Creative Coding

in Processing

Christian Gwiozda

FH Trier, FB Gestaltung, WS2013/2014

Warum Programmierung?

Warum Processing?

Unabhängigkeit von anderen Programmen... (wenn nötig)

Keine Limitierung...

Computer als Universalwerkzeug

Man kann schneller viele Variationen einer Idee testen

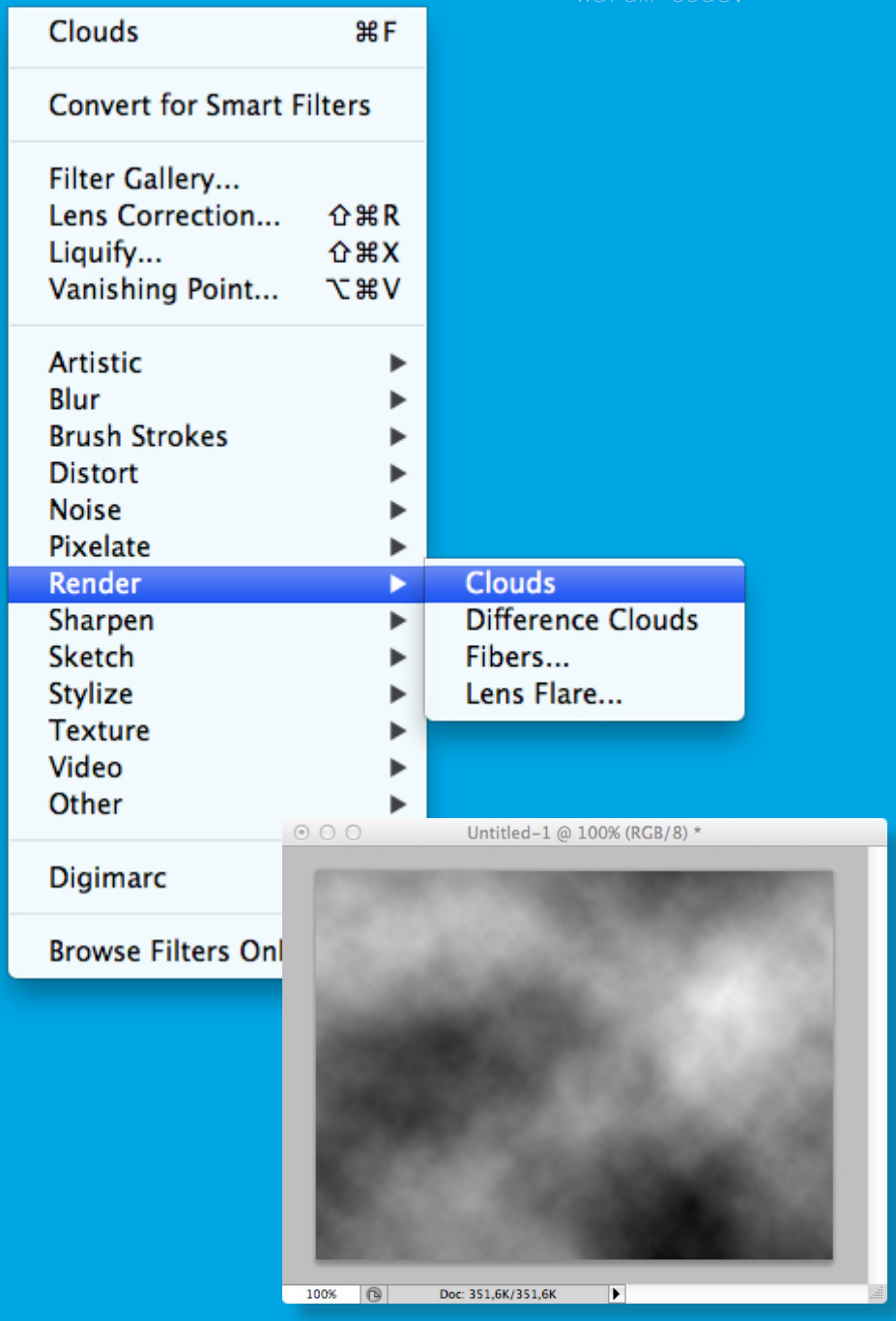
Verschiedene Medien: Bild, Ton, Animation, Print, Physical...

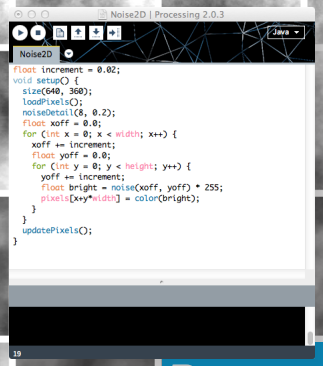
Modularität

Es macht manchmal Spass ;)

Photoshop Clouds Effekt

Processing Noise
Example->Math->Noise2D

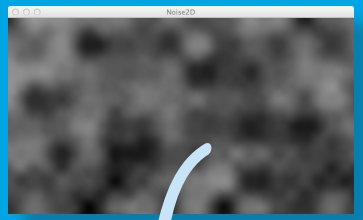


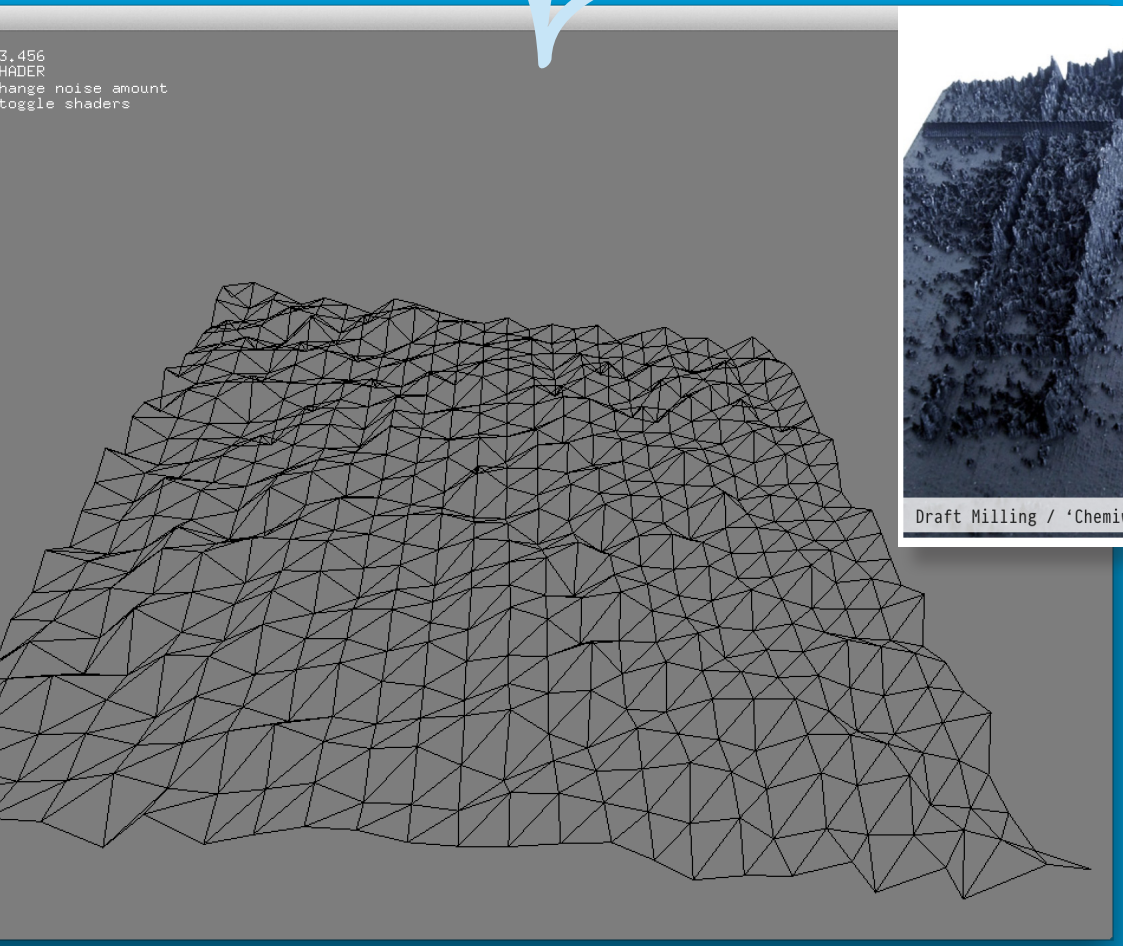


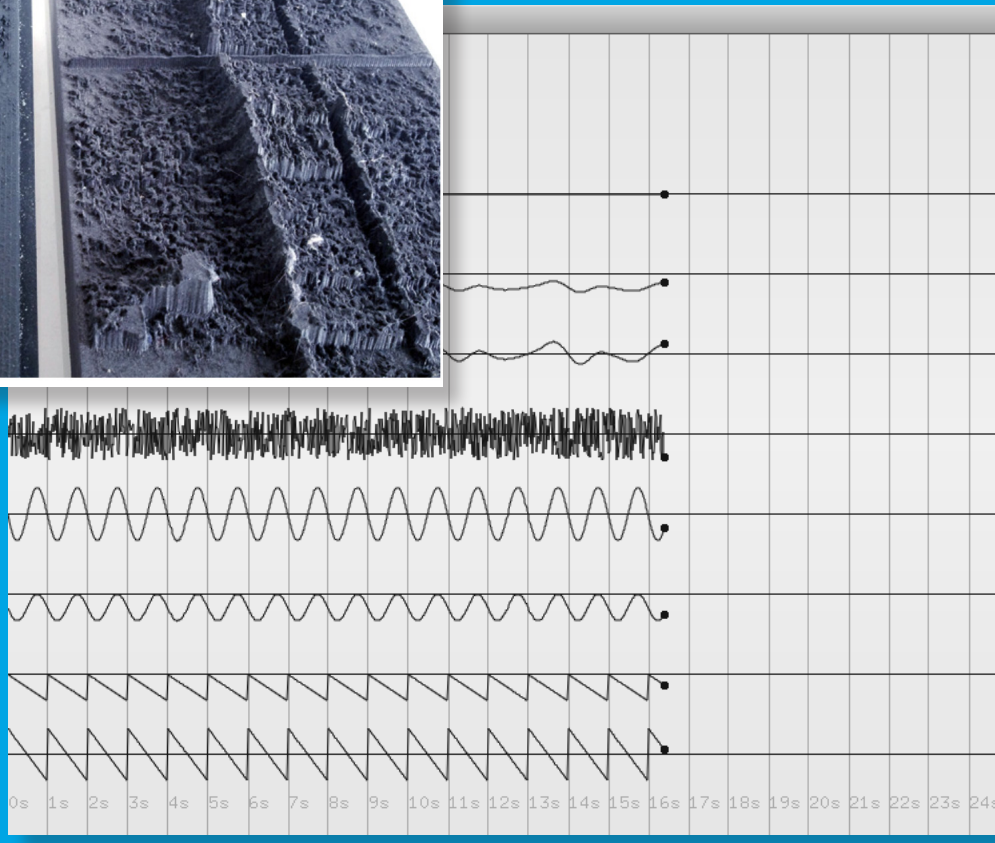
Processing:
Direkte Variationsmöglichkeit

Photoshop Clouds Effekt:
Garkeine Settings ?!

„Abstrakte Bausteine“







Was ist Processing?

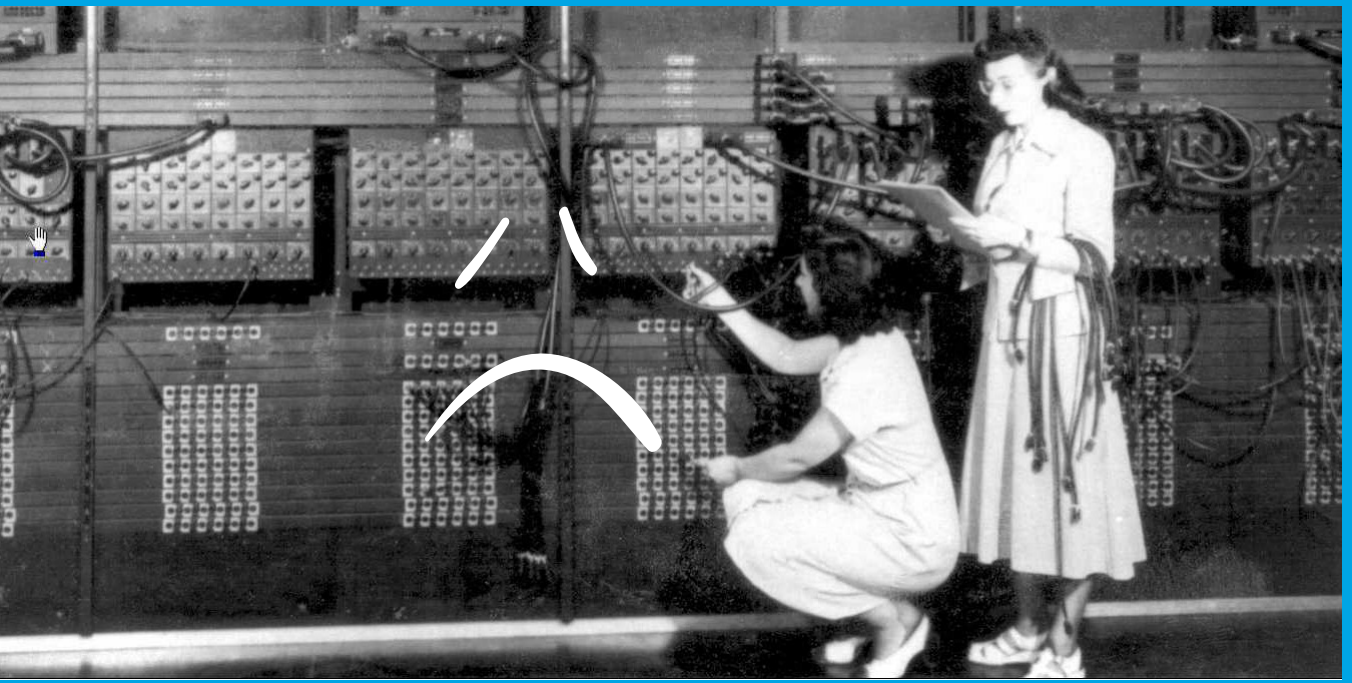
Programmiersprache -> Eine Variante von Java

Entwickelt 2001 am MIT in der Aesthetics & Computation Group

Zielgruppe sind Designer und Künstler

OpenSource!

Für fast alles gibt es schon fertigen Code in Libraries



Processing Mission Statement:

„Processing seeks to ruin the careers of talented designers by tempting them away from their usual tools and into the world of programming and computation. Similarly, the project is designed to turn engineers and computer scientists to less gainful employment as artists and designers.“

Anleitung:

1. Zeichne 10 Linien.

2. Die erste Line startet in der Mitte.

3. Sie geht ein “Kästchen” nach oben.

4. Jede weitere Linie beginnt am Ende der letzten Line.

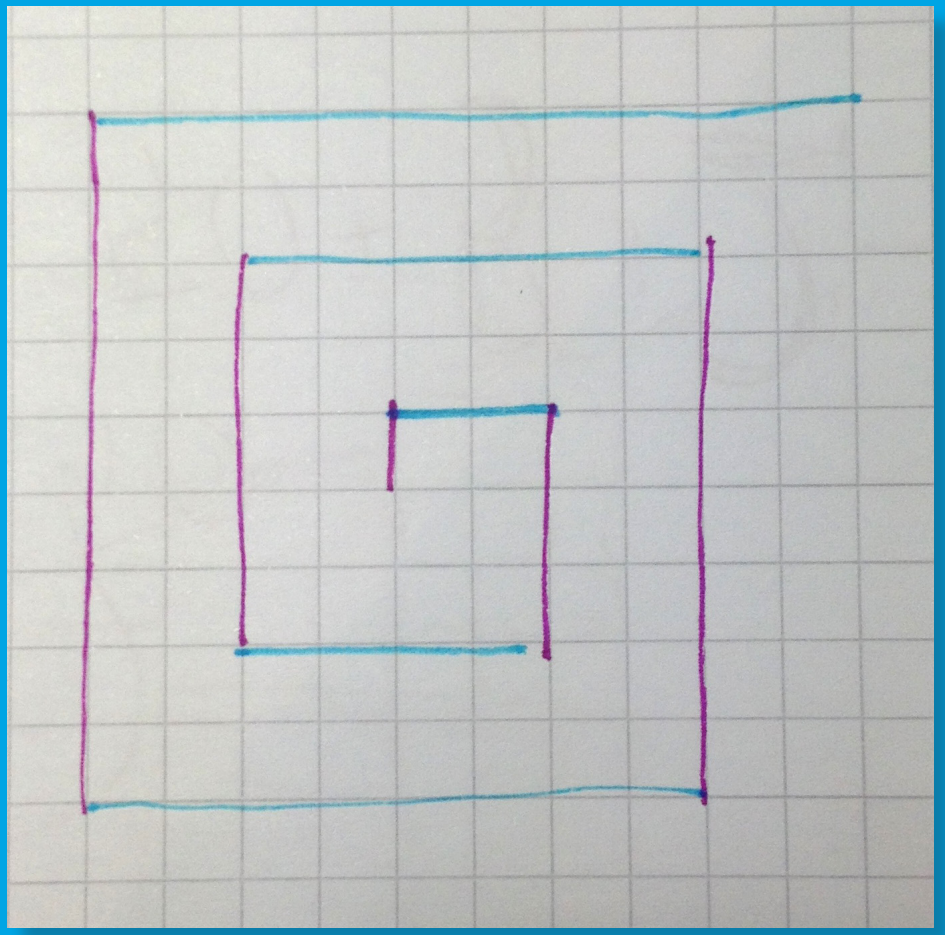
5. Jede neue Linie ist ein Kästchen länger als die vorherige.

6. Jede neue Linie dreht sich um 90° im Uhrzeigersinn.

7. Zwei Farben müssen sich immer abwechseln.

Code = Präzise Auflistung von Befehlen die der Computer ausführen soll.

```
0010001010110
0100100101010
```

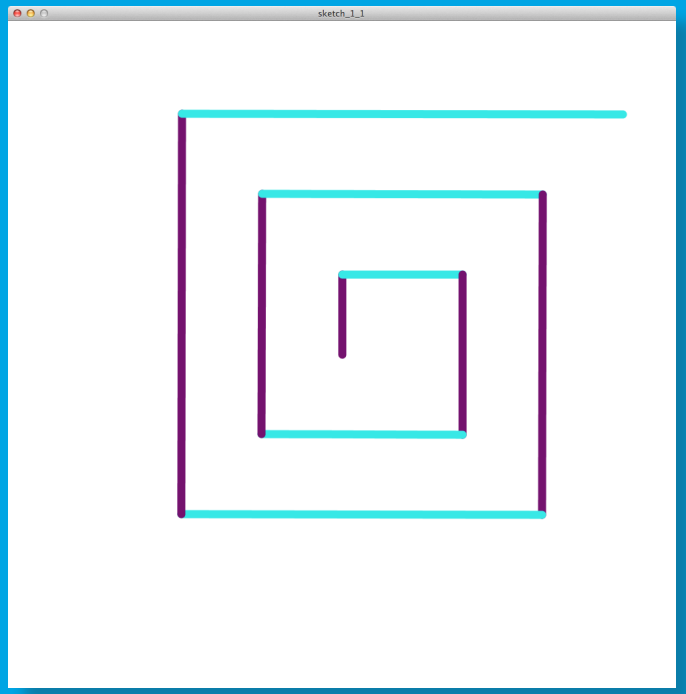


```
draw(“bild.jpg”, 10,10)
```

```
background(255, 255, 255);
size(1000, 1000);
smooth();
strokeWeight(12);
int step = 10;
float abstand = 120;
float plus = abstand/2;


translate(width/2, height/2);
for (int i = 1; i < step+1; i++) {

  if (i%2==0) {
    stroke(33, 232, 230);
  }
  else {
    stroke(118, 10, 108);
  }

  line(0, 0, 0, -abstand);
  translate(0, -abstand);

  rotate((TWO_PI/4));
  abstand += plus;
}
```

sketch_1_1

Die gleiche Anleitung in Processing “Sprache”



Processing Syntax

befehl(_,_,_);

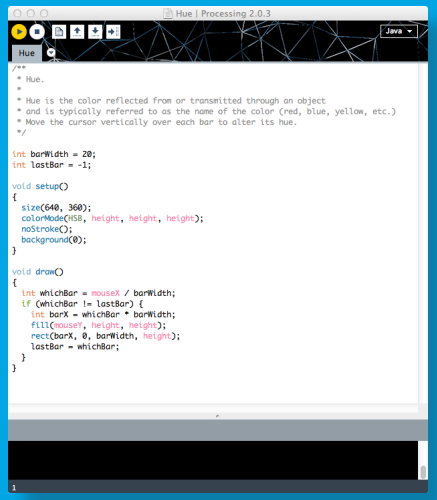
Funktion

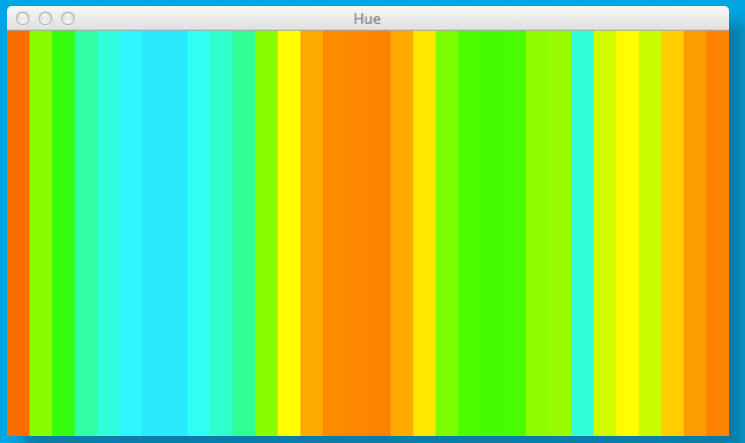
Argumente

z.B.

```
rect(20,100,40,20);
ellipse(200,200,10,10);
fill(120,120,0);
loadFont(helvetica, 16);
...
...
```

Problem 1:

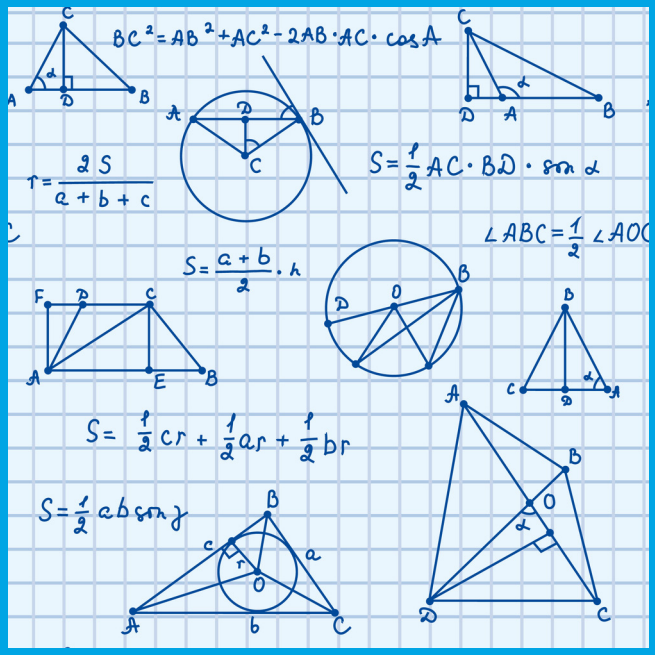

Hue | Processing 2.0.3
Java
Hue
/**
* Hue.
*
* Hue is the color reflected from or transmitted through an object
* and is typically referred to as the name of the color (red, blue, yellow, etc.)
* Move the cursor vertically over each bar to alter its hue.
*/
int barWidth = 20;
int lastBar = -1;
void setup()
{
size(640, 360);
colorMode(HSB, height, height, height);
noStroke();
background(0);
}
void draw()
{
int whichBar = mouseX / barWidth;
if (whichBar != lastBar) {
int barX = whichBar * barWidth;
fill(mouseY, height, height);
rect(barX, 0, barWidth, height);
lastBar = whichBar;
}
}

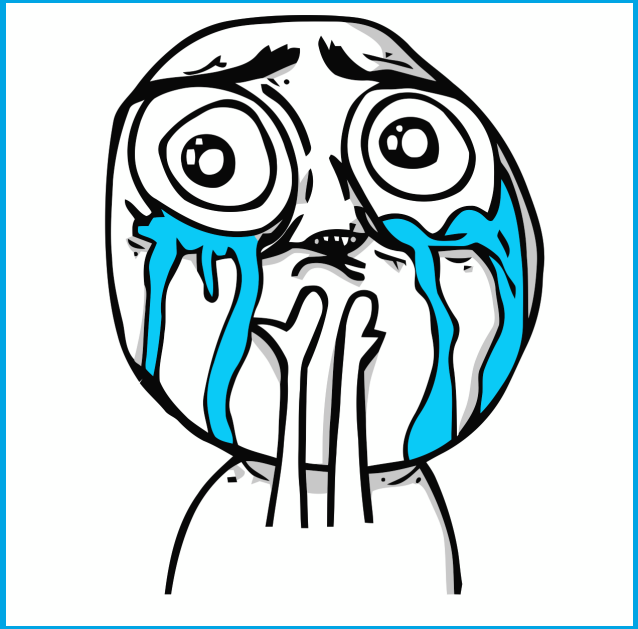

Hue


IT'S MAGIC.
I AIN'T GOTTA EXPLAIN SHIT.

