# PROCESSING

## EINE EINFÜHRUNG IN DIE INFORMATIK

Created by Michael Kirsch & Beat Rossmy

# INHALT

# EINLEITUNG

# ZIELE

- Programmierung (imperativ und objektorientiert)
- Algorithmen und Datenstrukturen
- Anhand von Java

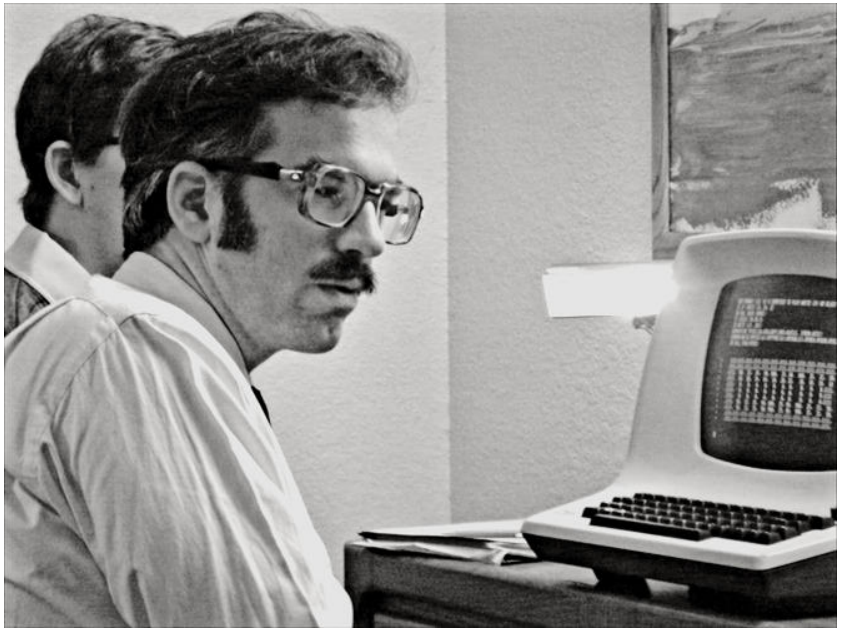
# DER INFORMATIKER – EIN KLISCHEE

- Programmiert den ganzen Tag zuhause alleine in seinem dunklen Zimmer.
- Beschäftigt sich mit kryptischen Botschaften auf seinem Rechner (vor allem 0en und 1en).

http://www.cinema52.com/2013/wp-content/uploads/2013/09/Nedry.png

# DER INFORMATIKER – EIN KLISCHEE

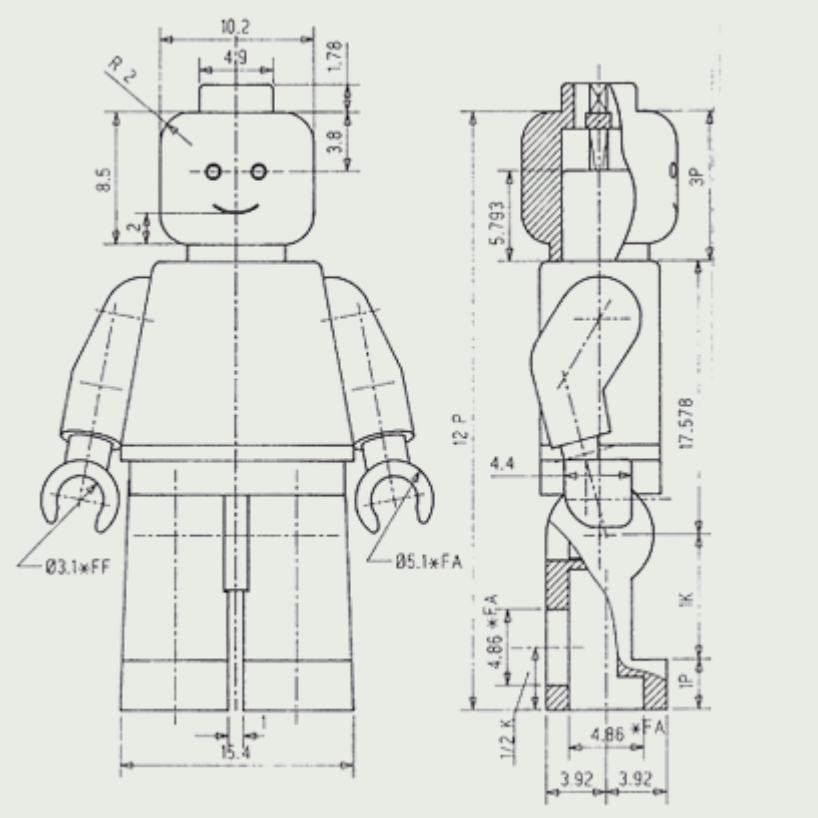
- Das Größte ist für ihn sein System neu aufzusetzen und bei Fehlern anzupassen.
- Programmieren um des Programmieren willens! Wozu auch sonst?



