird eine draw()-
Methode
  // zwingend benötigt, selbst wenn nichts gezeichnet wird!
}

void mousePressed() {
  fill(175);
  rect(mouseX, mouseY, 15, 15);
}

void keyPressed() {
  background(255);
```

```
}
```

l0-processing-mouse-4.mp4

Auf unterschiedliche Tastaturevents, also das Drücken von unterschiedlichen Tasten gehen wir hier noch nicht ein, da wir dazu erst noch die bedingten Anweisungen brauchen, das kommt aber auch bald also keine Sorge .

Maus- und Tastaturevents

Linien mit der Maus zeichnen

Zusätzlich zu `mouseX` und `mouseY` stellt Processing mit `pmouseX` und `pmouseY` noch zwei weitere Systemvariablen zur Verfügung. Diese zwei Schlüsselwörter stehen für die vorhergehende (previous) Position des Mauszeigers, d.h. die Positionskoordinaten im vorhergehenden Durchlauf der draw()-Methode. Wir erinnern uns: Die Methode aktualisiert unsere Graphik standardmäßig 60 mal pro Sekunde. Auf diese Weise steht uns eine weitere interessante Interaktionsmöglichkeit zur Verfügung. Wir können beispielweise mit der line()-Methode zeichnen, indem wir sowohl Startpunkt als auch Endpunkt dynamisch verändern, dadurch dass wir den Endpunkt der Linie zur Startpunkt der Linie im nächsten Frame machen.

```
void setup() {
  size(200, 200);
  background(200);
}
```

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

l0-processing-mouse-5.mp4

Maus- und Tastaturevents

Aufgabe 3 - Reagieren auf Maus und Tastatur

Erstelle ein Programm, das das Schreiben auf der Zeichenfläche mit Hilfe der Maus ermöglicht. Beachte dabei, dass der "Stift" auch abgesetzt werden muss.

l0-processing-mouse-6.mp4

Hilfe

Setze die Farbe des Strichs (`stroke()`) je nach Bedarf auf unterschiedliche Graustufen.

Lösungsvorschlag

```
void setup() {
  size(200, 200);
  background(255);
  stroke(255);
}
```

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

void mousePressed() {
  stroke(175);
}

void mouseReleased() {
  stroke(255);
}
```

Das Prinzip EVA

Schaut euch das Fachkonzept: Das EVA-Prinzip an und beantwortet folgende Frage:

- Was ist das EVA-Prinzip? Erläutere am Beispiel einer Smartphone-App deiner Wahl.

Wie führt der Computer unsere Programme aus und wie war das noch mal mit den Einsen und Nullen?

Bisher haben wir schon ein paar Programme geschrieben. Wie aber wird der Code den wir schreiben vom Computer interpretiert. Lest die dazu das Fachkonzept: Compiler durch und beantwortet die folgende Frage:

- Welche Funktion erfüllt der Compiler?

Wie schreiben wir Code ordentlich auf?

Euch ist sicherlich bereits aufgefallen, dass wir den Code in unseren Übungsbeispielen immer in einer gewissen Form aufschreiben. Das hat durchaus einen Sinn. Wenn ihr euch beim Aufschreiben an gewissen Richtlinien oder einen gewissen "Style" haltet, wird euer Code besser lesbar für euch und für andere. Um dies etwas besser greifbar zu machen, lest euch das Fachkonzept: Code Style 1 durch.

[1)]

Aufgabe von: CommeniusWiki unter CC BY-SA

From:
https://wiki-mathe-info.de/ - **Wiki: Mathe und Info**

Permanent link:
https://wiki-mathe-info.de/info/sek-ii/e2/processing/p2-grundlagen-2

Last update: **2025-02-19 14:22**