Problem 2

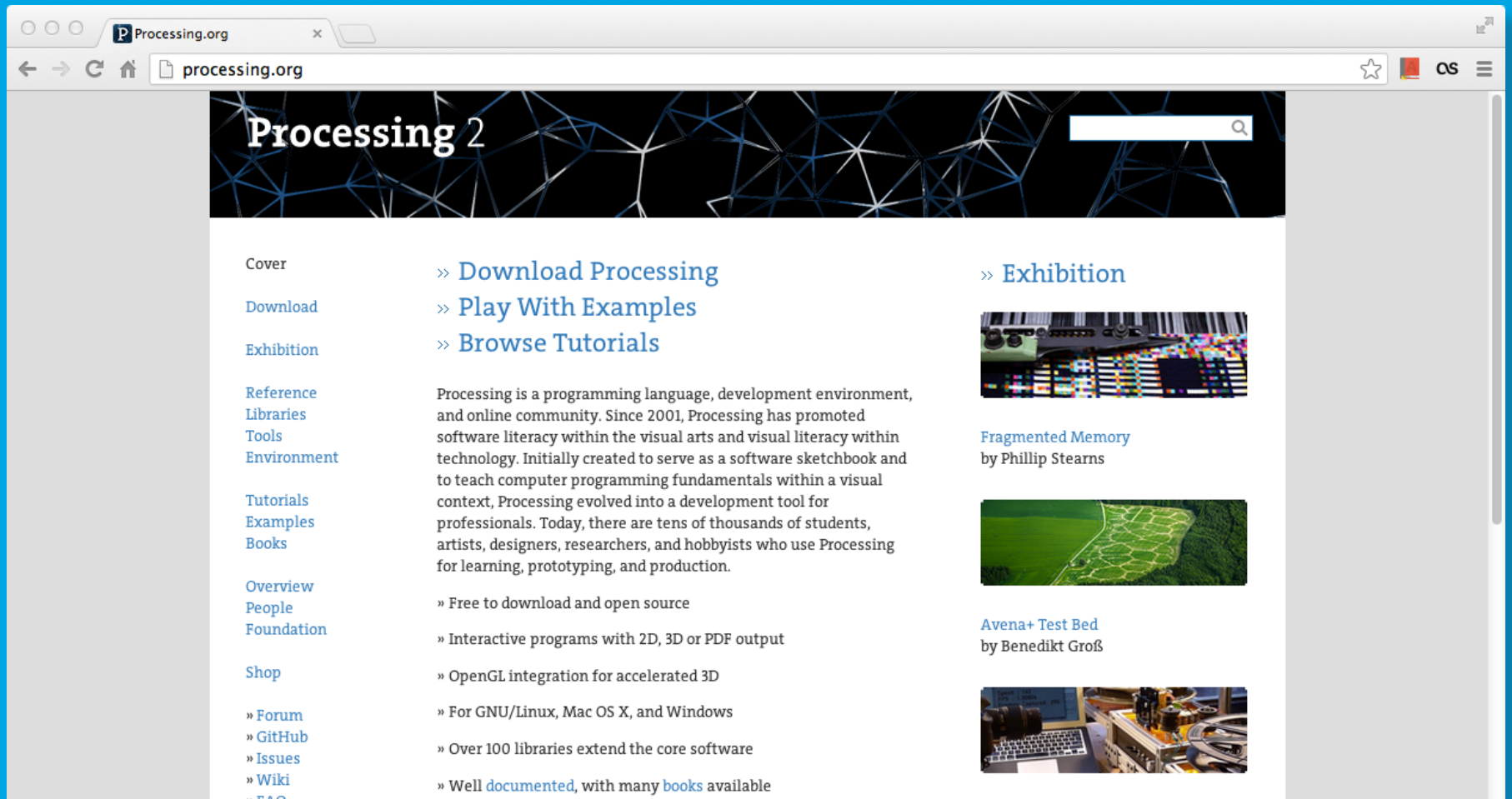

MATH
3×6=18


$BC^2 = AB^2 + AC^2 - 2AB \cdot AC \cdot \cos A$
$r = \frac{2S}{a+b+c}$
$S = \frac{1}{2} AC \cdot BD \cdot \sin\alpha$
$\angle ABC = \frac{1}{2} \angle AOC$
$S = \frac{a+b}{2} \cdot h$
$S = \frac{1}{2} cr + \frac{1}{2} ar + \frac{1}{2} br$
$S = \frac{1}{2} ab \sin\gamma$

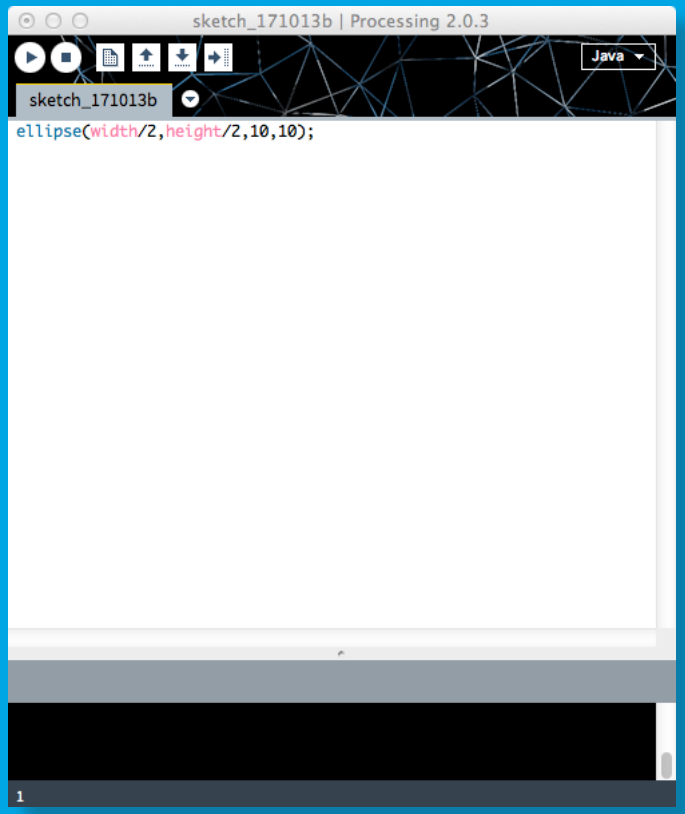


Processing Application

www.processing.org -> Download

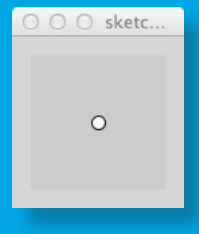


2.0.3 ??