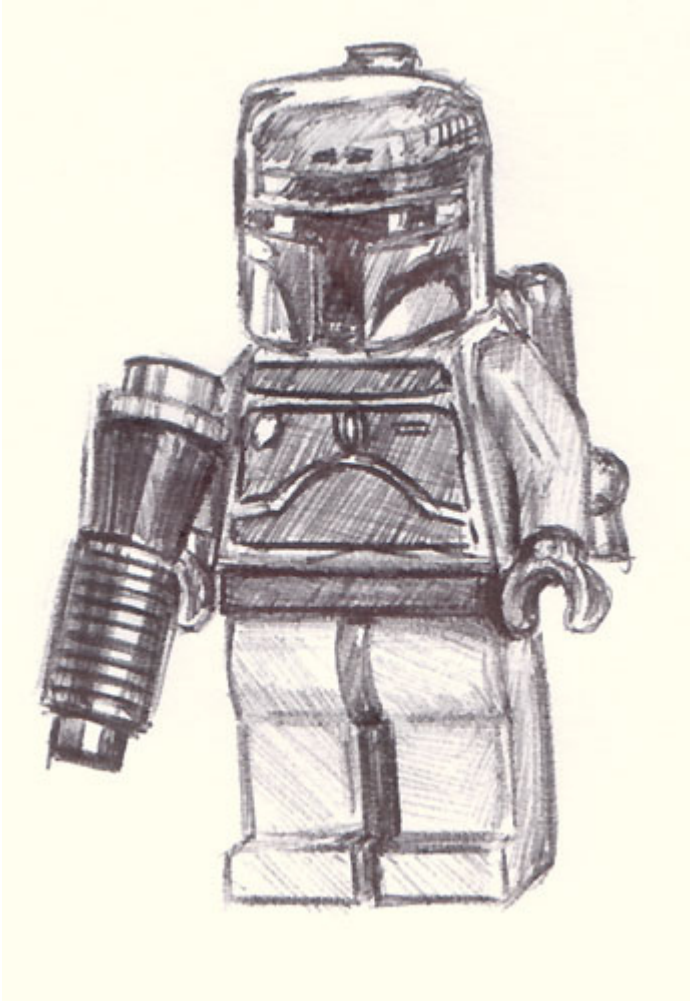
http://h.fastcompany.net/multisite_files/cocreate/imagecache/slideshow_large/slideshow/2013/07/1683410-slide-s-7-vintage-geek-computer-chess-looks-back-at-1980s-era-nerd-culture.jpg

Muss das so sein?