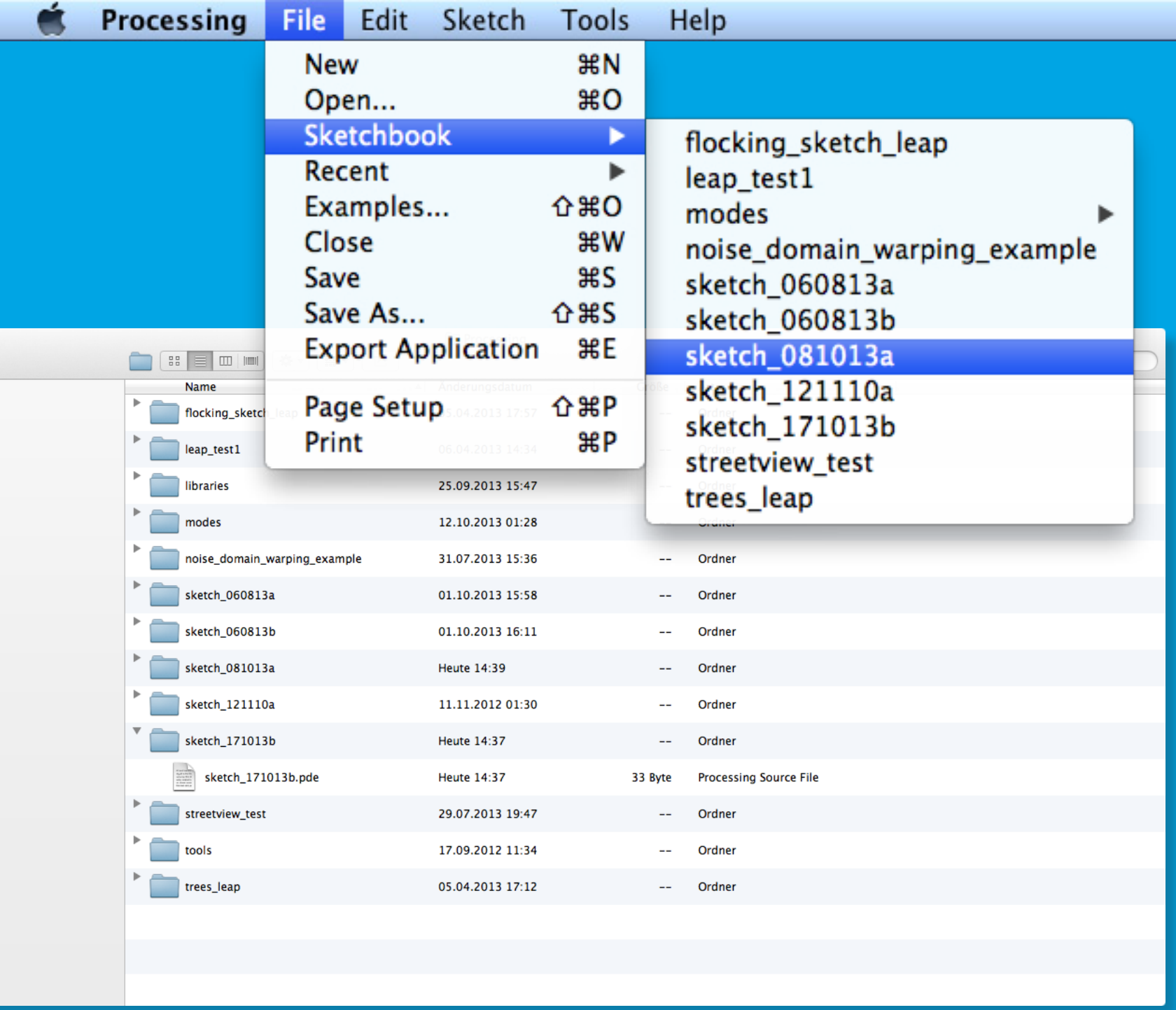


Processing Textfenster

Play / Stop
Buttons



Programm Fenster



Sketchbook Folder

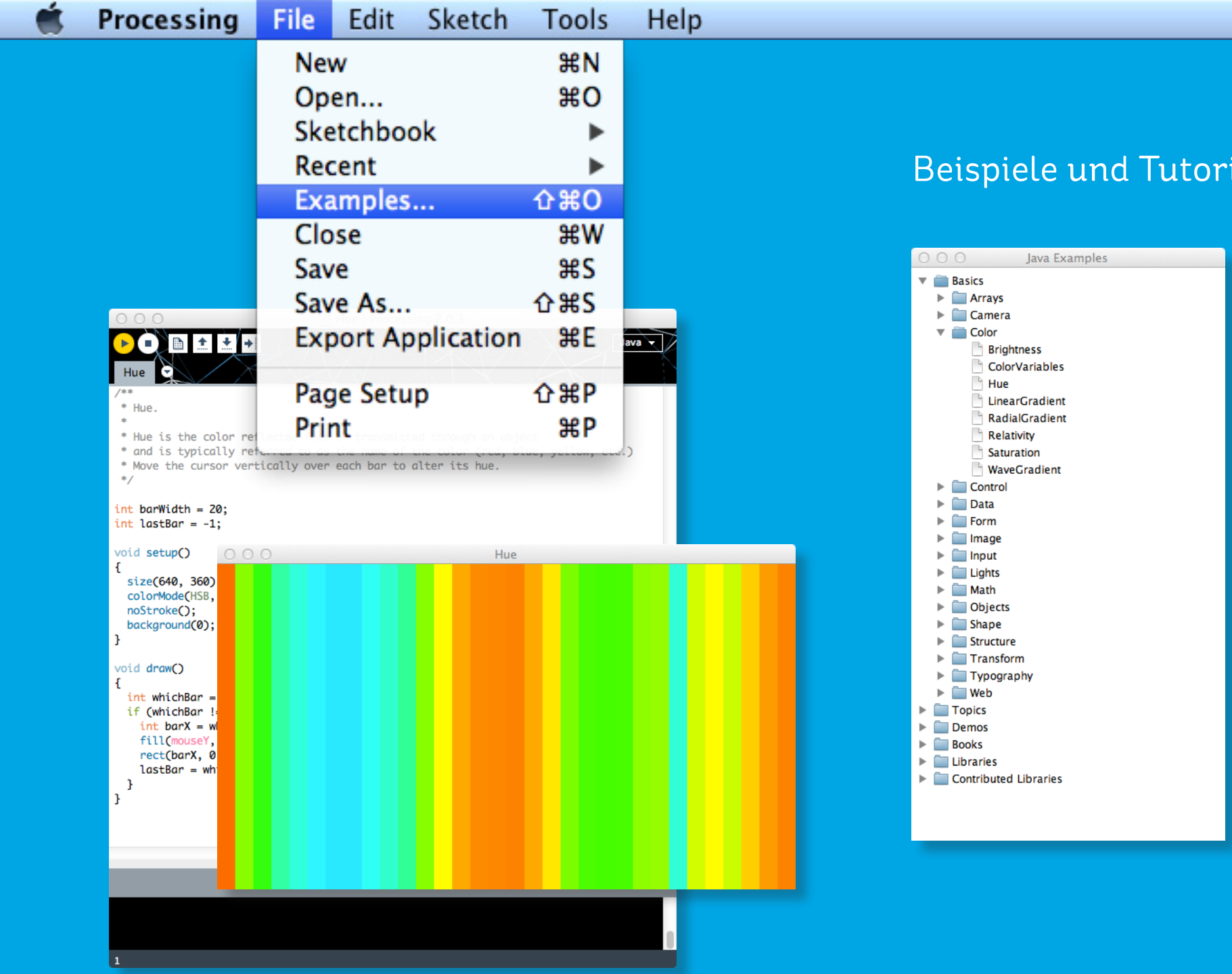
->

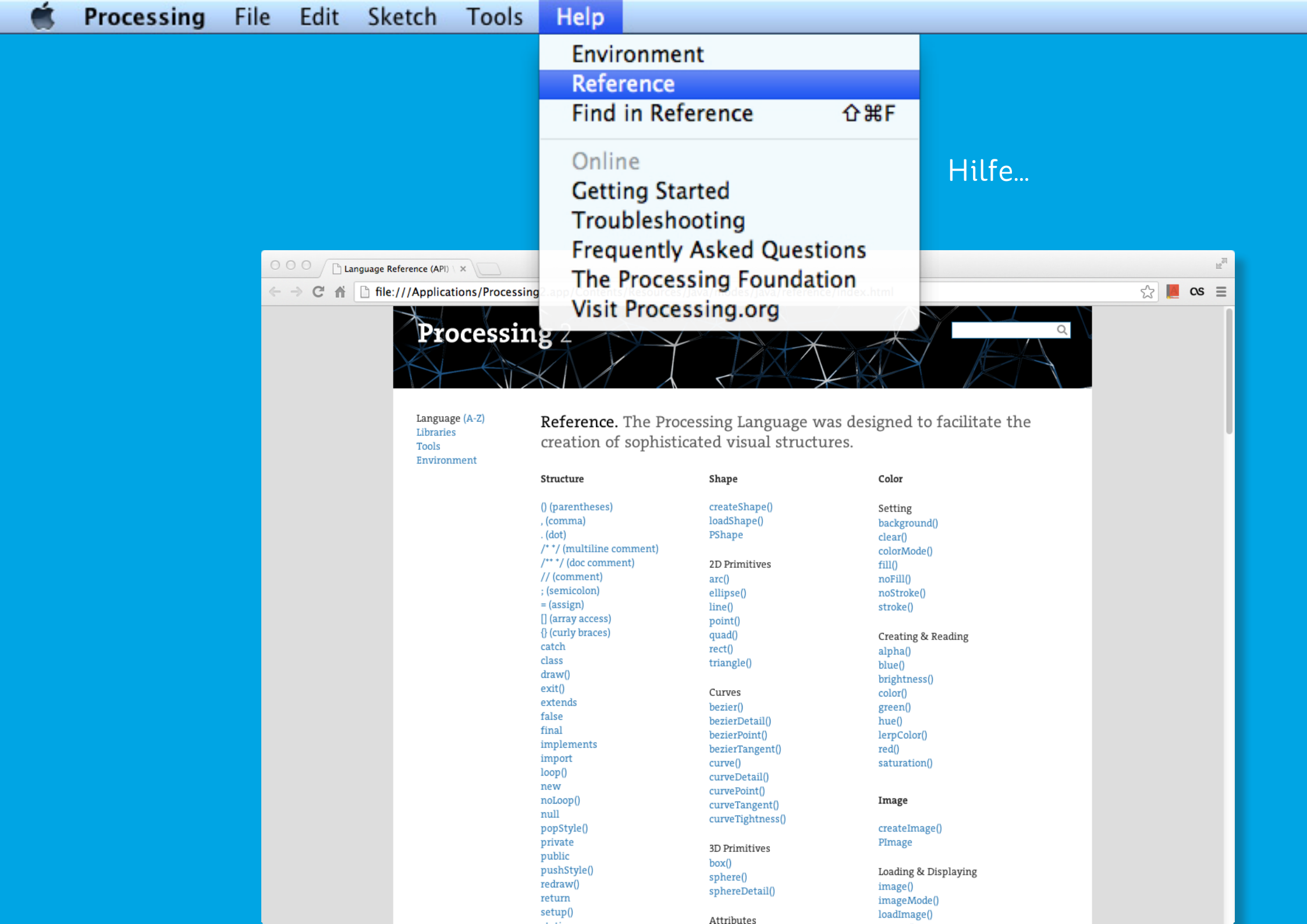
Sketch Folder

->

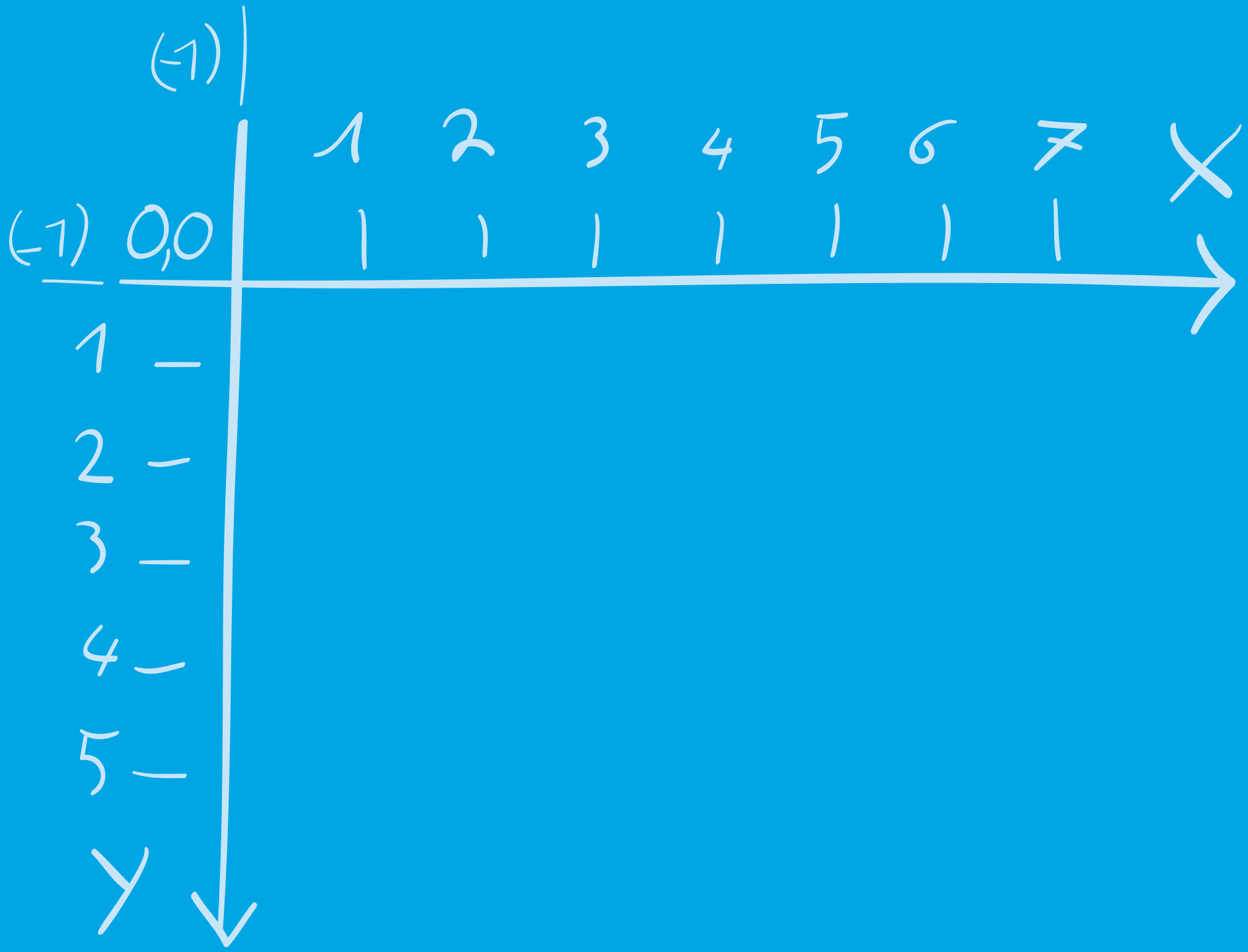
sketch_datum.pde

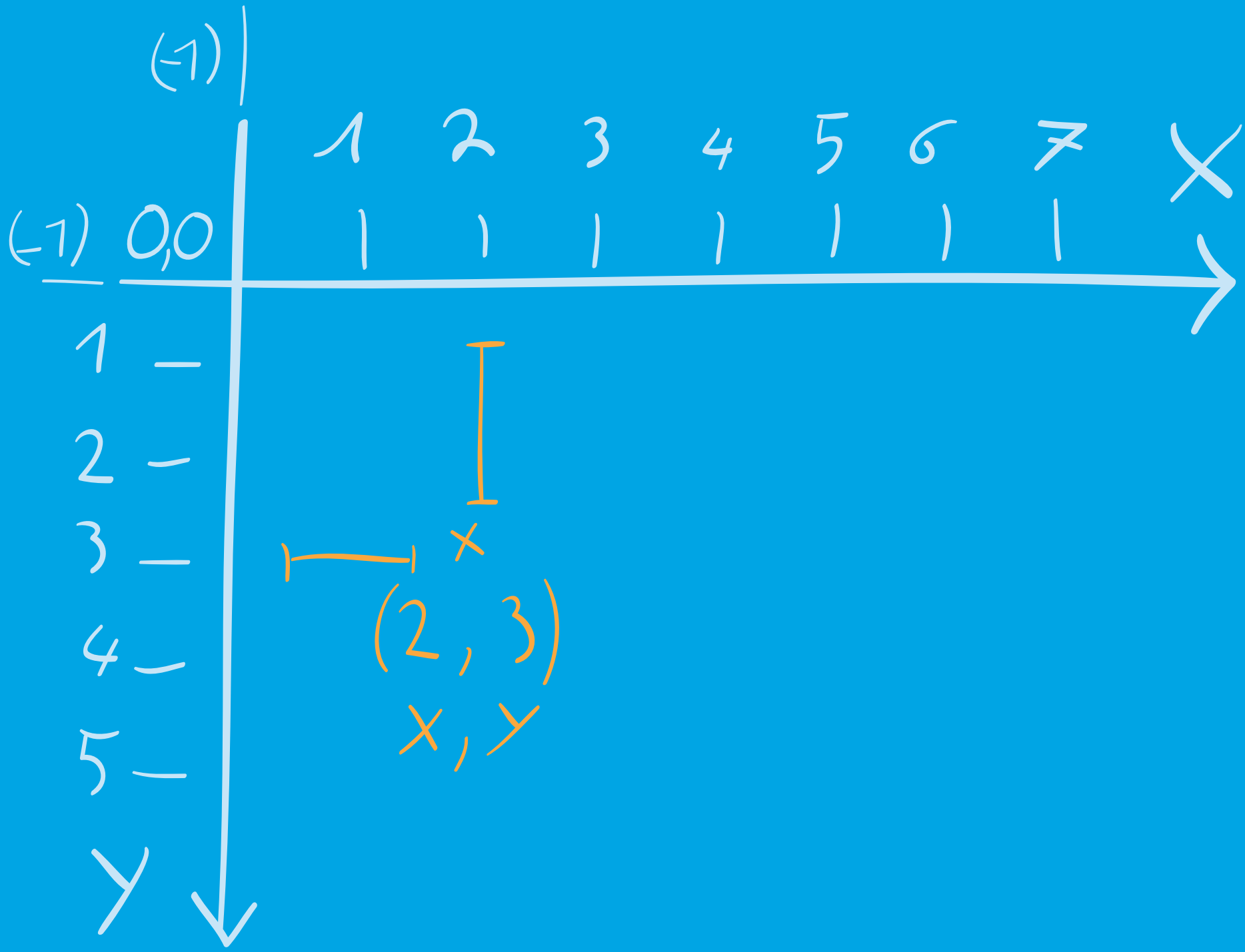
Beispiele und Tutorials

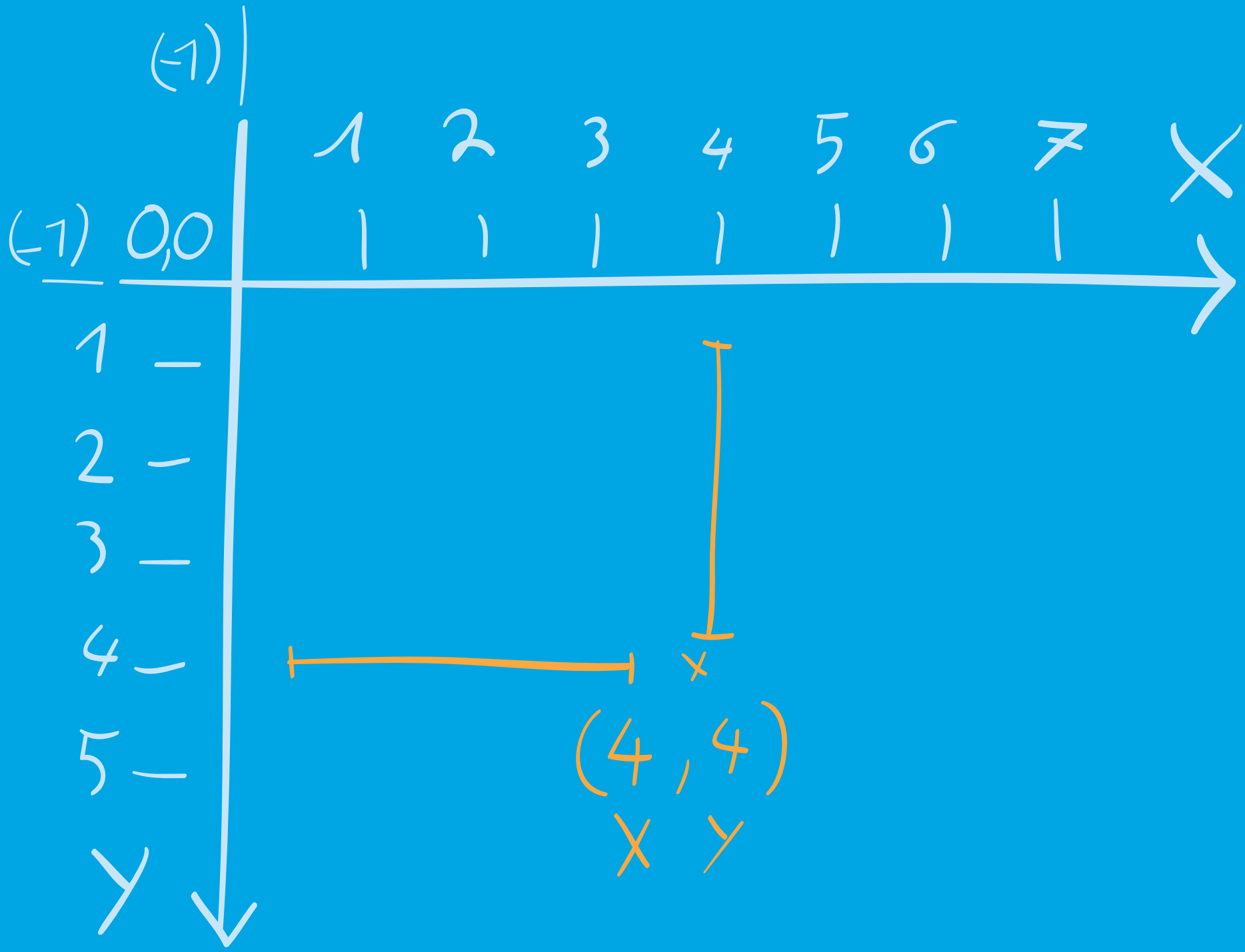



Processing File Edit Sketch Tools Help
Environment
Reference
Find in Reference ⇧⌘F
Online
Getting Started
Troubleshooting
Frequently Asked Questions
The Processing Foundation
Visit Processing.org
Hilfe...
Language Reference (API)
file:///Applications/Processing
Processing 2
Language (A-Z)
Libraries
Tools
Environment
Reference. The Processing Language was designed to facilitate the creation of sophisticated visual structures.
Structure
() (parentheses)
, (comma)
. (dot)
/* */ (multiline comment)
/** */ (doc comment)
// (comment)
; (semicolon)
= (assign)
[] (array access)
{} (curly braces)
catch
class
draw()
exit()
extends
false
final
implements
import
loop()
new
noLoop()
null
popStyle()
private
public
pushStyle()
redraw()
return
setup()
Shape
createShape()
loadShape()
PShape
2D Primitives
arc()
ellipse()
line()
point()
quad()
rect()
triangle()
Curves
bezier()
bezierDetail()
bezierPoint()
bezierTangent()
curve()
curveDetail()
curvePoint()
curveTangent()
curveTightness()
3D Primitives
box()
sphere()
sphereDetail()
Attributes
Color
Setting
background()
clear()
colorMode()
fill()
noFill()
noStroke()
stroke()
Creating & Reading
alpha()
blue()
brightness()
color()
green()
hue()
lerpColor()
red()
saturation()
Image
createImage()
PImage
Loading & Displaying
image()
imageMode()
loadImage()

Koordinaten und Pixel

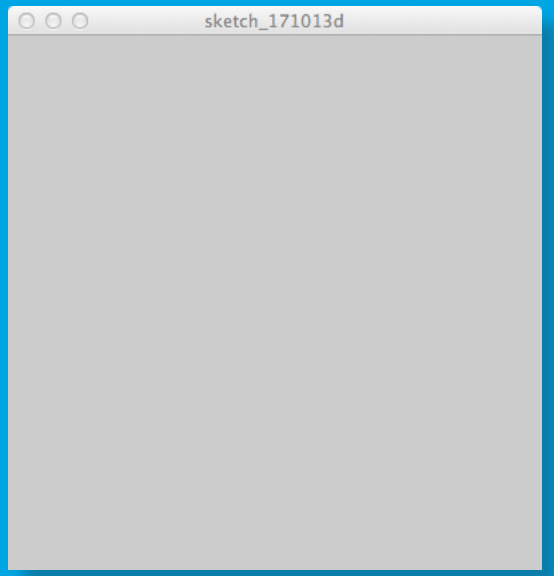

(-1)
1 2 3 4 5 6 7 X
(-1) 0,0
1 1 1 1 1 1 1
1 —
2 —
3 —
4 —
5 —
Y


(-1)
1 2 3 4 5 6 7 X
(-1) 0,0
1
2
3
4
5
Y
(2, 3)
x, y

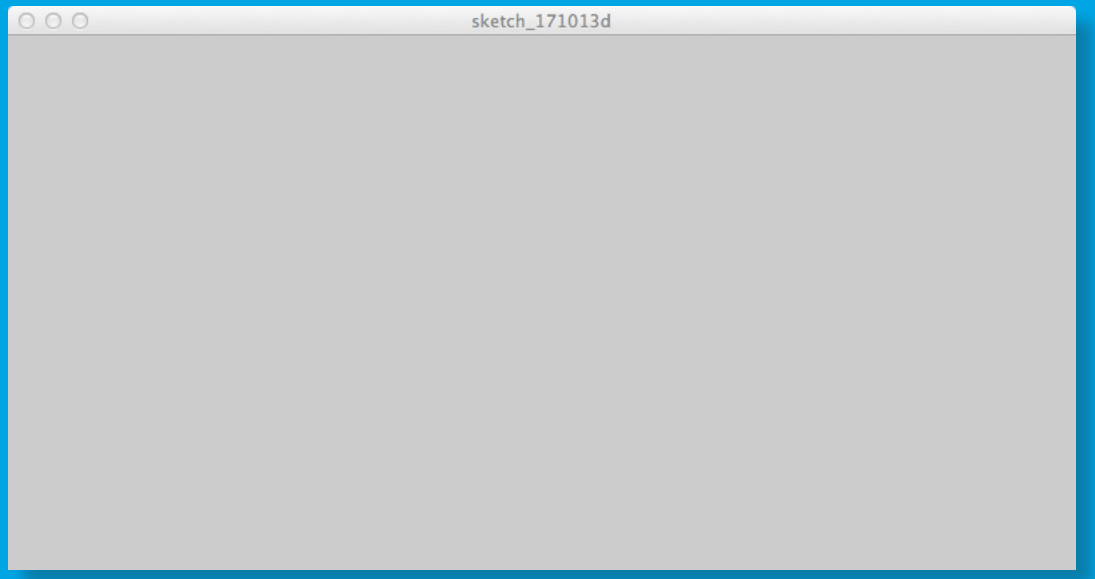

(-1)
1 2 3 4 5 6 7 X
(-1) 0,0
1
2
3
4
5
Y
(4,4)
X Y

size(Breite, Höhe) <- in Pixeln

```
size(400,400);
```

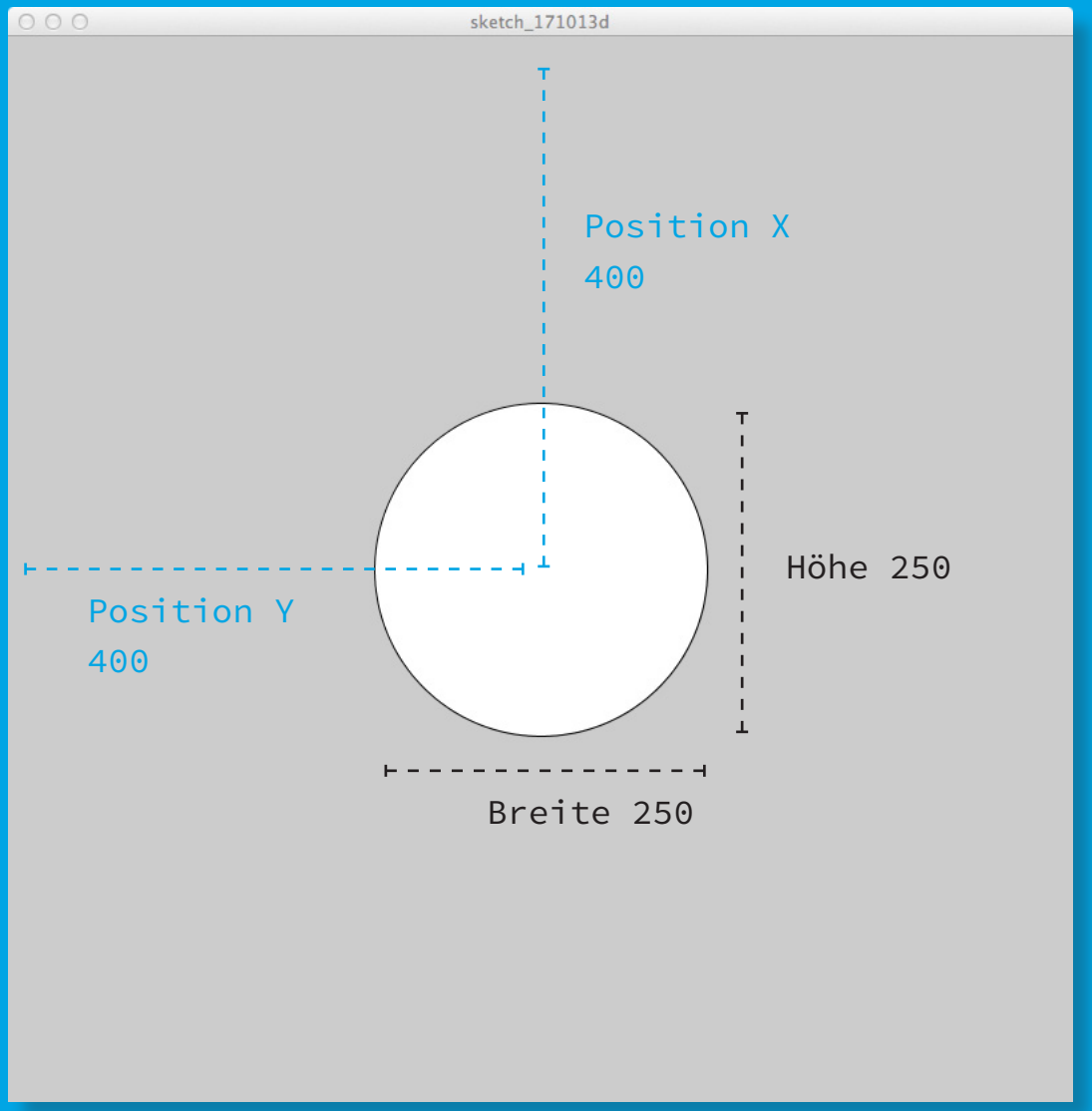



```
size(800,400);
```



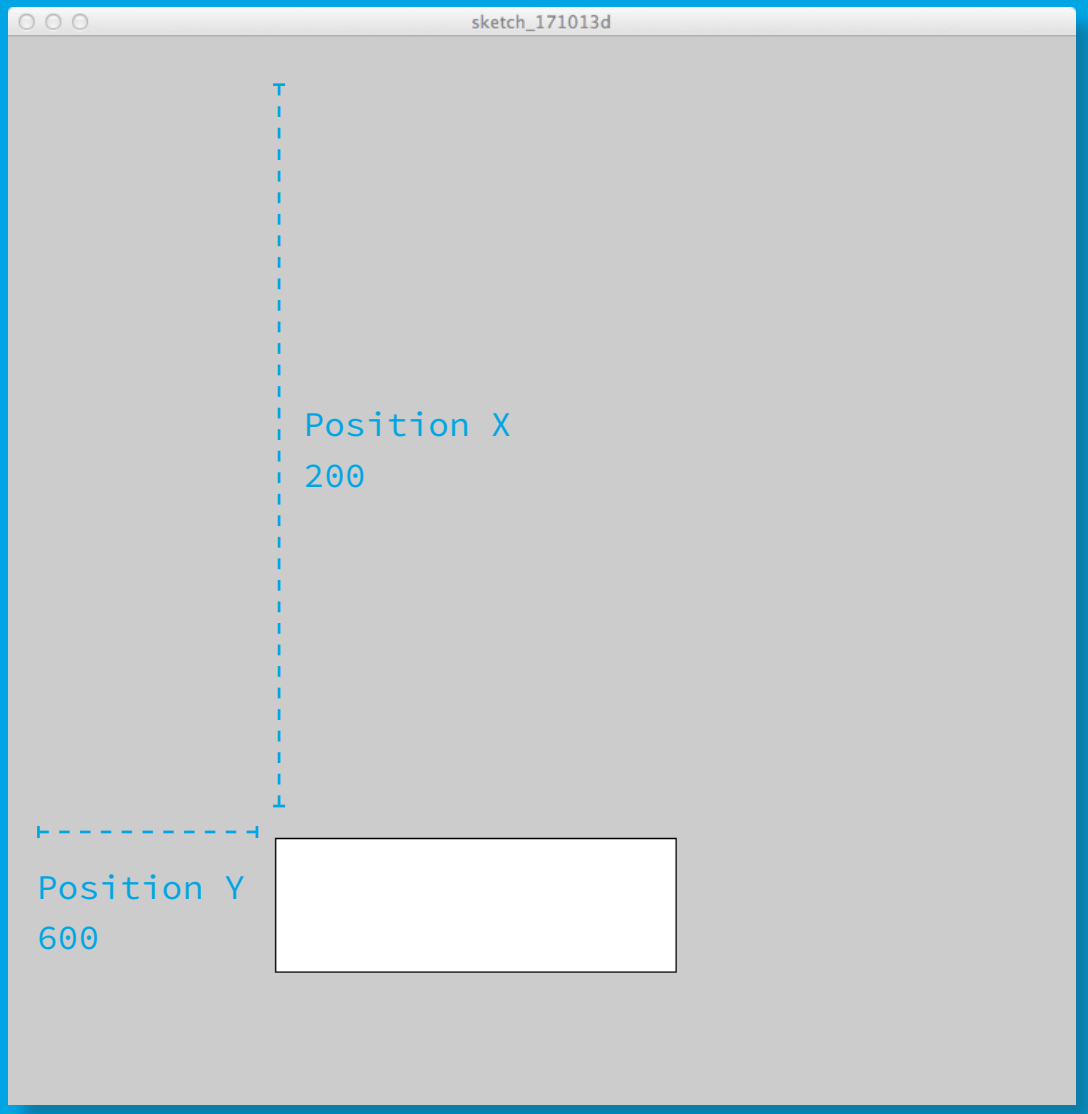
Grundformen

ellipse(Position X, Position Y, Breite, Höhe);




```
size(800,800);
ellipse(400,400,250,250);
```

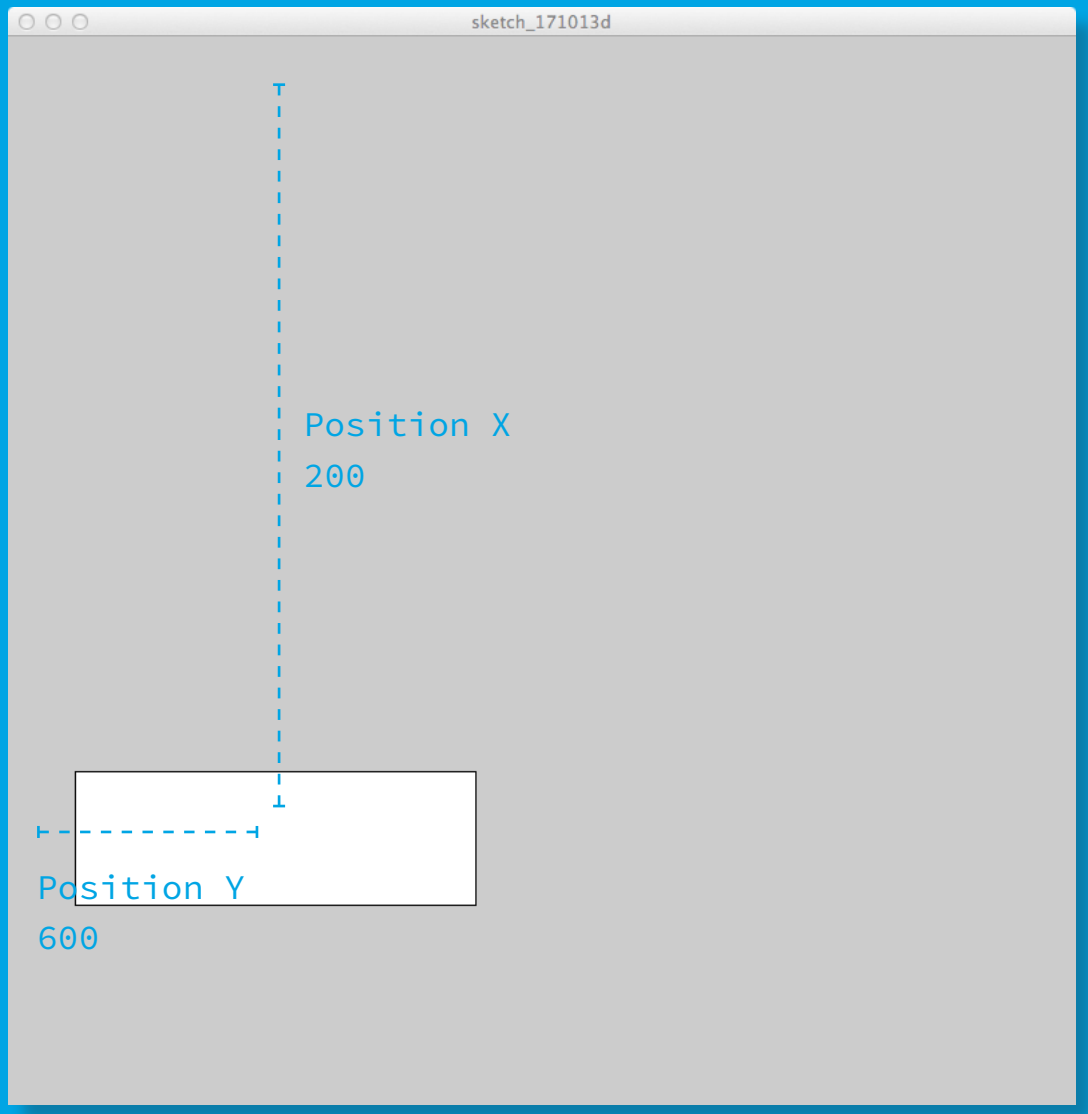
rect(Position X, Position Y, Breite, Höhe);