# WERKZEUG NICHT SELBSTZWECK!



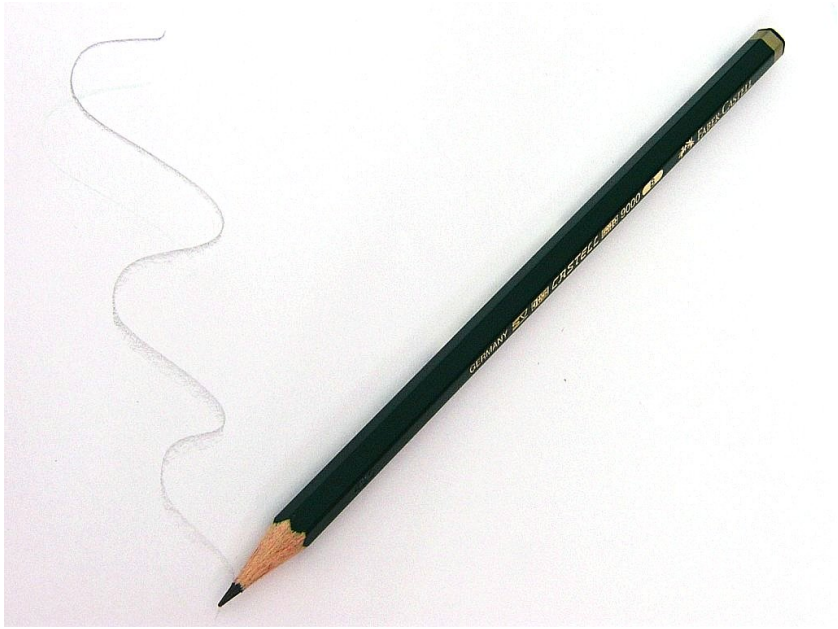

http://blog.dersven.de/user/files/gadgets/lego30-technischeZeichnung.png



https://c1.staticflickr.com/1/140/330870137_e8dec5b331.jpg

# WERKZEUG NICHT SELBSTZWECK!

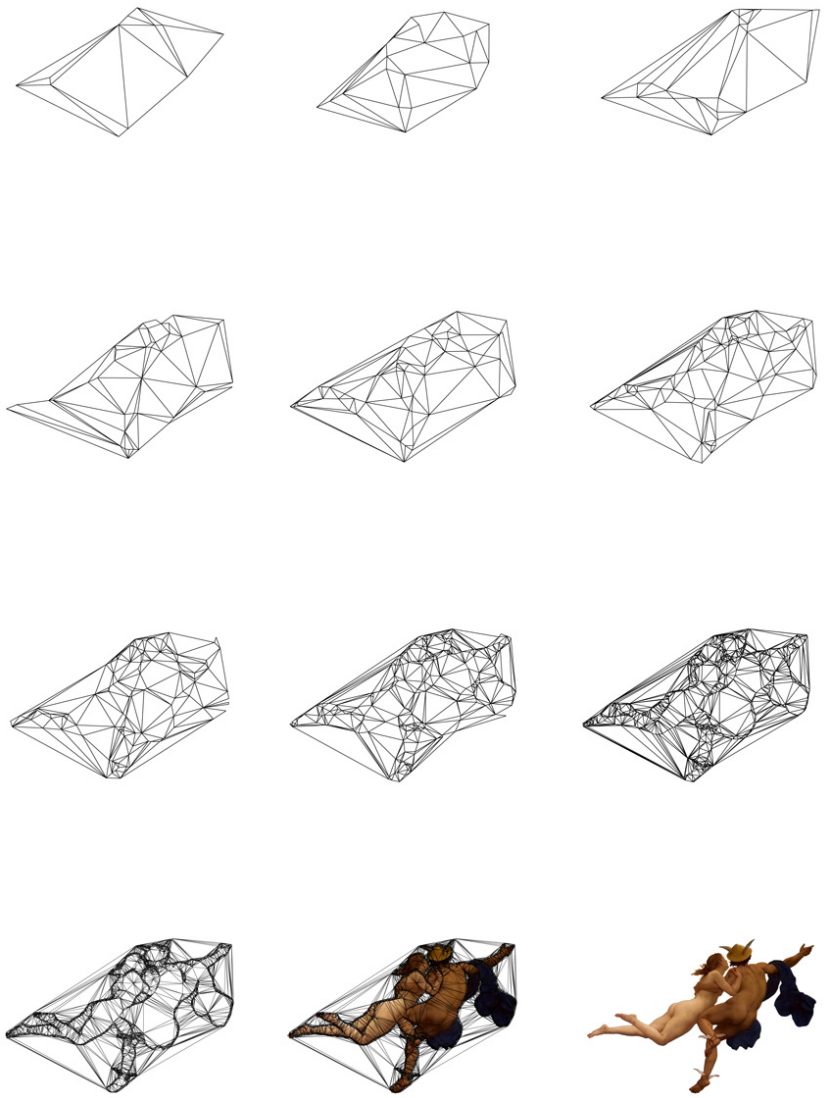
- Informatik kann ein Werkzeug sein wie Malerei, Zeichnung, …
- Wie und wozu wir diese einsetzen, ist uns überlassen!