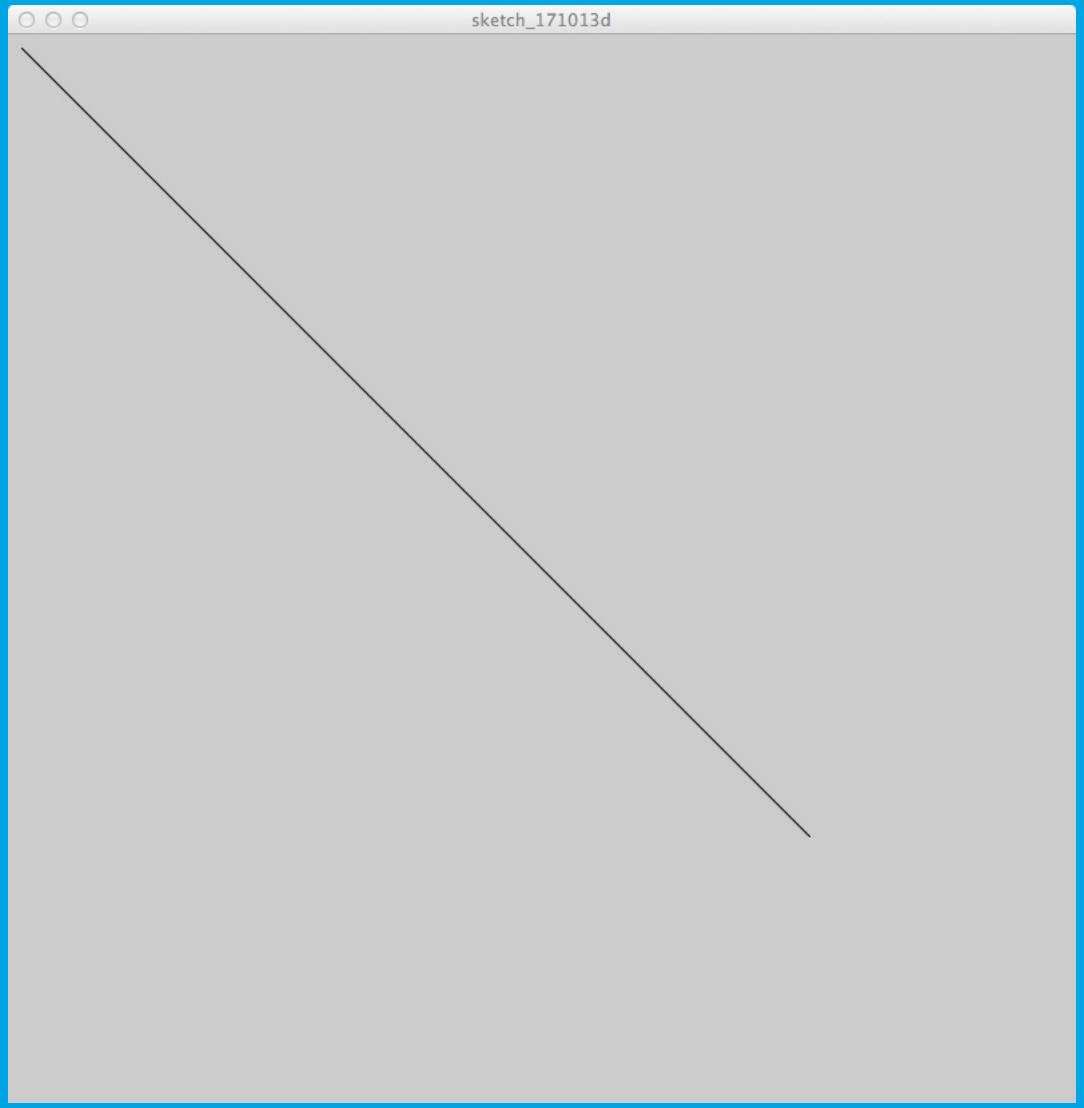

```
size(800,800);
rect(200,600,300,100);
```

rectMode(modus); CENTER, CORNER ...

```
size(800,800);
rectMode(CENTER);
rect(200,600,300,100);
```

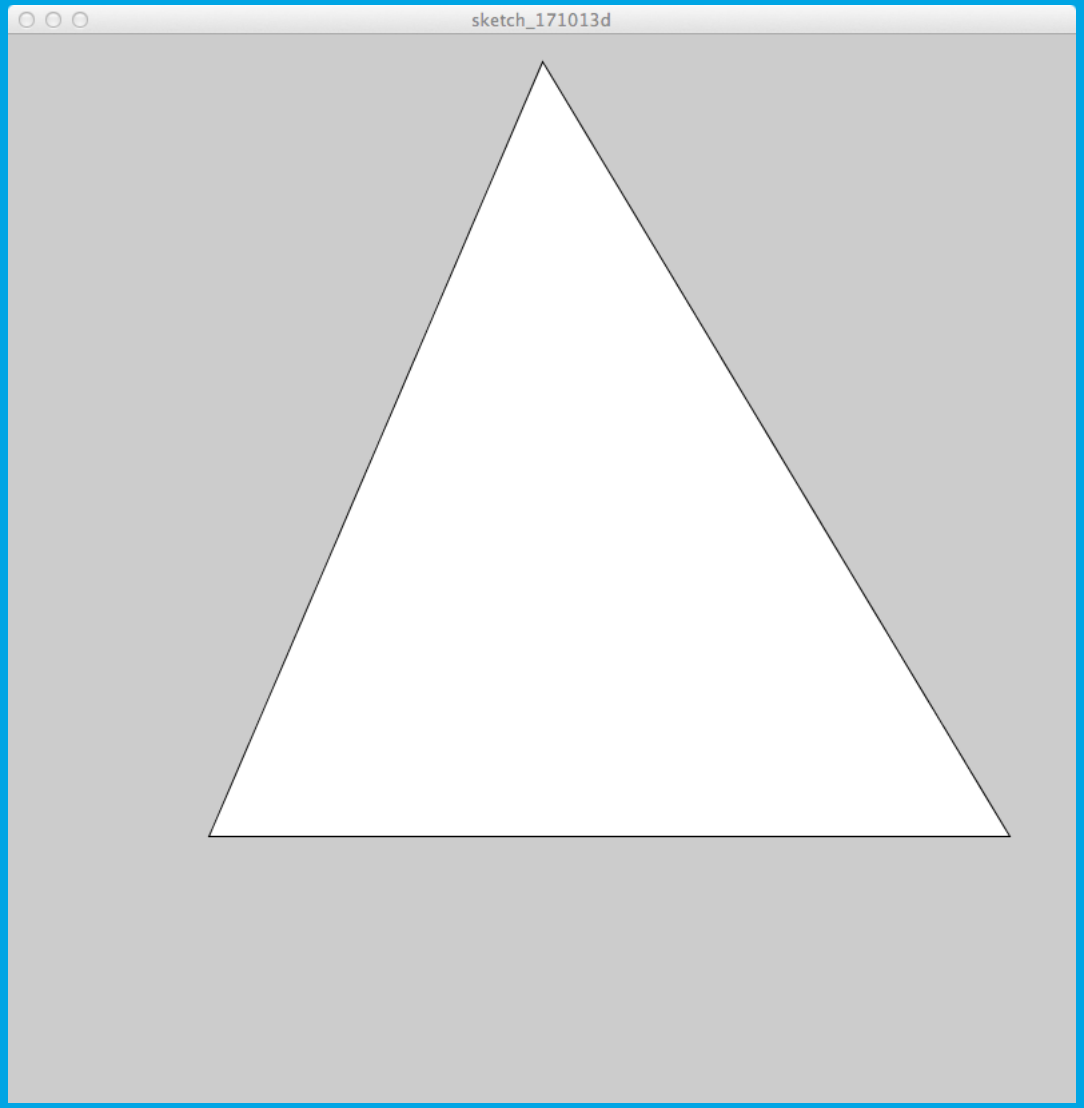


line(Start X, Start Y, Ende X, Ende Y);




```
size(800,800);
line(10,10,600,600);
```

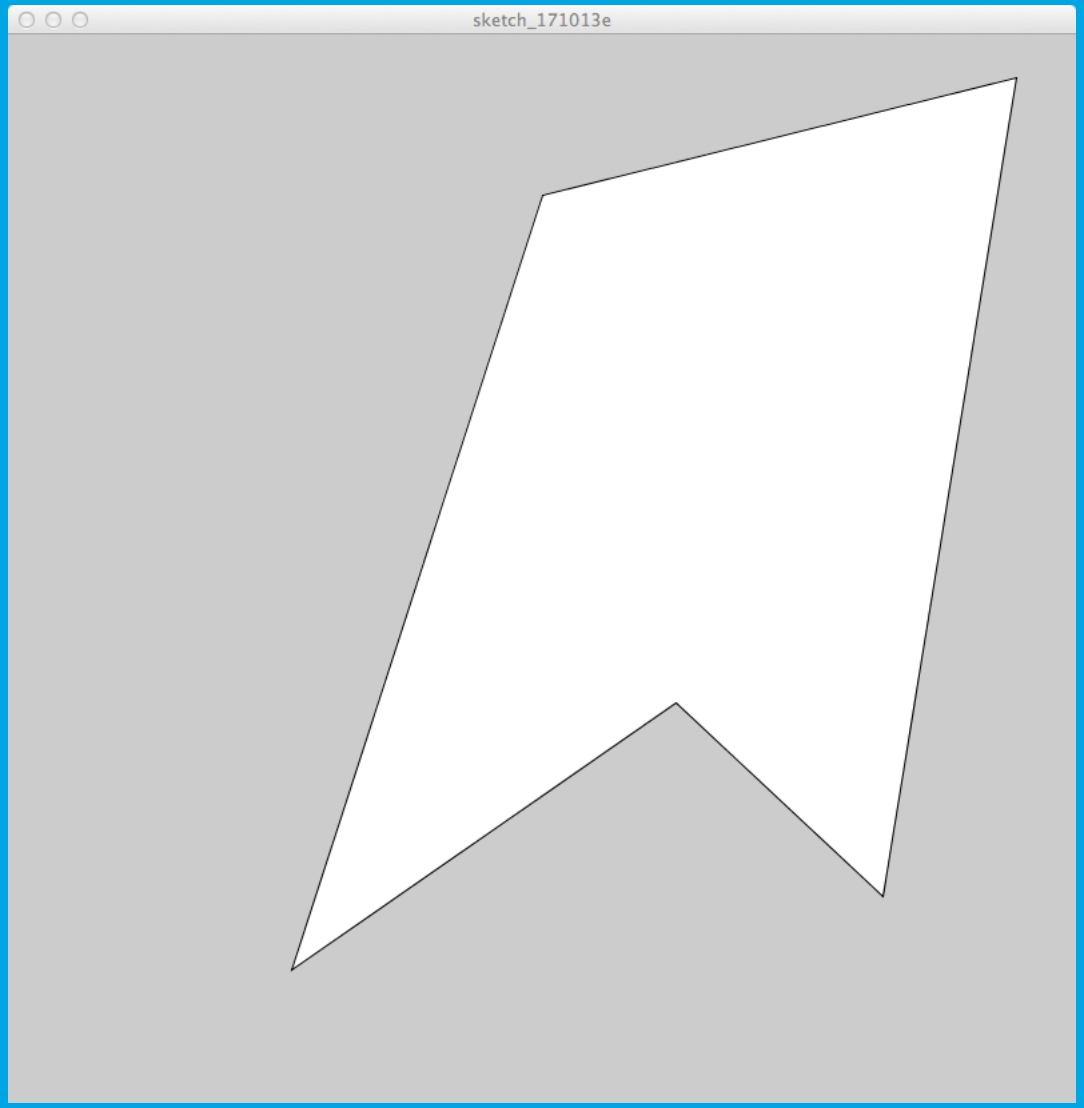
triangle(Punkt1 X, Punkt1 Y, Punkt2 X, Punkt2 Y, Punkt3 X, Punkt3 Y);

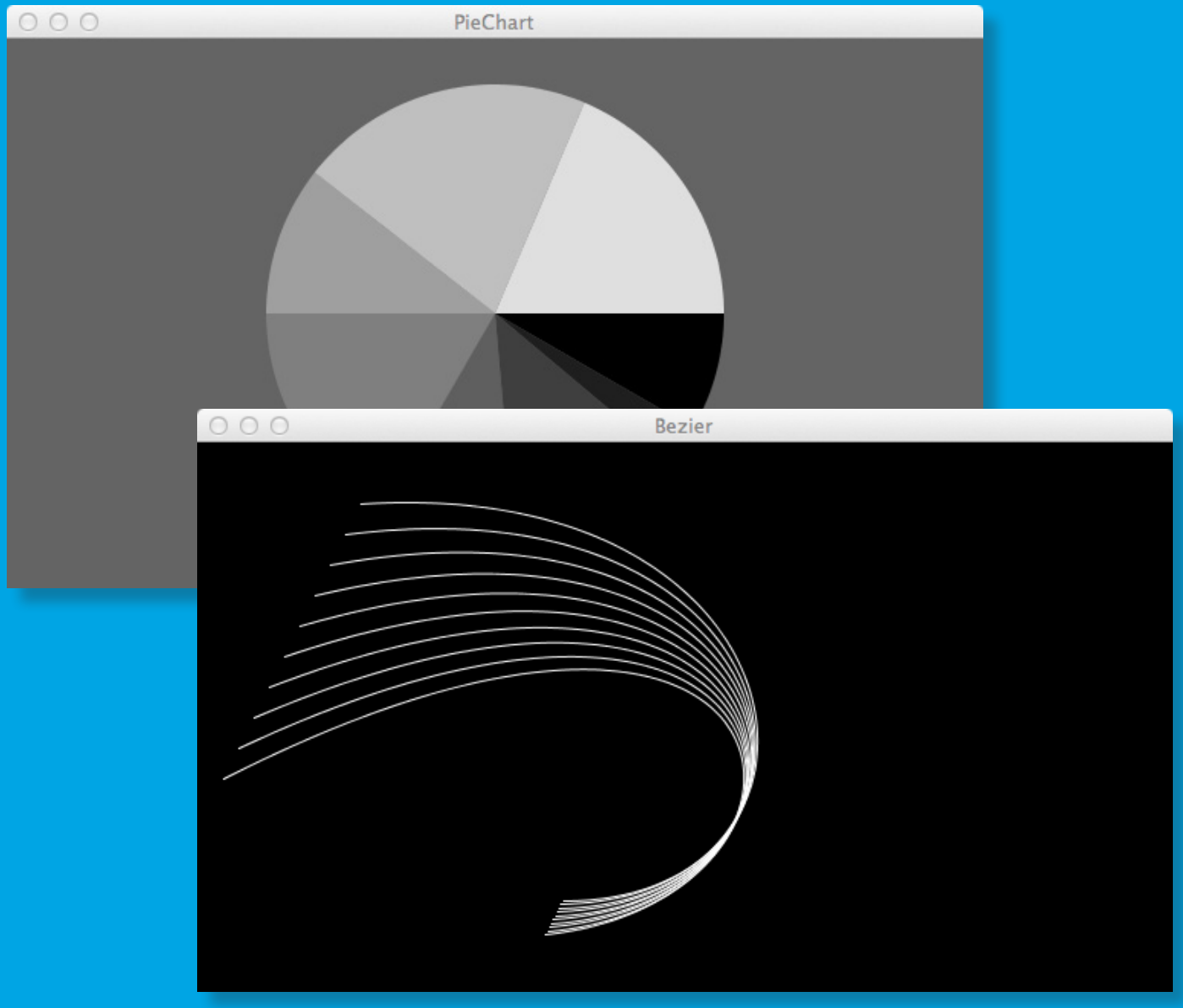



```
size(800,800);
triangle(150,600,750,600,400,20);
```

beginShape(); vertex(Position X, Position Y); endShape(CLOSE);

```
size(800, 800);
beginShape();
vertex(755, 32);
vertex(400, 120);
vertex(212, 700);
vertex(500, 500);
vertex(655, 645);
endShape(CLOSE);
```





Weitere Grundformen

arc();
bezier();
curve();
point();
...
...

<- Examples

Farben

RGB (alle Werte 0-255)

Red, Green, Blue

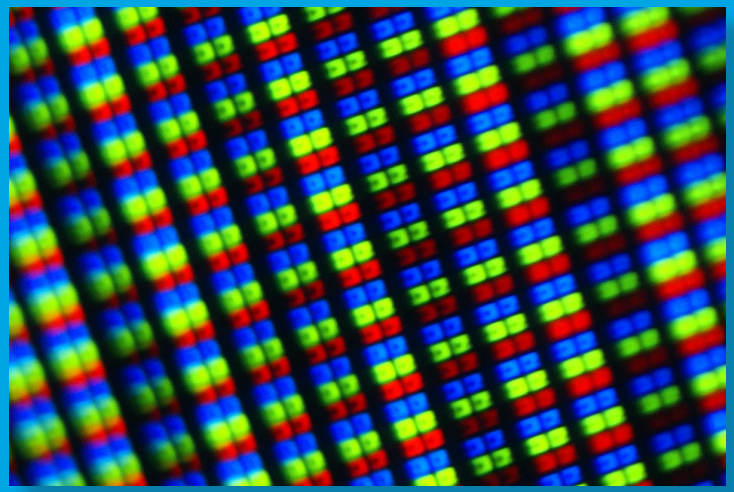
255,0,0

0,255,0

0,0,255

255,255,255

0,0,0



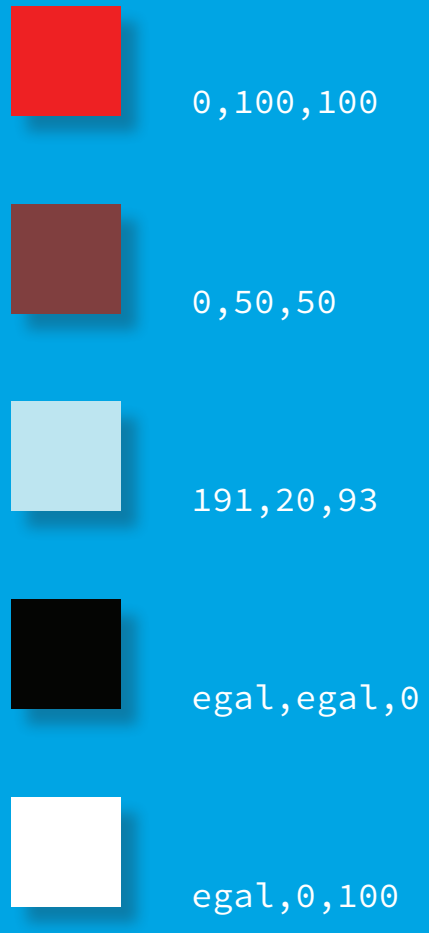
65,250,224

255,0,153

140,113,152

HSB, Hue, Saturation, Brightness

```
colorMode(HSB, 360, 100, 100);
```

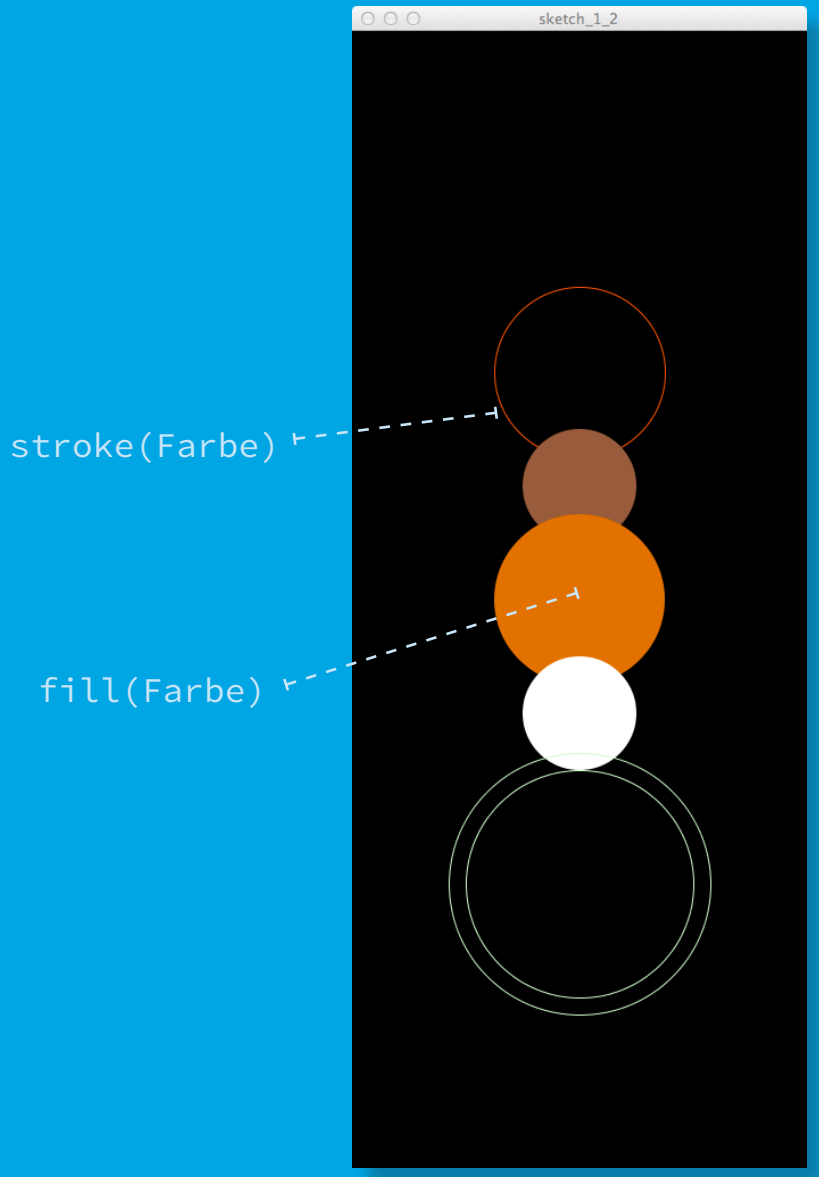


Hue -> Farbkreis 0-360°

Saturation und Brightness -> 0-100 %

background(Farbe); fill(Farbe); stroke(Farbe); fill(); noFill();

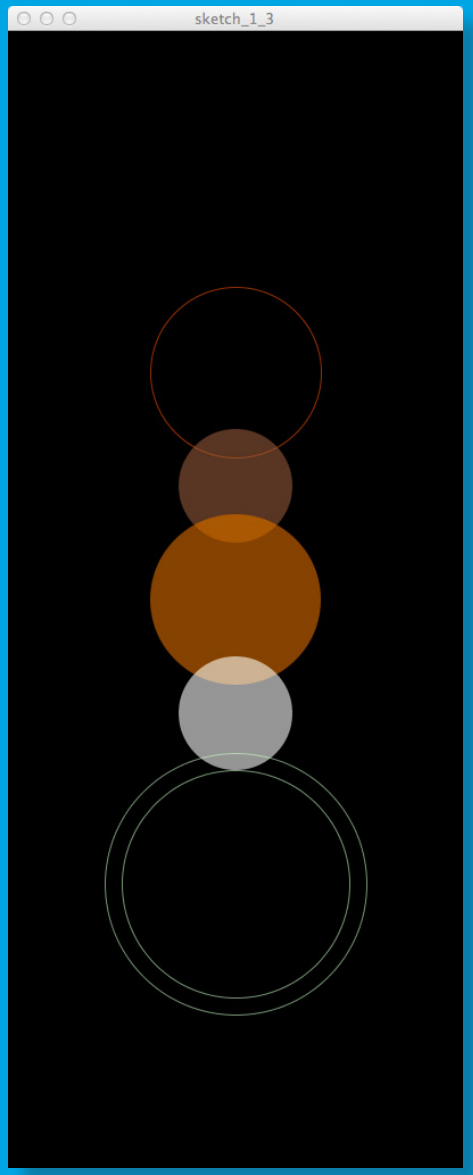
```
size(400, 1000);
colorMode(HSB, 360, 100, 100);

background(0, 0, 0);

noFill();
stroke(20, 200, 200);
ellipse(200, 300, 150, 150);

fill(20, 60, 60);
noStroke();
ellipse(200, 400, 100, 100);

fill(30, 100, 90);
ellipse(200, 500, 150, 150);

fill(0, 0, 100);
ellipse(200, 600, 100, 100);

noFill();
stroke(120, 20, 100);
ellipse(200, 750, 230, 230);
ellipse(200, 750, 200, 200);
```



sketch_1_2

Transparenz -> 4. Wert in einer Farbe (Standart 0-255)

```
size(400, 1000);
colorMode(HSB, 360, 100, 100);

background(0, 0, 0);

noFill();
stroke(20, 200, 200, 150);
ellipse(200, 300, 150, 150);

fill(20, 60, 60, 150);
noStroke();
ellipse(200, 400, 100, 100);

fill(30, 100, 90, 150);
ellipse(200, 500, 150, 150);

fill(0, 0, 100, 150);
ellipse(200, 600, 100, 100);

noFill();
stroke(120, 20, 100, 150);
ellipse(200, 750, 230, 230);
ellipse(200, 750, 200, 200);
```



sketch_1_3

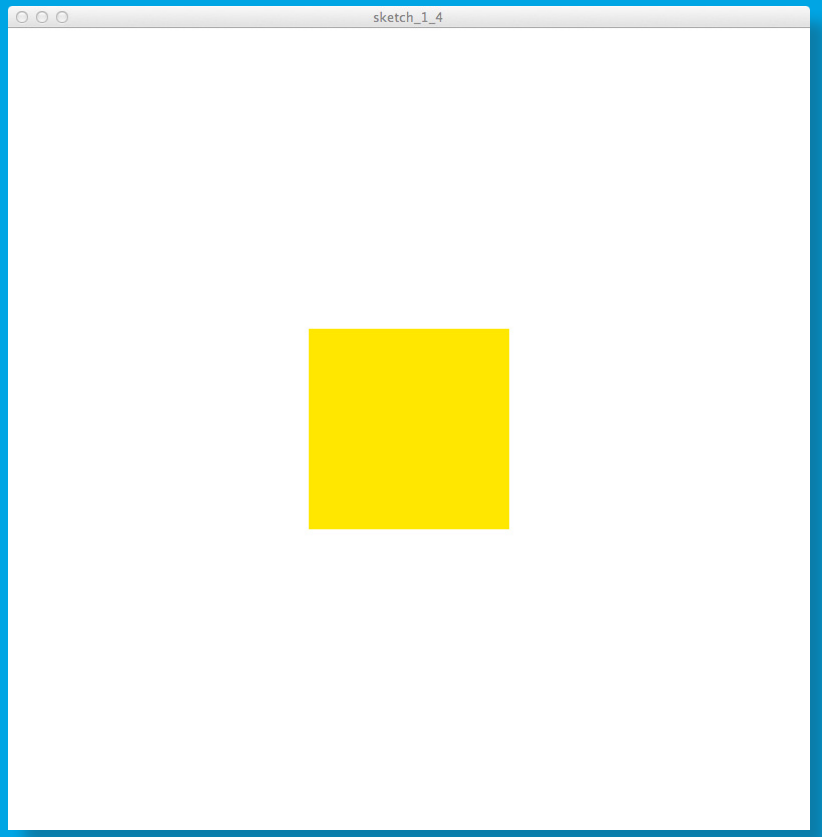
Der Flow,
void() & setup()

Dieser Code läuft, von oben nach unten, genau einmal durch!