https://upload.wikimedia.org/wikipedia/commons/2/23/Bleistift1.jpg

# DIGITALE KUNST

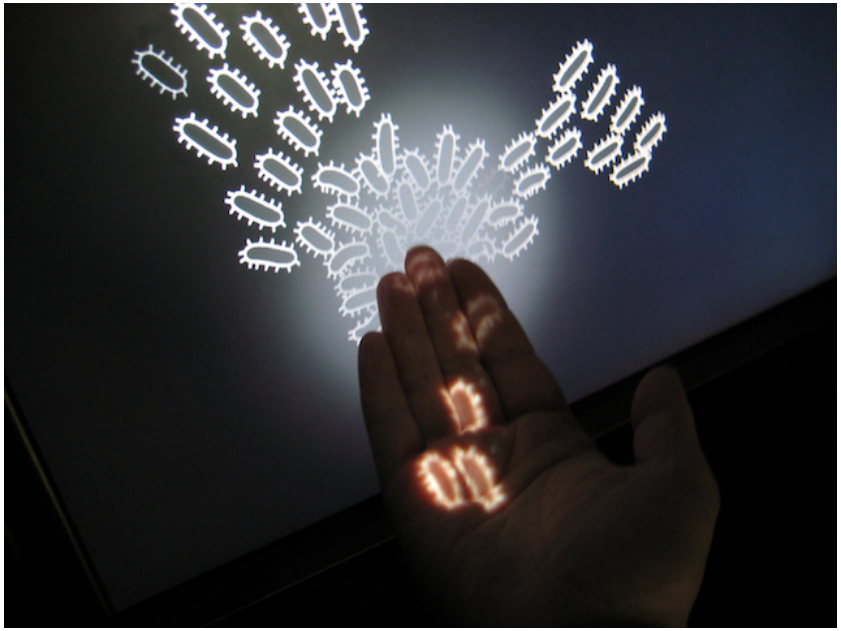
- Programmierung als Mittel um Ideen zu verwirklichen, die so nicht mit herkömmlicher Software umsetzbar sind.
- Videoeffekte, 3D-Animation, ...
- Strata #3



http://www.quayola.com/Qsite/wp-content/uploads/2011/07/process.jpg

# GRENZBEREICH DIGITAL/ANALOG

- Aufeinandertreffen digitaler Technologie und realer Umgebung.
- Programmierung zur Koordinierung und Generierung.
- KEYFLEAS



http://csugrue.com/images/db/db_webpic_1.jpg

# GRAUE EMINENZ

- Programmierung als unsichtbares Gehirn hinter Installationen, Maschinen, ...
- Worin steckt eigentlich überall Informatik?
- LIGHT KINETICS

http://payload372.cargocollective.com/1/4/156755/9744712/prt_594x203_1429610653.jpg

# THEORIE

# PROGRAMMIERSPRACHEN

- Programmiersprache: unser Weg zur Kommunikation mit dem Computer.
- Wie bei jeder Sprache gibt es Regeln und Konventionen, an welche man sich halten muss, damit man auch verstanden wird!

# JAVA EINE SPRACHE VON VIELEN.

- Die unterschiedlichen Sprachen unterscheiden sich in ihrer Nähe entweder zur Maschine oder eben zum Menschen.
- D.h.: Eine Sprache ist entweder für uns leicht zu verstehen, muss dann aber für den Computer in mehreren Schritten verständlich gemacht werden (automatisch).
- Oder die Sprache ist schwer zu lesen/verstehen, kann aber schneller vom Computer verarbeitet werden.

# WARUM JAVA?

- Java findet die richtige Balance zwischen Verständlichkeit und Computernähe.
- Java ist Plattform-unabhängig (läuft auf Windows, Mac und Linux).
- Java bietet einen Dialekt, der uns den Einstieg noch weiter vereinfacht: Processing!