Animationen und Interaktion in diesem Code nicht möglich.

```
size(800,800);
colorMode(HSB, 360, 100, 100);

background(0,0,100);

noStroke();
rectMode(CENTER);

fill(55, 255, 120);
rect(400,400, 200, 200);
```

sketch_1_4



Processing Syntax

code(){ ... }

Abschnitt

Inhalt

Abschnitte sind meistens eingerückt.

Mit cmd/strg + T kann man im Editor den Code automatisch formatieren.

```
code(){
    size(800,800);
    colorMode(HSB, 360, 100, 100);
    background(0,0,100);
    noStroke();
    rectMode(CENTER);
    fill(55, 255, 120);
    rect(400,400, 200, 200);
}
```

Code in Processing hat zwei grundlegende Abschnitte:

Dieser Block wird einemal!!! von oben nach unten durchlaufen.
z.b. size() oder colorMode() machen Sinn in void setup()

```
void setup() {
...
...
...
}
```

1x

void draw() läuft durch und fängt wieder oben an.

```
void draw() {
...
...
...
}
```

∞

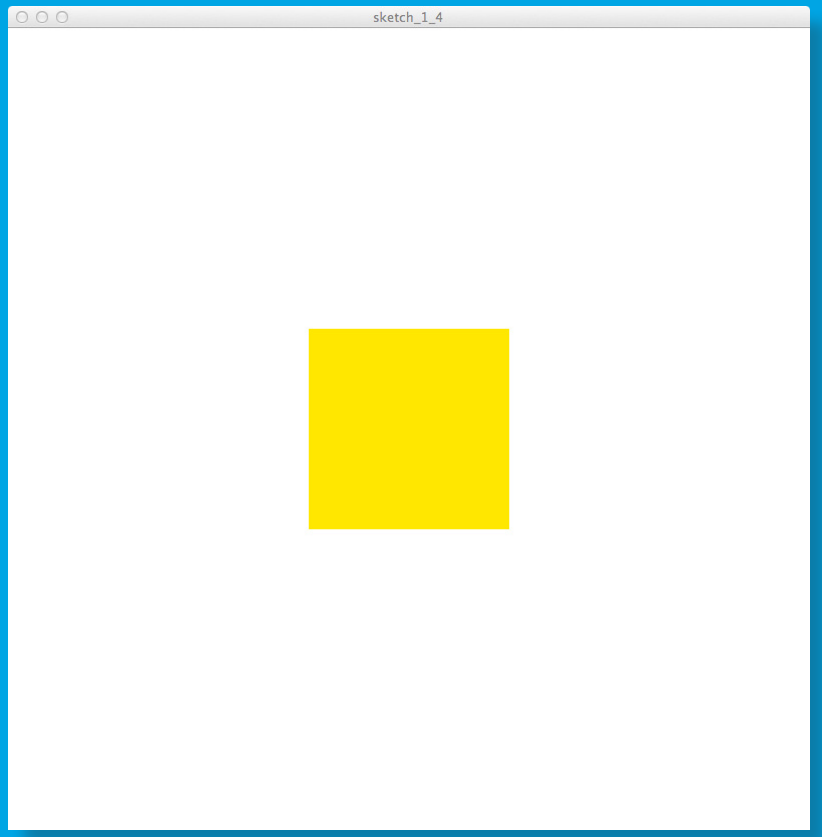
Der gleiche Code aufgeteil in setup() und draw().

Das Programm läuft jetzt in Echtzeit.

Aber weil die Werte in draw() nicht varieren, sieht es immer gleich aus.

```
void setup() {
  size(800, 800);
  colorMode(HSB, 360, 100, 100);

  background(0, 0, 100);
}

void draw() {
  noStroke();
  rectMode(CENTER);

  fill(55, 255, 120);
  rect(400, 400, 200, 200);
}
```

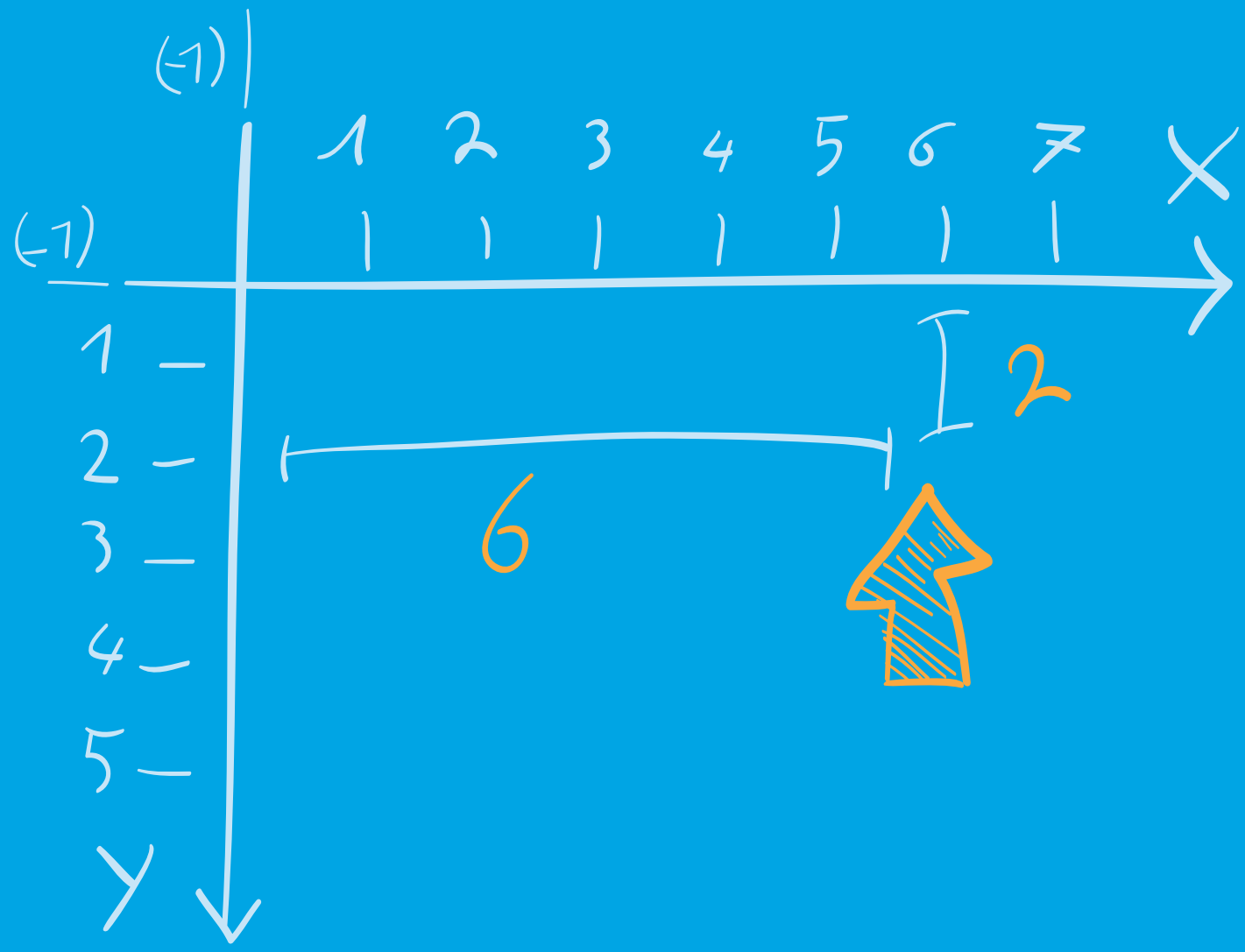
sketch_1_5



mouseX und mouseY sind dynamische Variablen die immer die aktuelle Mausposition speichern.

mouseX ist **6**

mouseY ist **2**

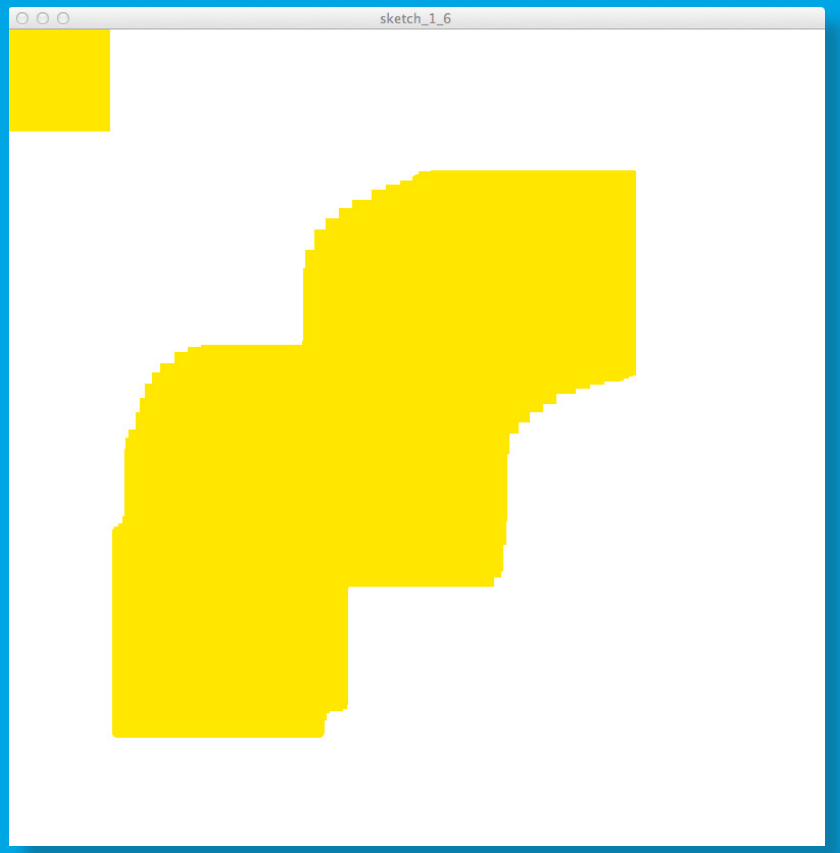


Weil die Werte für mouseX und mouseY sich immer ändern, und void draw() sich wiederholt, läuft das Programm jetzt dynamisch.

Variante: background() in void draw() verschieben...

```
void setup() {
  size(800, 800);
  colorMode(HSB, 360, 100, 100);

  background(0, 0, 100);
}

void draw() {
  noStroke();
  rectMode(CENTER);

  fill(55, 255, 120);
  rect(mouseX, mouseY, 200, 200);
}
```

sketch_1_6



Andere void Blöcke.... keyPressed() und mousePressed()

```
void setup() {
  size(800, 800);
  background(0, 0, 100);
}


void draw() {

}


void keyPressed() {
  background(0, 100, 100);
}


void mousePressed() {
  background(200, 50, 50);
}
```

sketch_1_7

// comment

/*...*/

```
void setup() {
//ich bin ein Kommentar
//und werde ignoriert
...
...
...

}
```

Variablen
int, float

Variablen sind Behälter die einen bestimmten Werte, z.b. Zahlen, enthalten. Sie haben einen Namen und verlinken an eine Position im Arbeitsspeicher, wo dieser Wert abgelegt ist.

`mouseX` , `mouseY` -> Variablen für die Position der Maus

`width` und `height` -> Variablen mit der Breite und Höhe des Fensters

```
int
12,
276
-231
1

float
0.3
0.322
12.377
-0.5
```

typ name;

int oder float

irgendein Name, z.b. blub

name = wert;

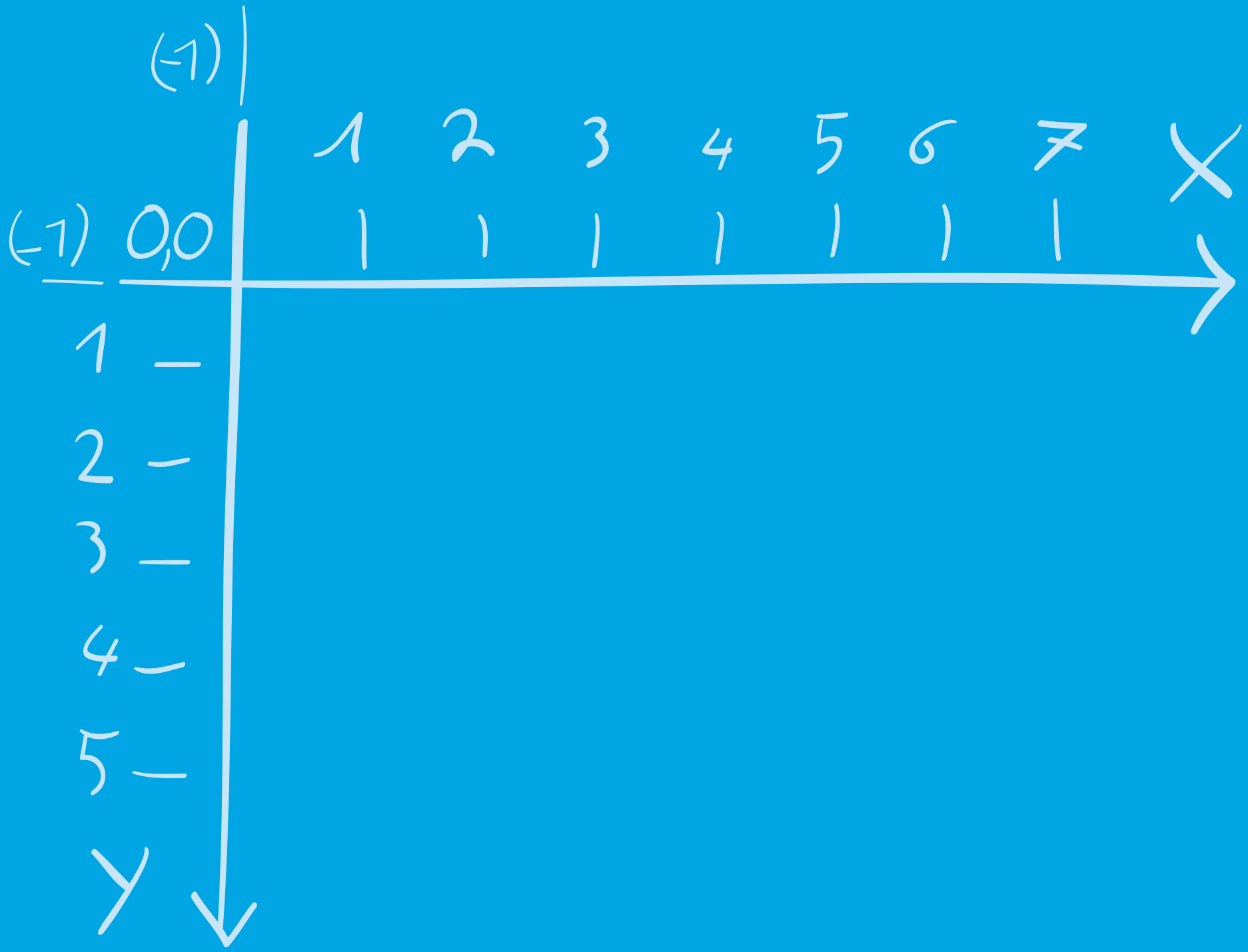
10 für int, 1.0 für float

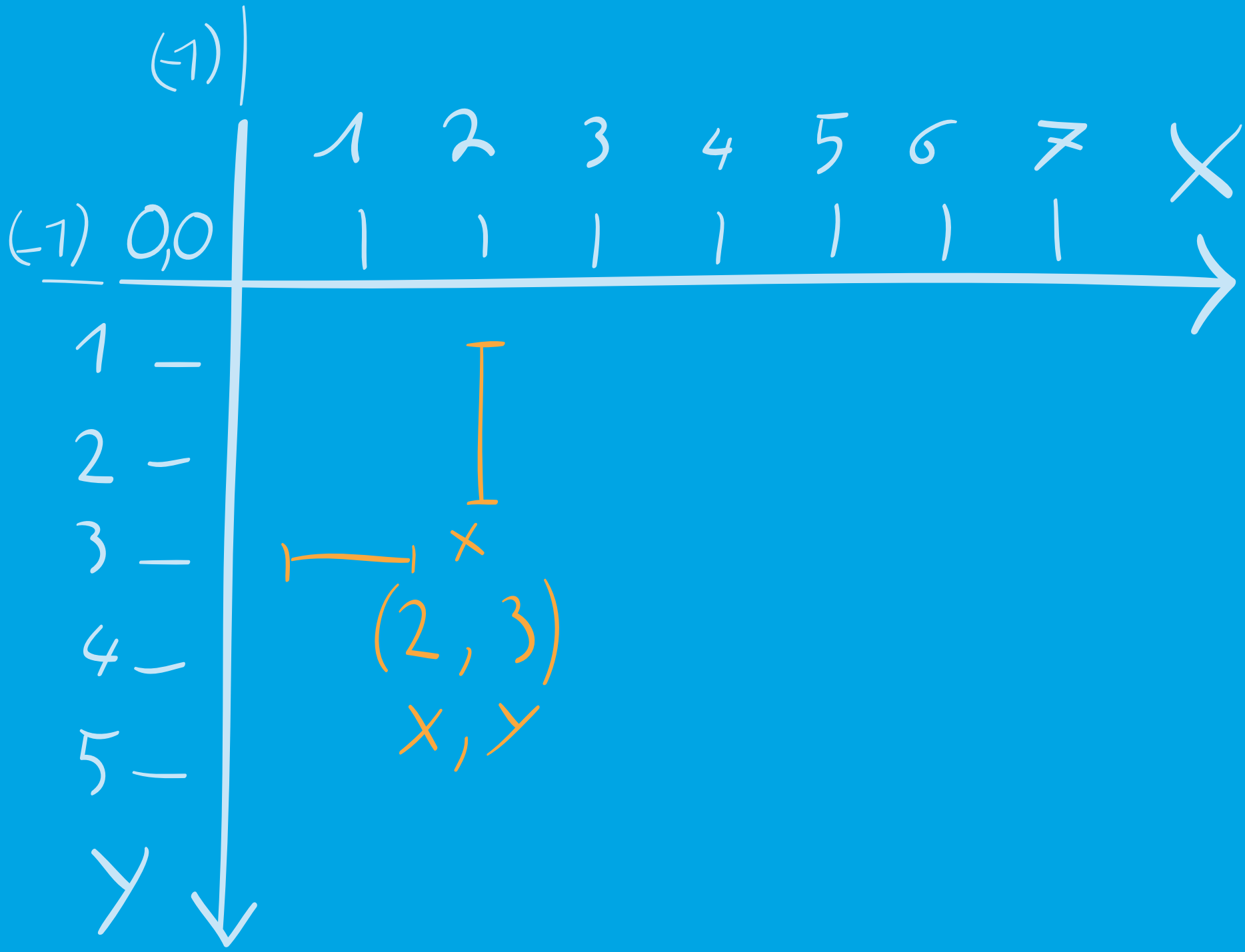
```
float green;

void setup() {
green = 20.3;
background(0,green,0);
}
```

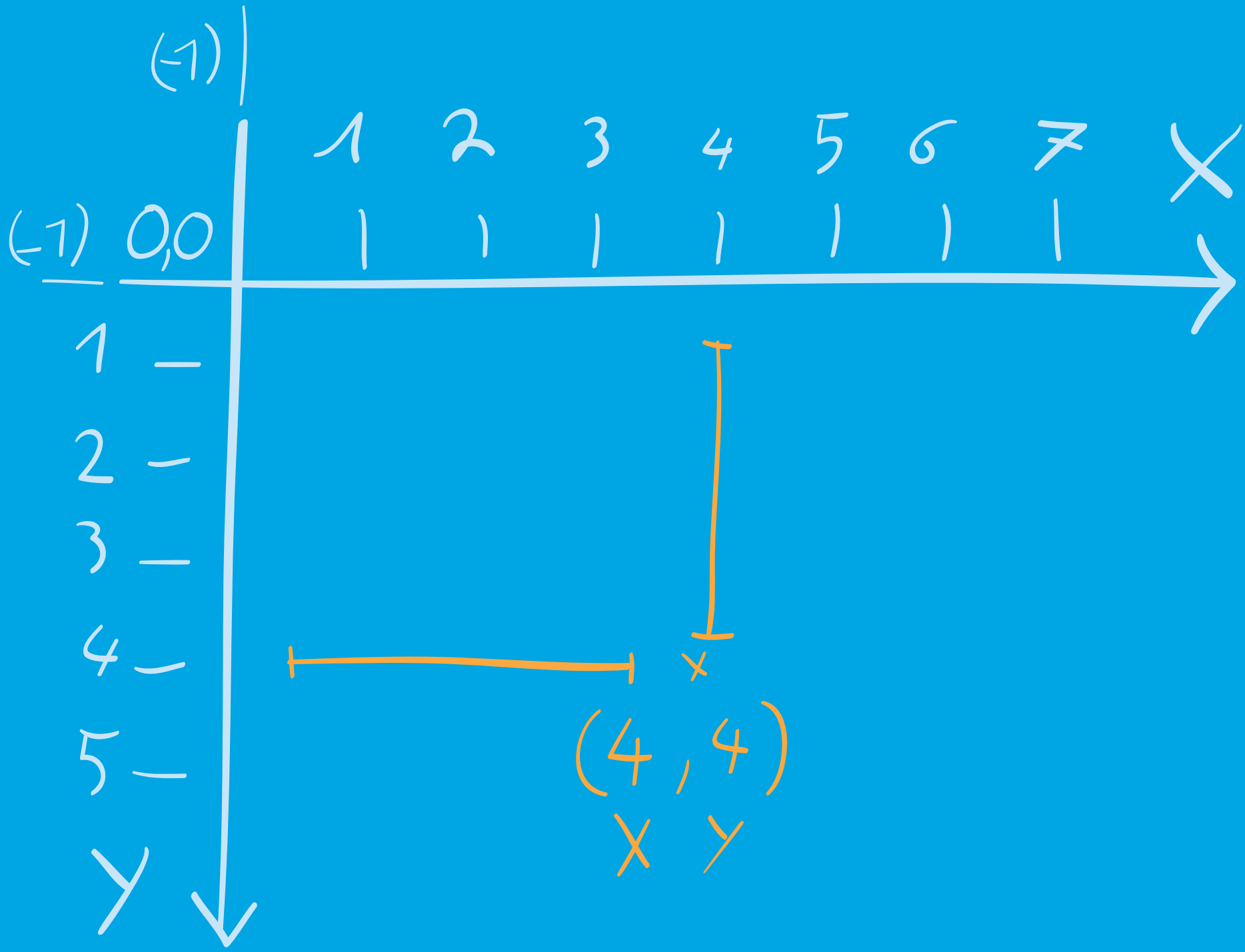
```
//es geht auch so...
float green = 20.5;

void setup() {
background(0,green,0);
}
```

Processing Syntax

+ - * / ++ --


(-1)
1 2 3 4 5 6 7 X
(-1) 0,0
1 1 1 1 1 1 1
1 —
2 —
3 —
4 —
5 —
Y


(-1)
1 2 3 4 5 6 7 X
(-1) 0,0
1
2
3
4
5
Y
(2, 3)
x, y


(-1)
1 2 3 4 5 6 7 X
(-1) 0,0
1
2
3
4
5
Y
(4,4)
X Y

```
befehl(_,_,_);
```

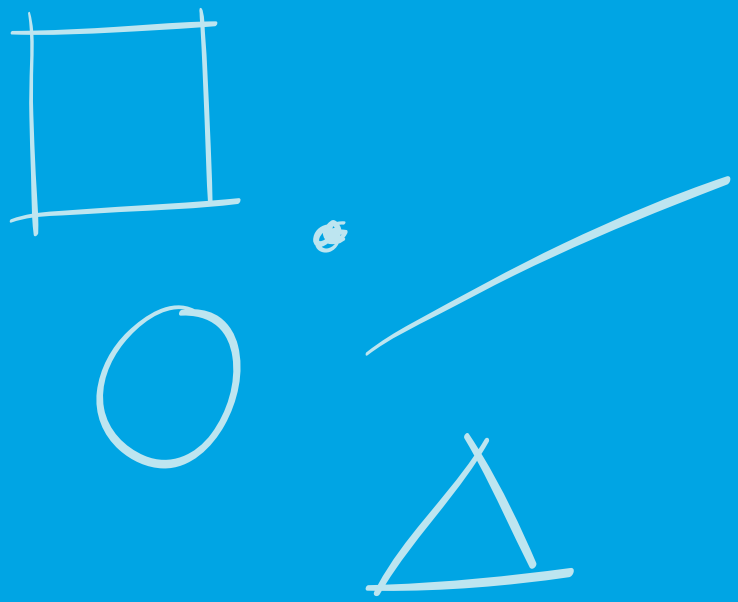
Funktion

Argumente

Anweisungen / Funktionen

Grundformen:

```
rect(x, y, Breite, Höhe);
ellipse(x, y, Breite, Höhe);
point(x, y);
line(x1, y1, x2, y2);
triangle(x1, y1, x2, y2, x3, y3);
```



Zeichnen:

```
size(Breite, Höhe);
colorMode(...);
fill(R, G, B);
noFill();
stroke(R, G, B);
noStroke();
background(R, G, B);
```

Code Block

```
code(){ ... }
```

Abschnitt

Inhalt

Code Block

```
void setup() {
...
}
```

```
void mousePressed() {
...
}
```

```
void draw() {
...
}
```

```
void keyPressed() {
...
}
```

Code in Processing hat zwei grundlegende Abschnitte:

Dieser Block wird einemal!!! von oben nach unten durchlaufen.
z.b. size() oder colorMode() machen Sinn in void setup()

```
void setup() {
...
...
...
}
```

1x

void draw() läuft durch und fängt wieder oben an.

```
void draw() {
...
...
...
}
```

∞

// comment

/*...*/

```
void setup() {
//ich bin ein Kommentar
//und werde ignoriert
...
...
...

}
```

```
typ name;
```

int oder float

irgendein Name, z.b. blub

```
name = wert;
```

10 für int, 1.0 für float

Variablen

int: Ganze Zahlen

```
int positionX;
int breite;
int tempo;


positionX = 10;
breite = 200;
tempo = -40;
```

float: Komma Zahlen

```
float positionX;
float breite;
float tempo;


positionX = 10.13;
breite = 200.99;
tempo = -40.5;
```

Allgemeine Variablen:

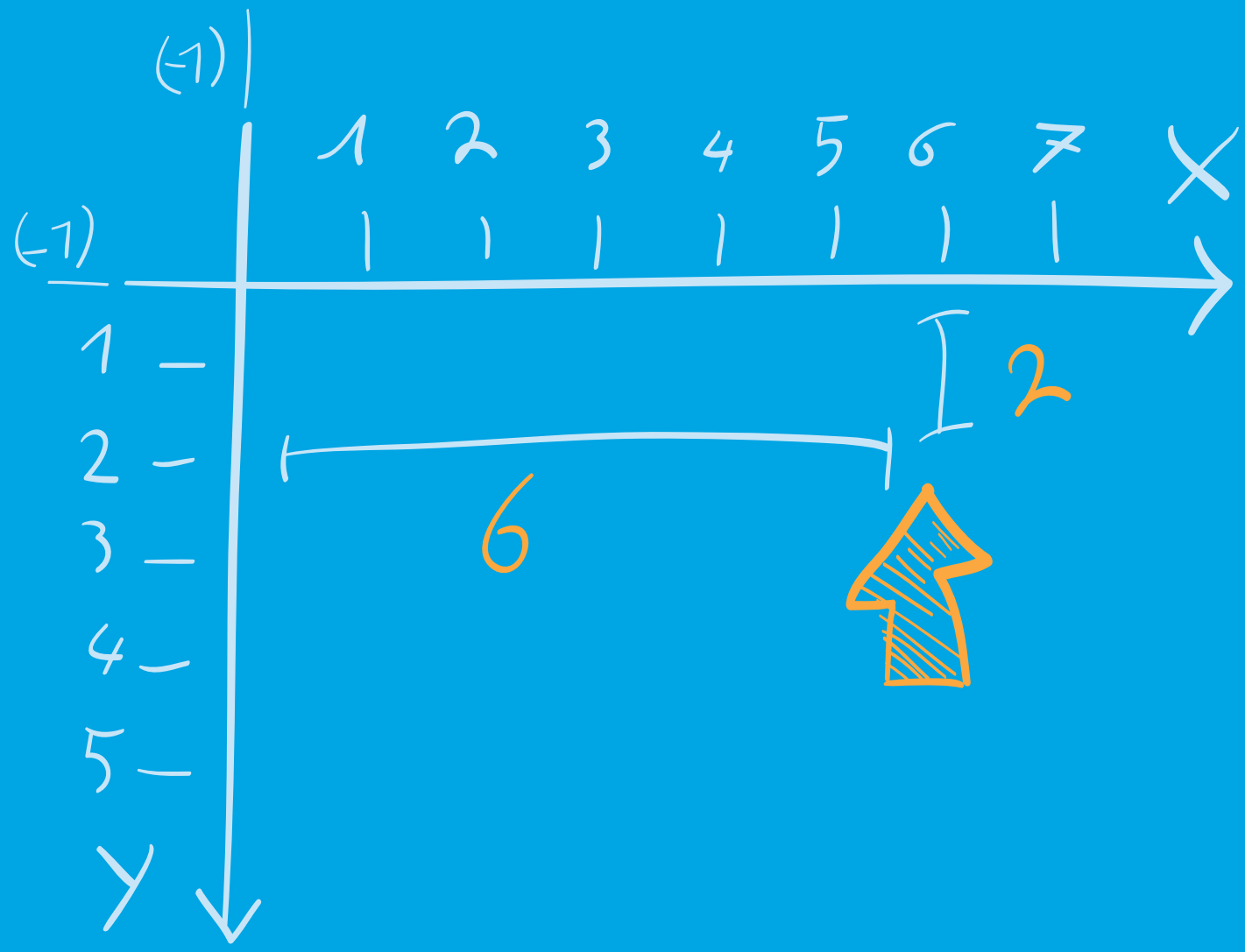
mouseX;
mouseY;
width;
height;

zb **int position = mouseX;**

mouseX und mouseY sind dynamische Variablen die immer die aktuelle Mausposition speichern.

mouseX ist **6**

mouseY ist **2**



Variablen Operationen

+, -, *, /, ++, --,

+=, -=, *=, /=

```
int tempo;
tempo = 10;

tempo = 10 + 10;  //20
tempo = 10 - 5;   //5
tempo = 10 * 3;   //30
tempo = 10 / 2;   //5
tempo++; //11 tempo = tempo + 1;
tempo--; //9  tempo = tempo - 1;
tempo *= 2 //20 tempo = tempo * 2;
tempo /= 2 //5 tempo = tempo/2;
```

Logik

```
if (...) {
  ...
}
```

```
if (boolean) {
...
}
```

Boolean

boolean sind true oder false, 1 oder 0.

```
boolean test;
test = true;

if (test == true) {
...
}

if (test == false) {
...
}
```

Boolean

boolean sind true oder false, 1 oder 0.

```
boolean test;
test = false;

if (test == true) {
...
}

if (test == false) {
...
}
```

Boolean Operatoren

<, >, <=, >=, ==, !=

```
int tempo;
tempo = 10;

(tempo < 22) -> TRUE
(tempo > 20) -> FALSE
(tempo <=10) -> TRUE
(tempo >= 1) -> TRUE
(tempo == 10)-> TRUE
(tempo == 8) -> FALSE
(tempo != 1) -> TRUE
```

Logikoperatoren

und &&

oder ||

Logikoperatoren

//beide Bedingungen müssen erfüllt sein

```
if ( (boolean1) && (boolean2) ) {
...
}
```

//eine Bedingung muss erfüllt sein

```
if ( (boolean1) || (boolean2) ){
...
}
```

if - else Varianten

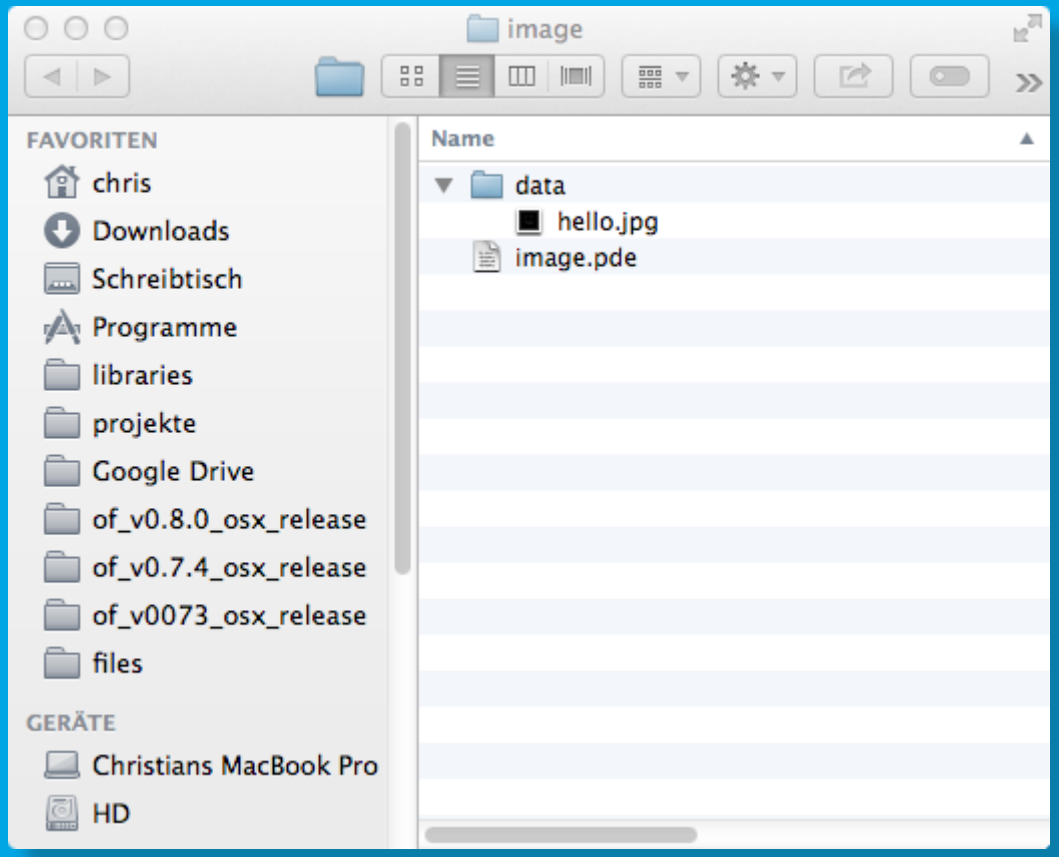
```
if () {
   ...
} else {
   ...
}
```

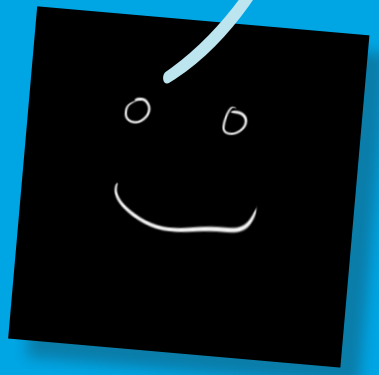
```
if () {
   ...
} else if () {
   ...
}
```

```
if () {
   ...
} else if () {
   ...
} else {
   ...
}
```

Images

Data Folder
cmd/strg + k





hello.jpg

Print Line

```
void draw() {
  //Gibt unten den Inhalt der Variable x aus
  println(x);
  //Erhöht x jeden Frame +1
  x++;
}
```

Done Saving.

```
81
82
83
84
85
86
6
```

println(...);

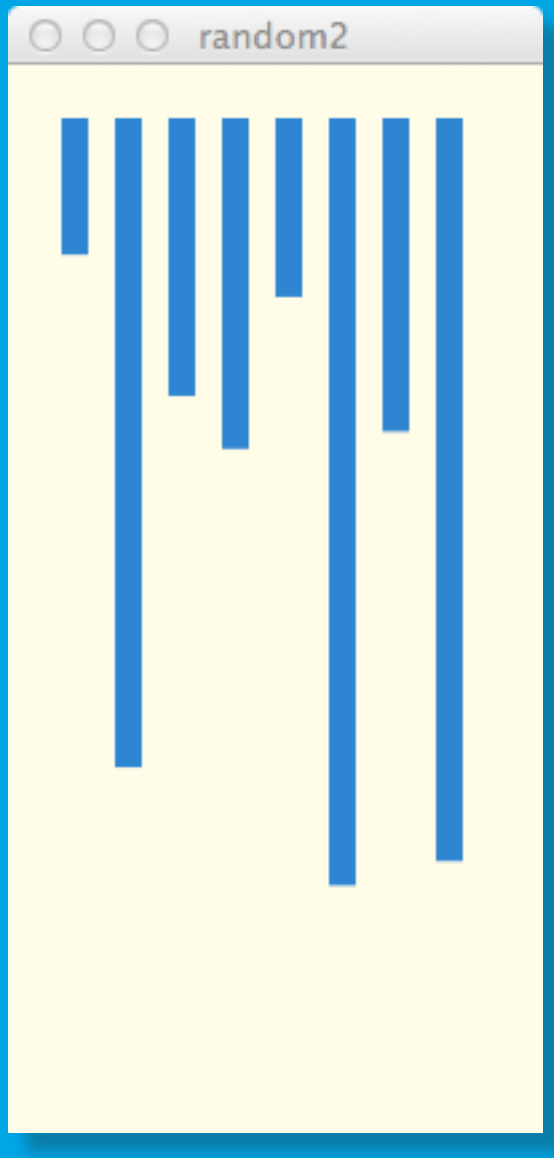
Zeigt in der Konsole den Inhalt einer Variable an

Random

Generiert eine Zufällige Zahl (float)

```
random(max);
```

```
random(min, max);
```



Arrays

Array

```
int b1 = 1;

int b2 = 21;

int b3 = 13;

int b4 = -9;

int b5 = 100;

...
```

Eine Liste von Variablen.
Z.b. für die Größe oder Position von verschiedenen Elementen

Array

```
int b1 = 1;
int b2 = 21;
int b3 = 13;
int b4 = -9;
int b5 = 100;
```

```
int[] liste = {1, 21, 13, -9, 100};
```

```
typ[]  name;
```

int, float,...

irgendein Name, z.b. blub

Array

```
int[] liste;
```

Array, Liste von Variablen

```
int tempo;
```

Einzelne Variable

Array

```
typ[] name = {x, x, x, x, x};
```

```
int[] liste = {12, 32, 12, 0, 1};
```

```
typ[] name = new int[x];
```

```
int[] liste = new int[5];
```

Alle 5 int's wäre so erstmal 0

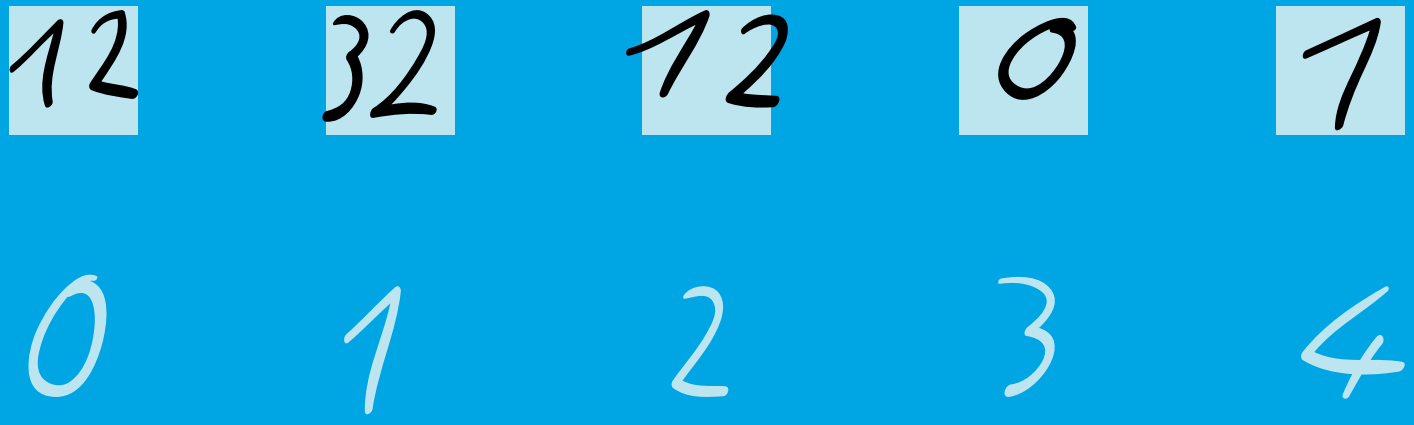
Array

```
int[] liste = {12, 32, 12, 0, 1};

liste[0]; // 12
liste[1]; // 32
liste[2]; // 12
liste[3]; // 0
liste[4]; // 1
```

5 Positionen und fängt immer bei 0 an!

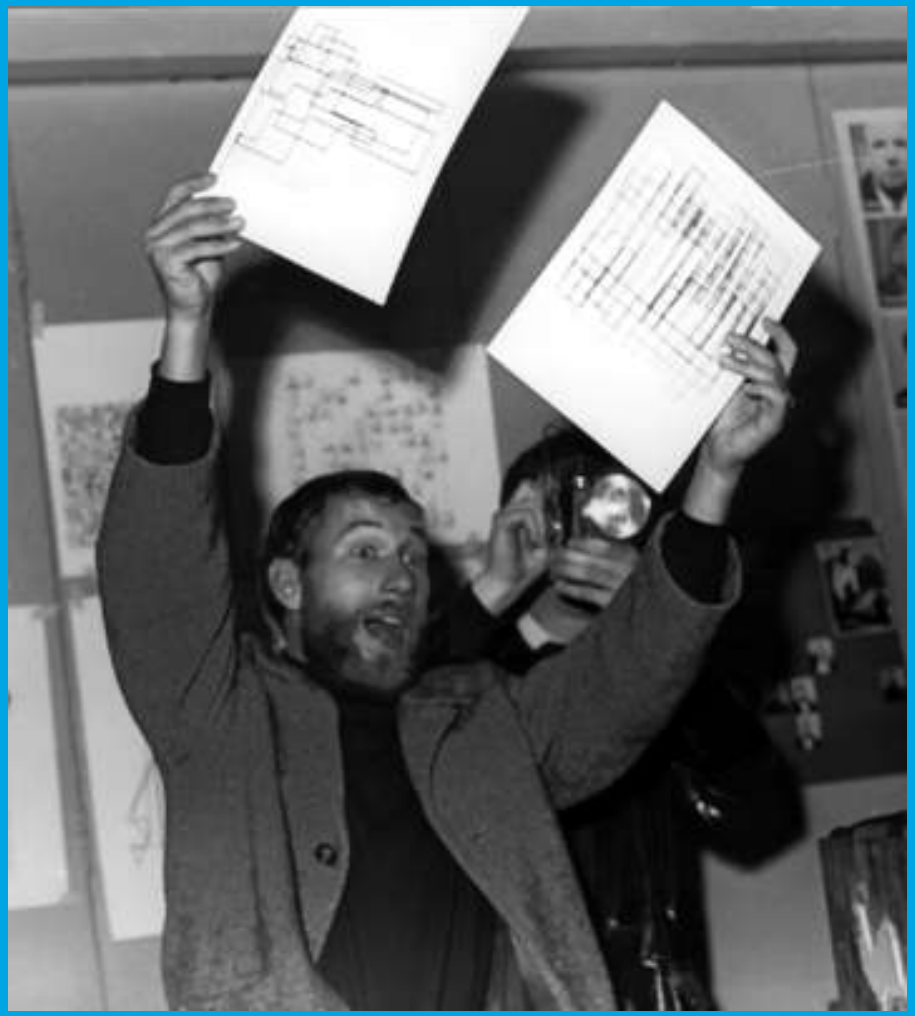
Array

```
int[] liste = {12, 32, 12, 0, 1};
```



```
float[] liste = new float[10];

for (int i = 0; i < liste.length; i++) {
liste[i] = random(100);
}


for (int i = 0; i < 10; i++) {
println(liste[i]);
}
```