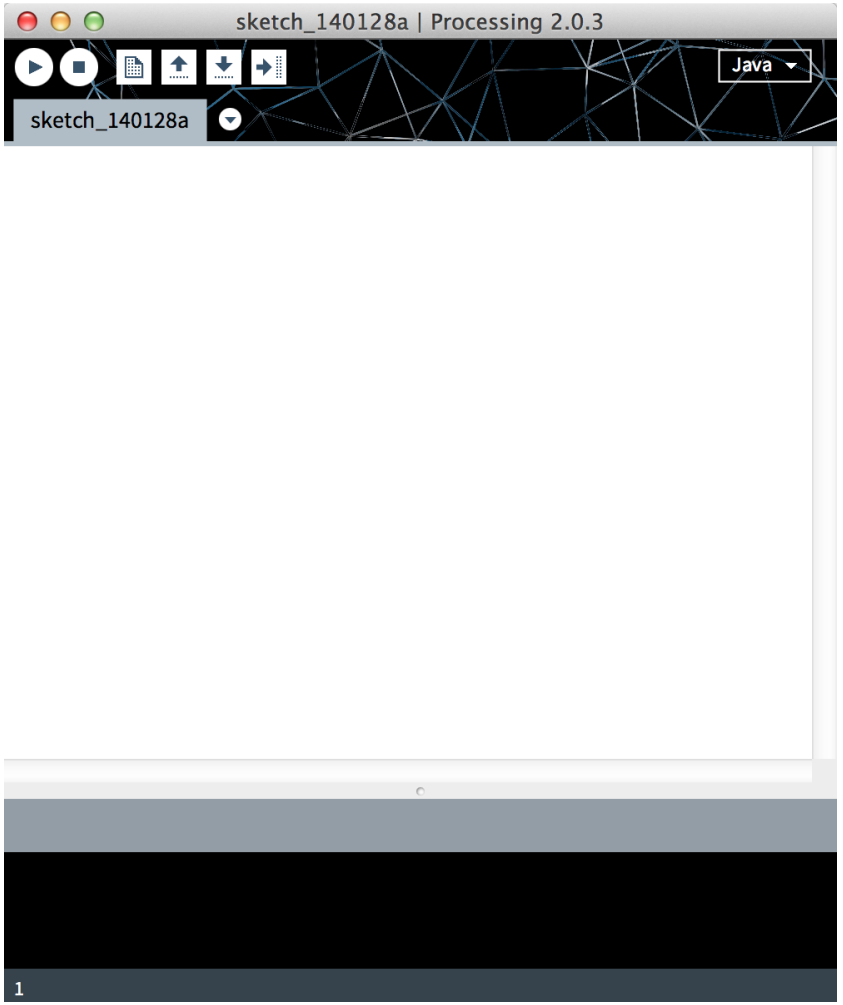
# PROCESSING

- Java Dialekt erfunden um Künstlern und anderen Kreativen den Zugang zu Programmierung zu erleichtern.
- Selbe Syntax wie Java.
- Aber weniger Code um grafische Ausgabe zu erreichen!

http://upload.wikimedia.org/wikipedia/commons/thumb/5/59/Processing_Logo_Clipped.svg/1000px-Processing_Logo_Clipped.svg.png

# PROCESSING IDE

- Play/Stop Button zum Starten des Programms.
- Textfeld zur Eingabe unseres Codes.
- Konsole: Textausgabe des Programms an uns, um Fehler mitzuteilen oder Werte zu ermitteln.




http://codecodigo.com/code/img/prosjs/points/proc_ide_02.png

# ANWENDUNG

# SPRUNG INS KALTE WASSER

| **Was ich sagen möchte:** | **Wie ich es sage:** |
|---|---|
| Male ein Rechteck! | `rect(x,y,w,h);` |
| Schreibe "Hello World!". | `print("Hello World!");` |
| Erzeuge eine zufällige Zahl zwischen 0 und 1. | `random(1);` |
| Fülle den Hintergrund schwarz. | `background(0);` |

# WIR BASTELN EIN DAUMENKINO

- Vorbereitung (**`setup`**): wir schneiden Papier in der Größe 100mm * 100mm
- Zeichnen (**`draw`**): Male roten Hintergrund, zeichne einen Kreis darauf.

```
void setup () {
   size(100,100);
}

void draw () {
   background(255,0,0);
   ellipse(50,50,50,50);
}
```

# SETUP? DRAW?

- **`setup`** wird genau einmal und zwar zu Beginn des Programms ausgeführt.
- **`draw`** wird danach immer und immer wieder Zeile für Zeile durchlaufen.
- Das betrifft alle Zeilen innerhalb der **`{}`**

```
void setup () {
   size(100,100);
}

void draw () {
   background(255,0,0);
   ellipse(50,50,50,50);
}
```