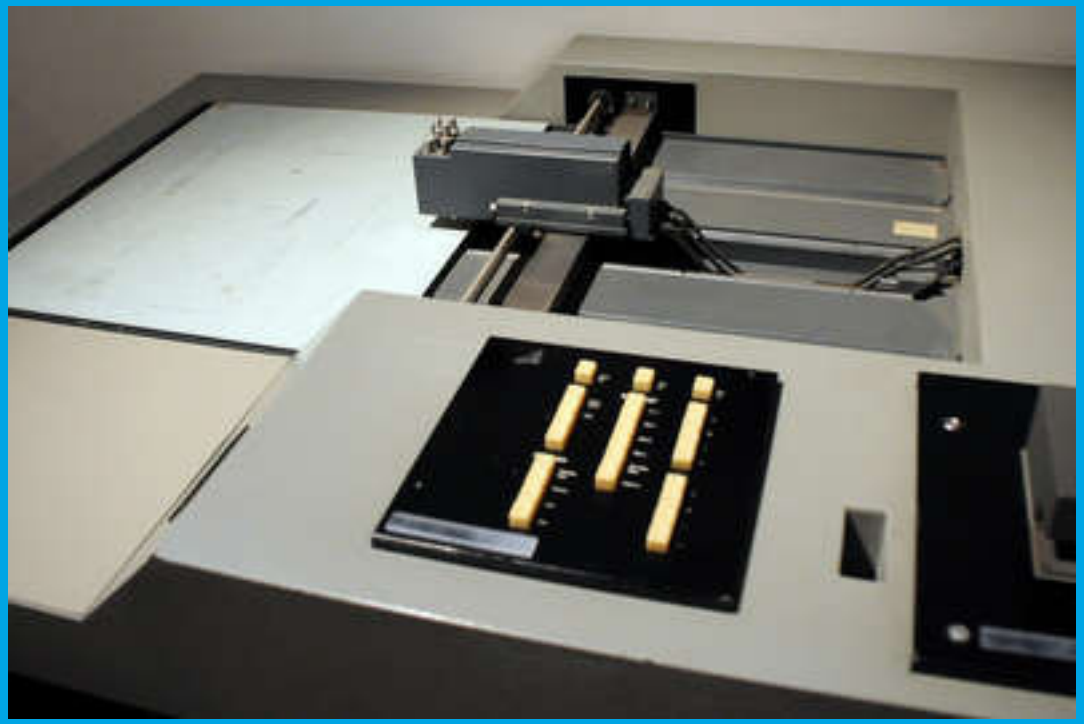
Frieder Nake



1938, Stuttgart

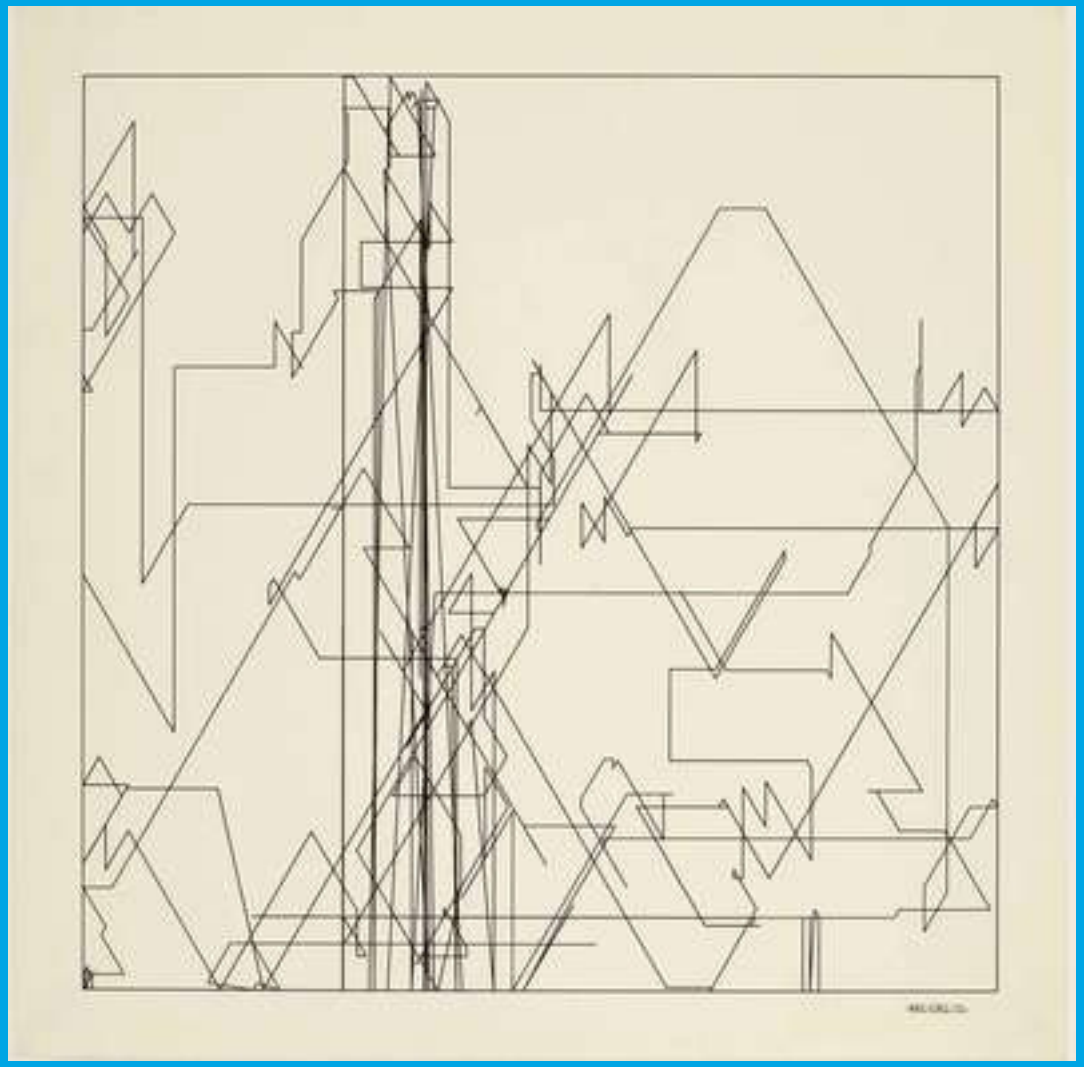
Mathematiker

1963 erste Computerzeichnungen

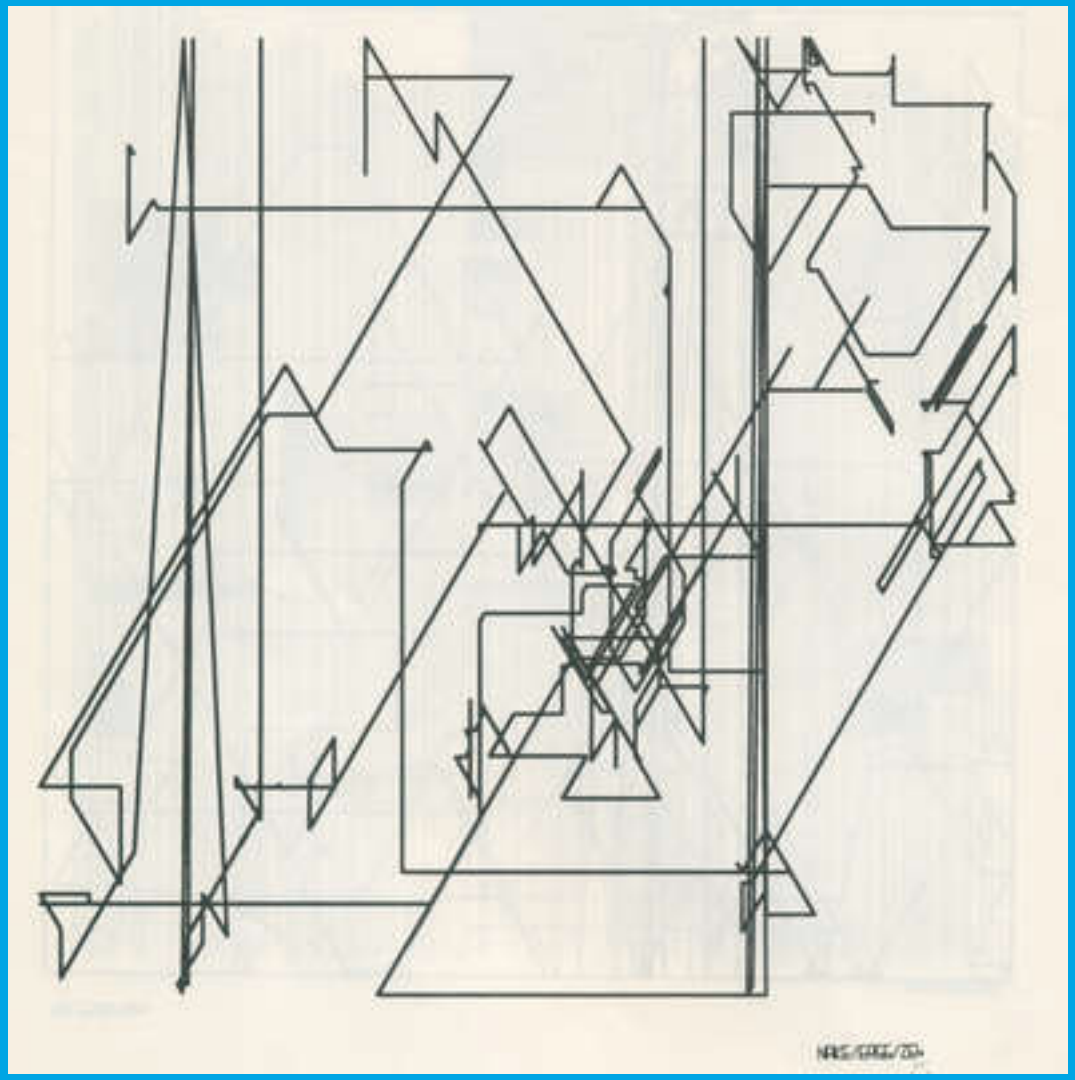


Zuse Z64 Graphomat 2

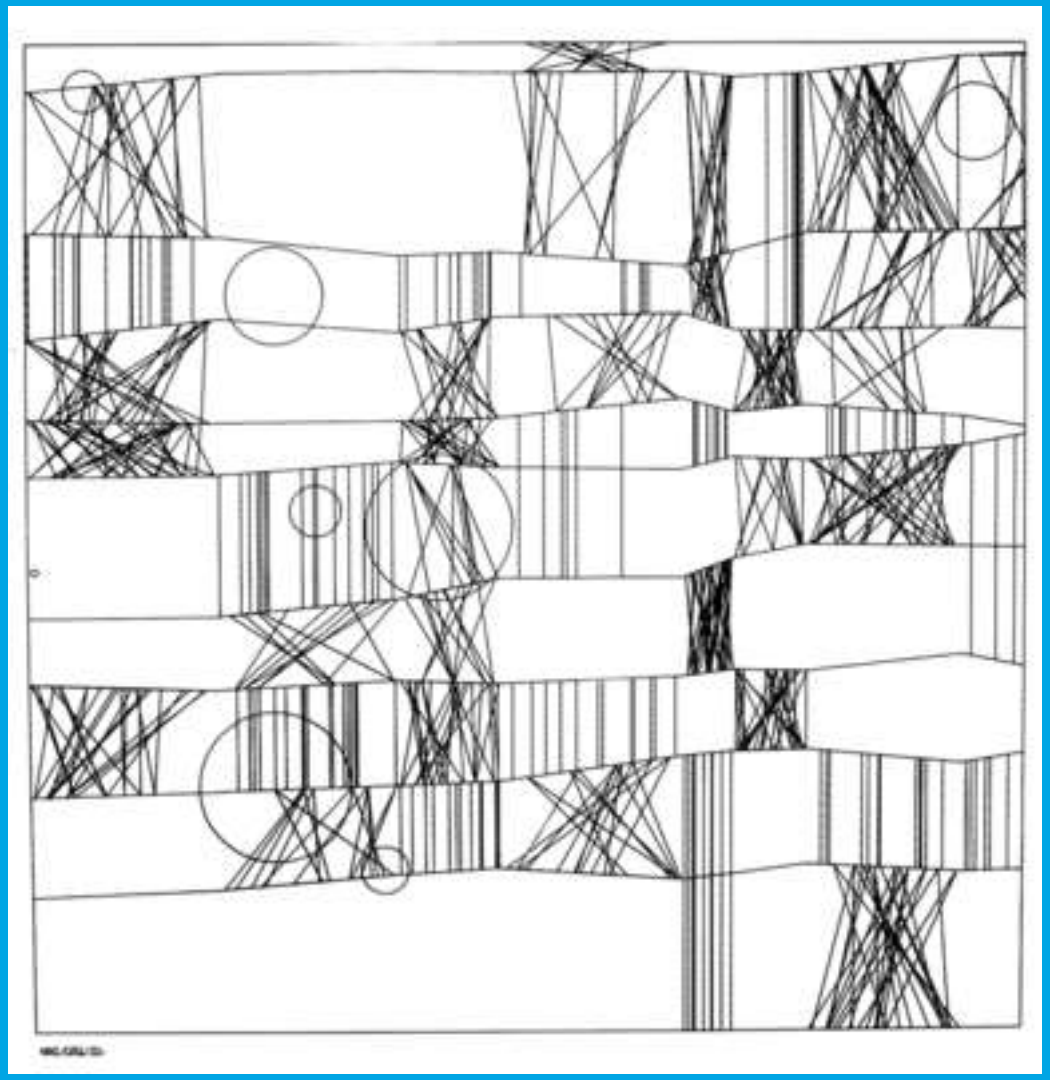
Computer from 1960's



13/9/65
Nr. 3 Zufälliger Polygonzug

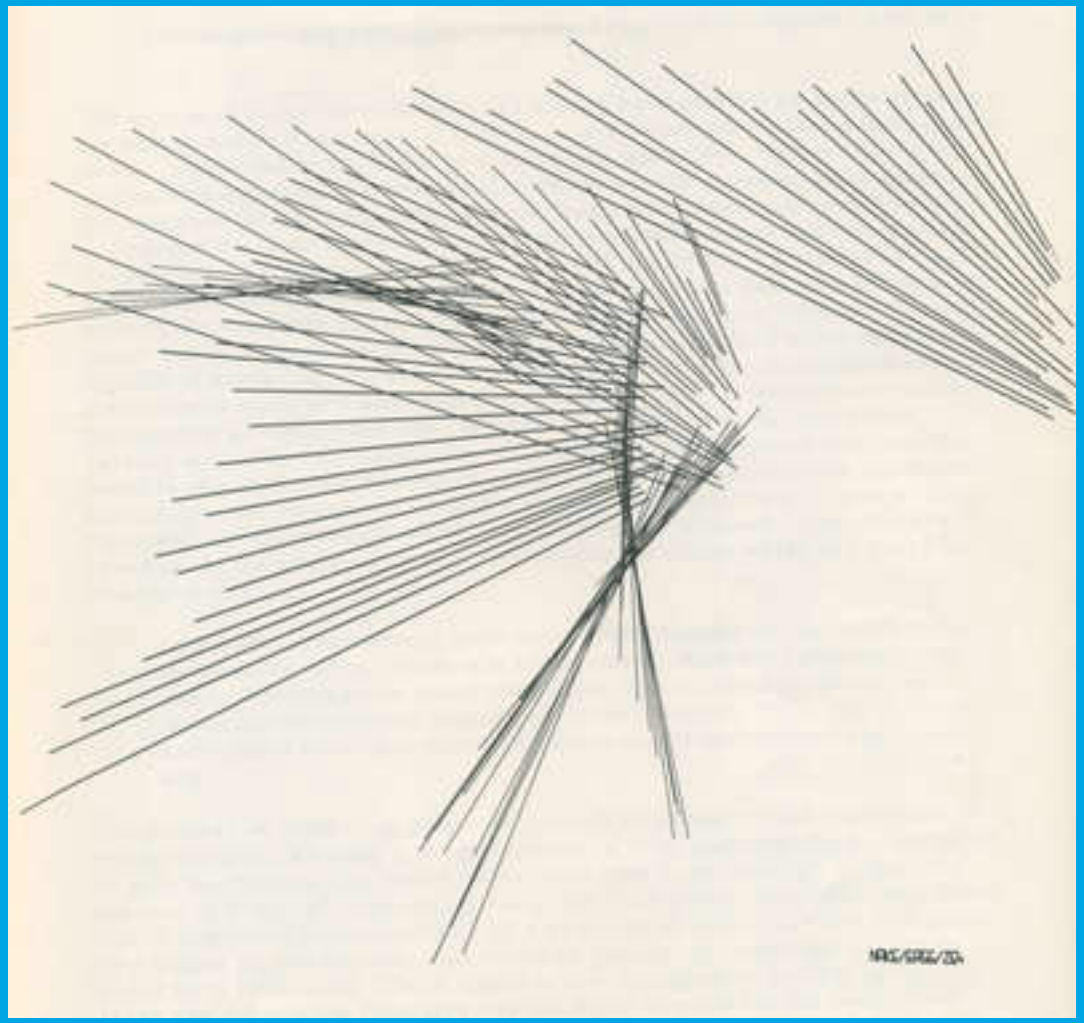


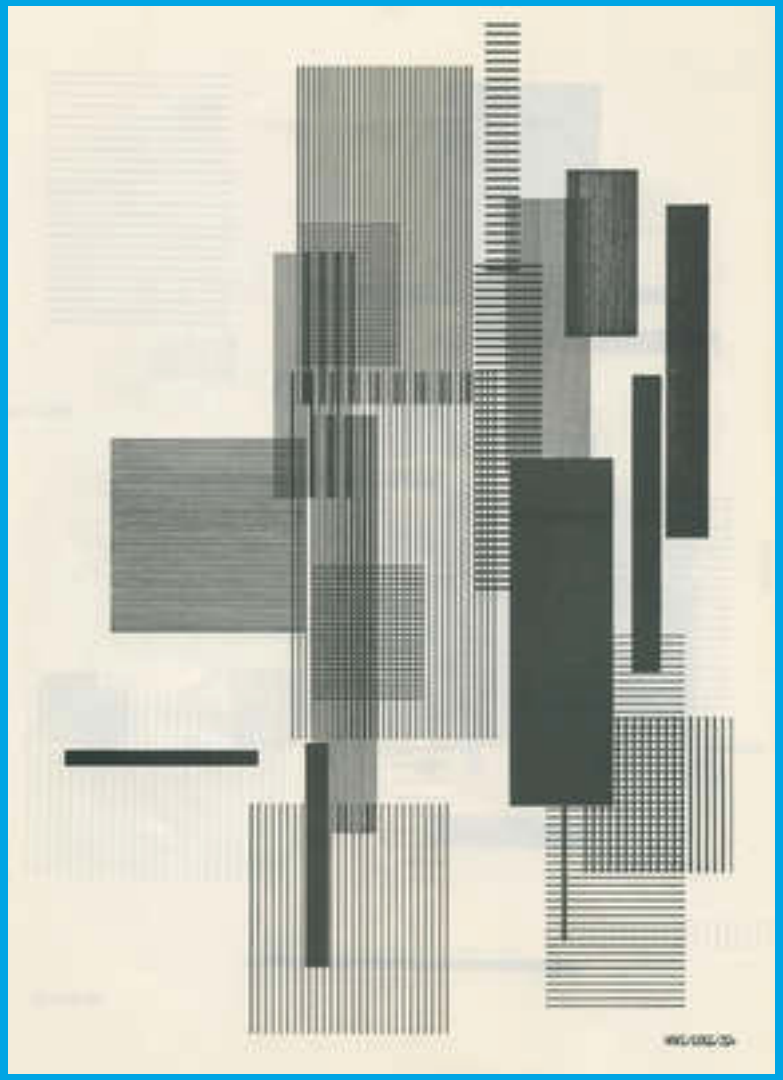
13/9/65 Nr. 7 Zufälliger Polygonzug



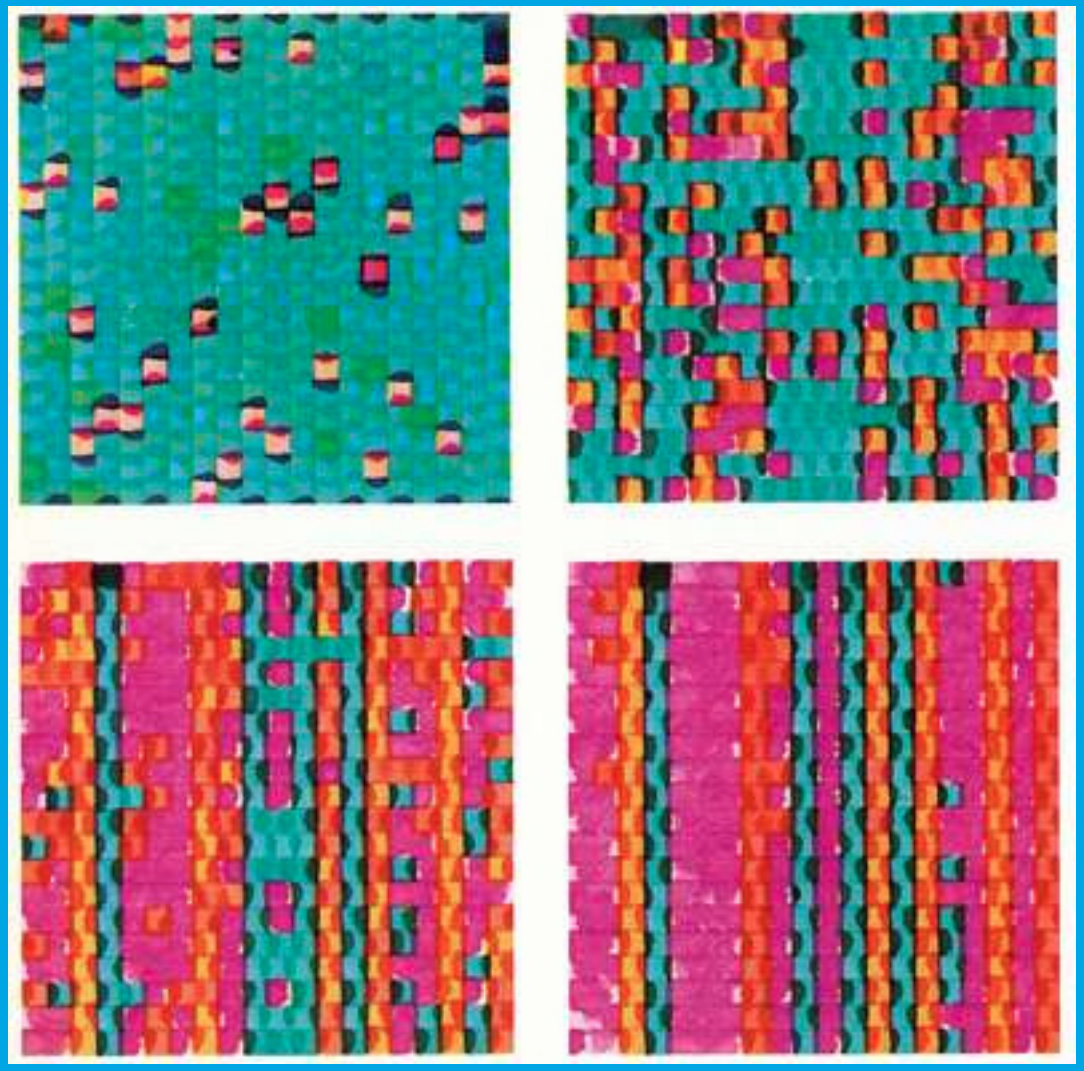
13/9/65 Nr. 2 (“Klee”)

Geradenscharen Nr. 3
(12/07/1965)



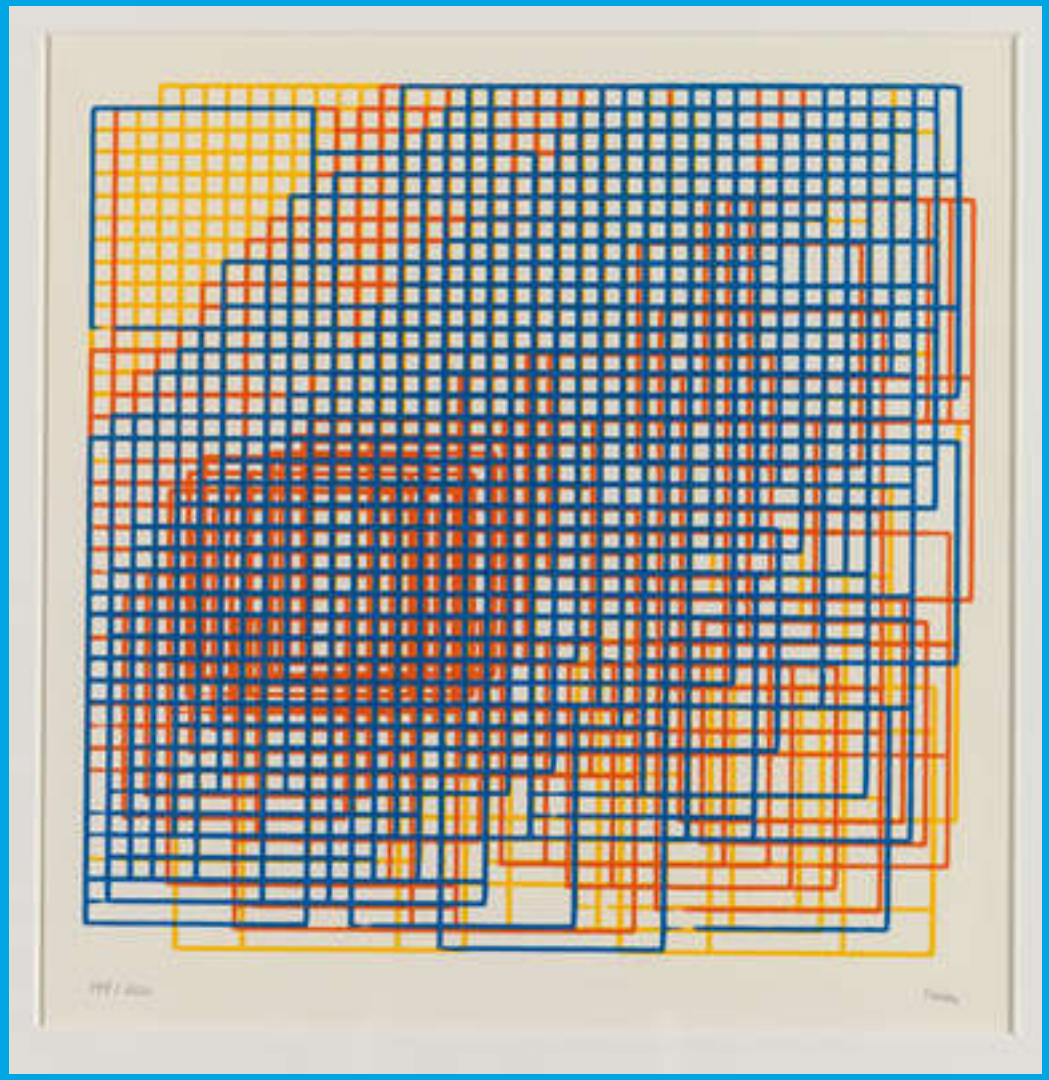


Rechteckschraffuren Nr. 3
(30/03/1965)