# PROCESSING BASICS

## STRUCTURE

| | |
|---|---|
| Setup Funktion | `void setup() {...}` |
| Zeichen Funktion | `void draw() {...}` |
| Größe des Programms | `size(w,h);` |
| Schreibe einen Wert in die Konsole | `println("something");` |

# PROCESSING BASICS

## 2D PRIMITIVES

| | |
|---|---|
| Rechteck | `rect(x,y,w,h);` |
| Ellipse | `ellipse(x,y,w,h);` |
| Linie | `line(x1,y1,x2,y2);` |
| Punkt | `point(x,y);` |

# PROCESSING BASICS

## COLOR

| | |
|---|---|
| Hintergrundfarbe | `background(r,g,b);` |
| Füllfarbe | `fill(r,g,b);` |
| Umrissfarbe | `stroke(r,g,b);` |

# VERKNÜPFUNG

# KANN MAN WERTE SPEICHERN?

- Vor **`setup`** können Variablen definiert werden, auf welche im ganzen Programm zugegriffen werden kann.
- In **`setup`** werden diese initialisiert. D.h. ihr Wert wird bestimmt. Z.B. soll **`x`** zu Beginn 0 sein.

```
int x;

void setup () {
   x = 0;
   size(100,100);
}

void draw () {
   background(255,0,0);
   ellipse(x,50,50,50);
}
```

# KANN MAN WERTE VERÄNDERN?

- **`x = x+1`**? mathematisch bedeutet das: 0=1?
- **`=`** fungiert in Java und Processing als Zuweisungsoperator. D.h. in die Variable **`x`** wird der Wert **`x+1`** gespeichert.

```
int x;

void setup () {
   x = 0;
   size(100,100);
}

void draw () {
   x = x+1;
   background(255,0,0);
   ellipse(x,50,50,50);
}
```

Was passiert in diesem Programm?

Wie können wir verhindern, dass der Kreis den sichtbaren Bereich verlässt?

# WENN DAS SO IST, DANN...

Bedingungen helfen uns bestimmte Situationen zu behandeln.

```
if (x > 100) {
  x = 0;
}
```

---

`if` markiert die in den `()` folgende Aussage als Bedingung.

```
if (x == 100) {
  x = 0;
}
```

---

In den `{}` steht die daraus resultierende Handlungsabfolge (falls wahr).

```
if (true) {
  x = 0;
}
```

# BOOLSCHE OPERATOREN

| | |
|---|---|
| Größer | `if (a > b) {...}` |
| Größer gleich | `if (a >= b) {...}` |
| Gleich | `if (a == b) {...}` |
| Kleiner gleich | `if (a <= b) {...}` |
| Kleiner | `if (a < b) {...}` |
| Ungleich | `if (a != b) {...}` |

# AUSBLICK

# NÄCHSTE SITZUNG

- Weitere mathematische Datentypen
- Strukturhilfen
- Processing Reference

# ÜBUNG

Unser Kreis soll nun bei Verlassen einer Seite seine Richtung ändern!

# QUELLEN


- http://www.cinema52.com/2013/wp-content/uploads/2013/09/Nedry.png
- http://h.fastcompany.net/multisite_files/cocreate/imagecache/slideshow_large/slideshow/2013/07/1683410-slide-s-7-vintage-geek-computer-chess-looks-back-at-1980s-era-nerd-culture.jpg
- http://blog.dersven.de/user/files/gadgets/lego30-technischeZeichnung.png
- https://c1.staticflickr.com/1/140/330870137_e8dec5b331.jpg
- https://upload.wikimedia.org/wikipedia/commons/2/23/Bleistift1.jpg
- http://www.quayola.com/Qsite/wp-content/uploads/2011/07/process.jpg
- http://csugrue.com/images/db/db_webpic_1.jpg
- http://payload372.cargocollective.com/1/4/156755/9744712/prt_594x203_1429610653.jpg
- http://upload.wikimedia.org/wikipedia/commons/thumb/5/59/Processing_Logo_Clipped.svg/1000px-Processing_Logo_Clipped.svg.png
- http://codecodigo.com/code/img/prosjs/points/proc_ide_02.png