Matrizenmultiplikation, Serie 40, 1968

Walk-Through Raster, 1972



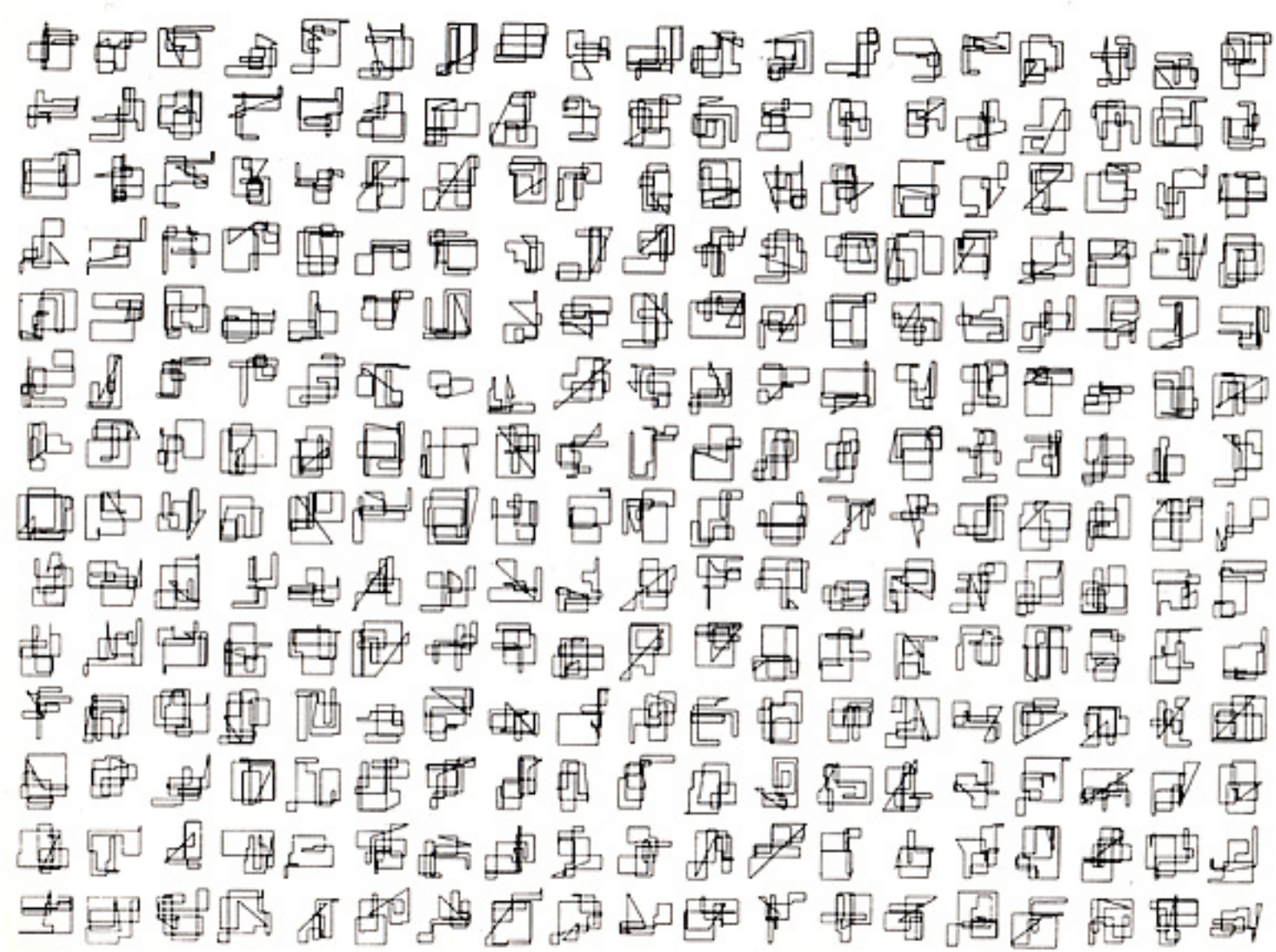
Georg Nees

1926, Nürnberg

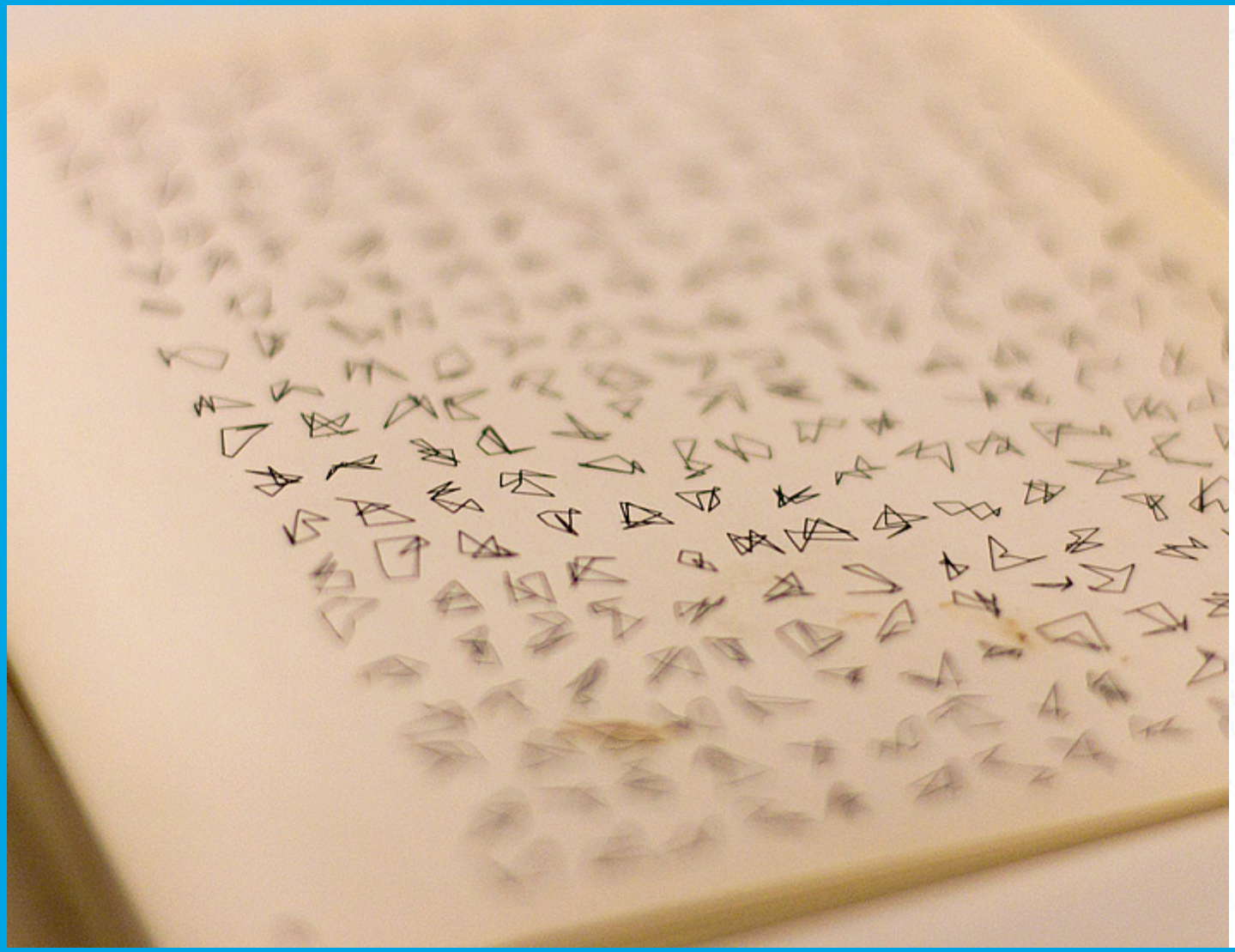
Mathematiker

Mathematiker, Philosoph

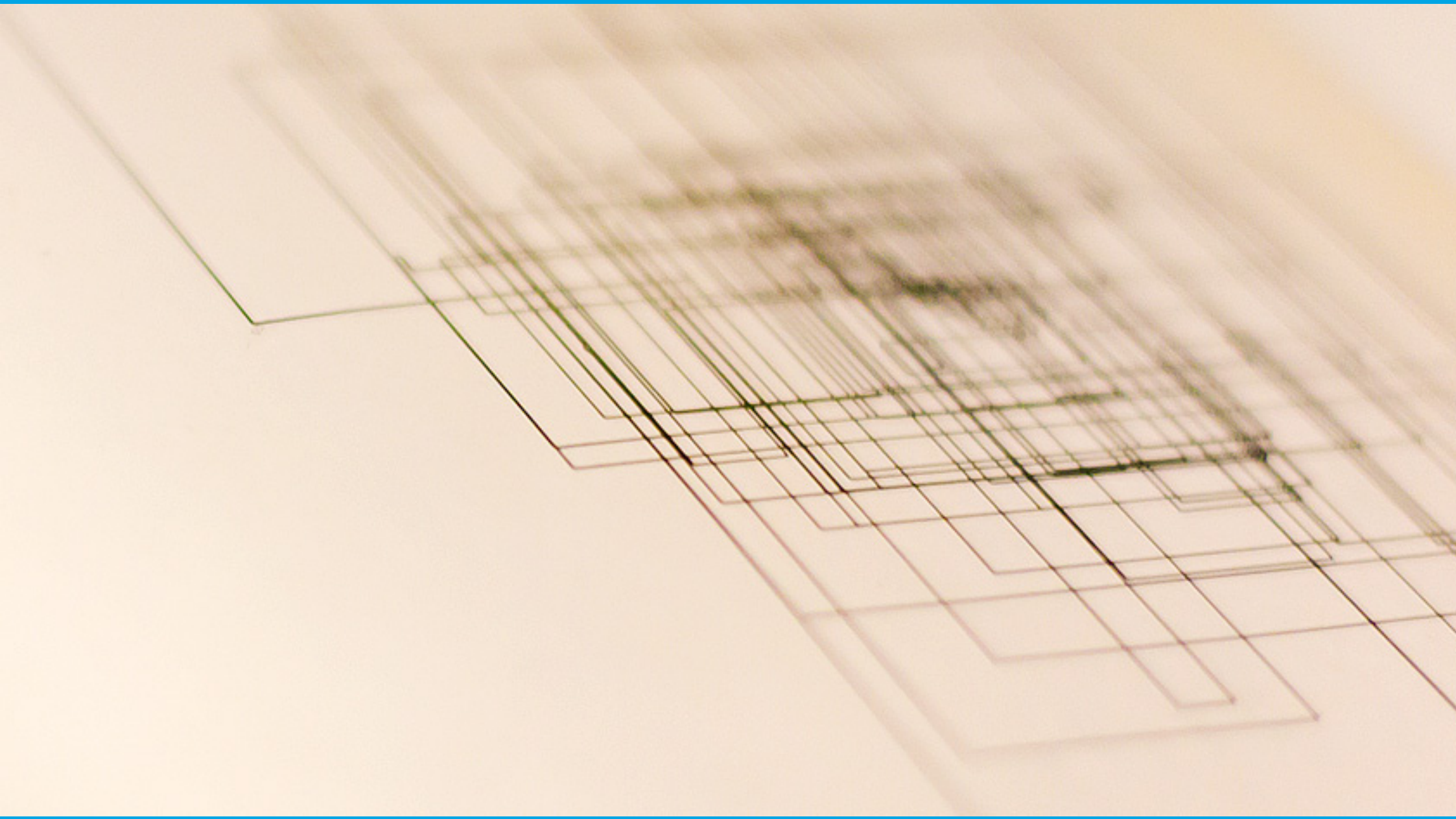
1968 Dissertation über Generative Computergraphik

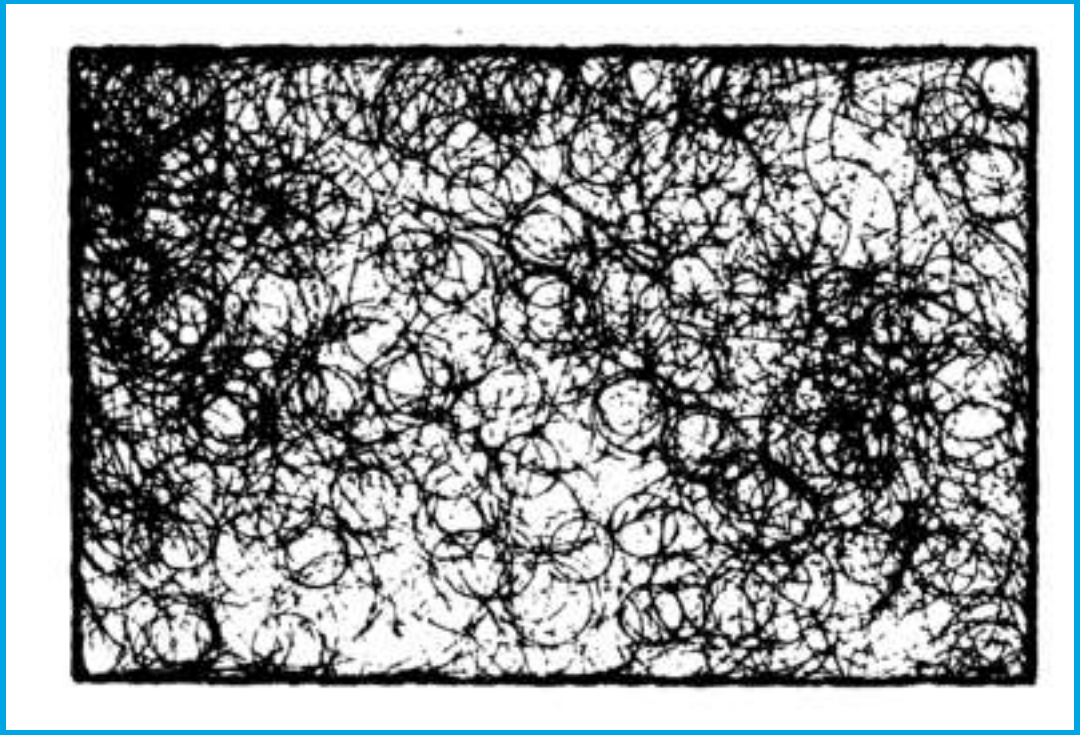


23-Ecke, 1964

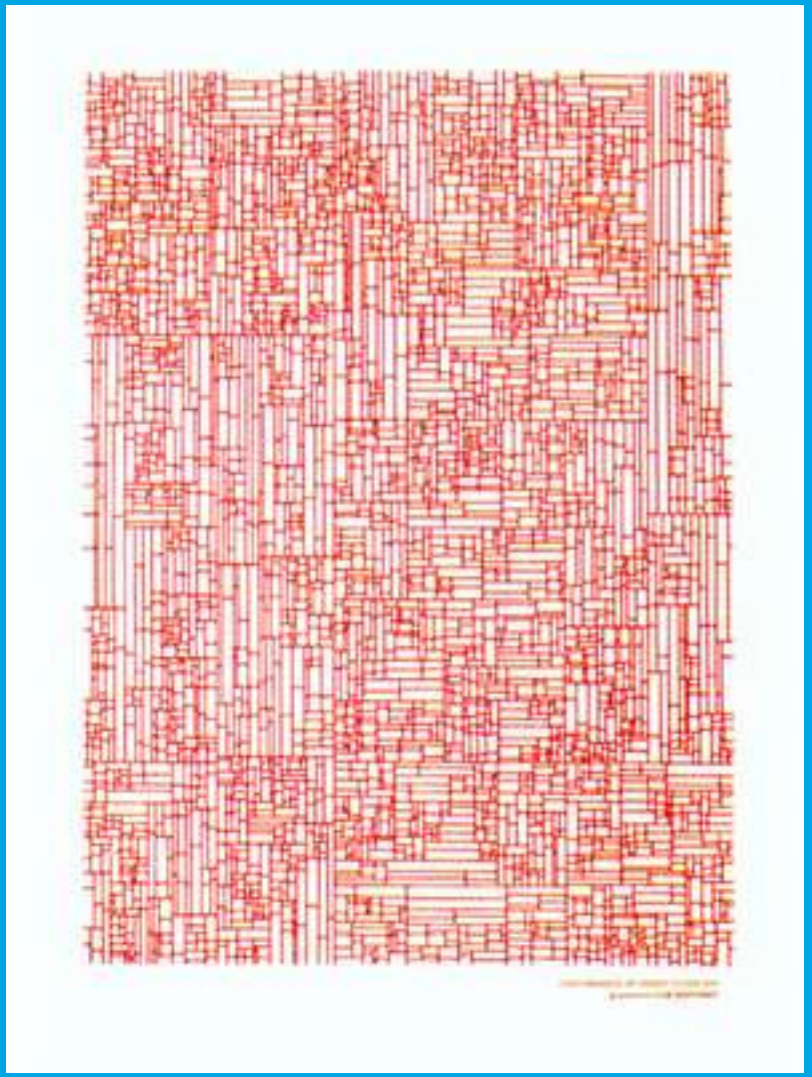


8-ecke



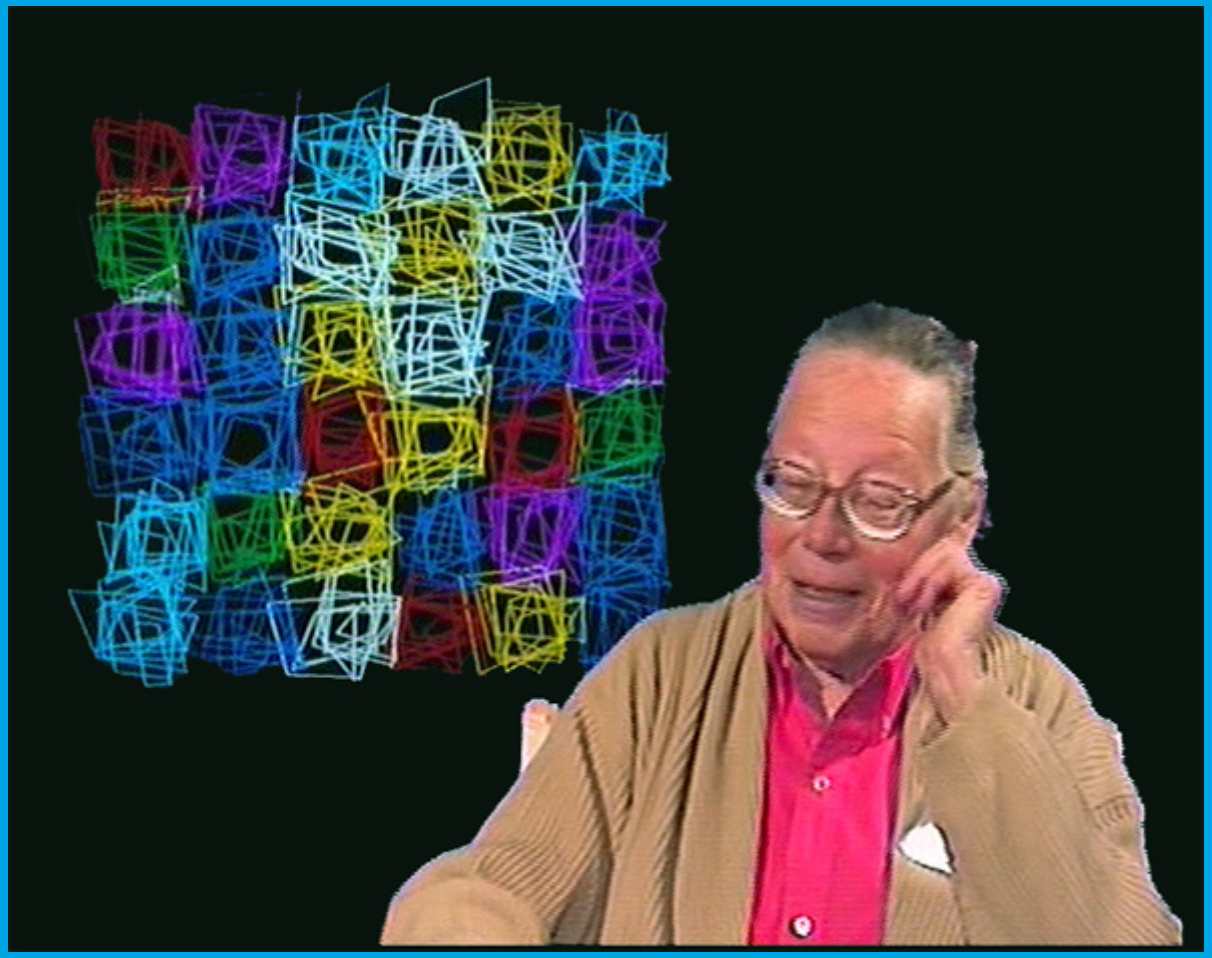


Kreisbogengewirre, vor 1969



untitled

Vera Molnar

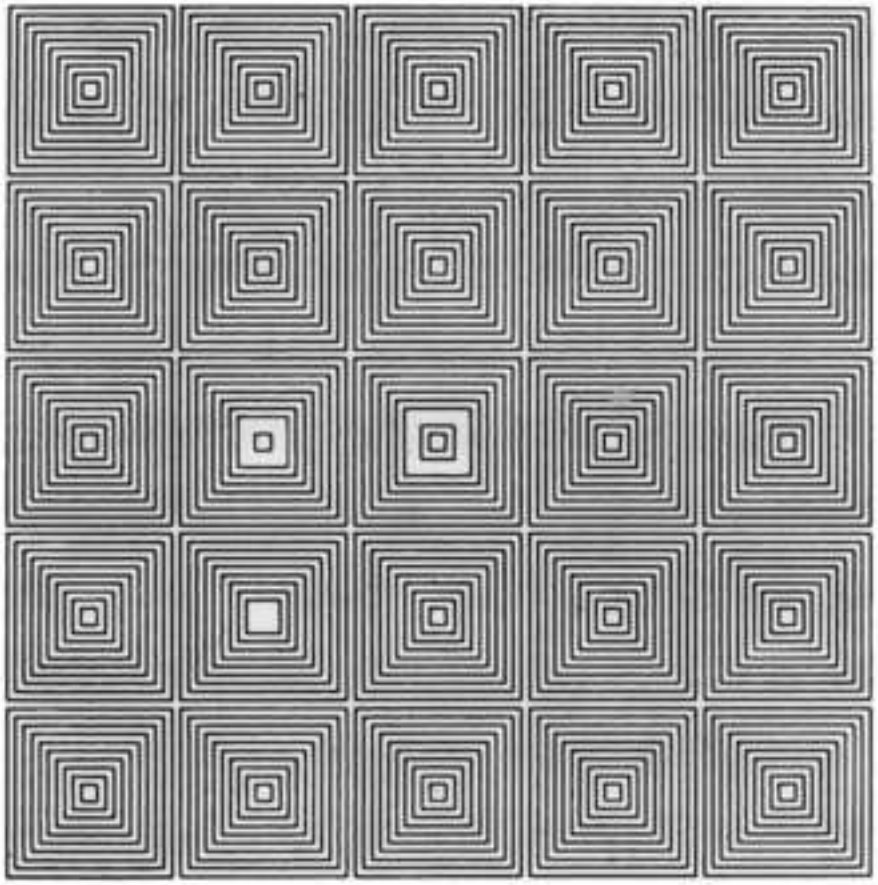


Dialog Between Emotion and Method, 1986

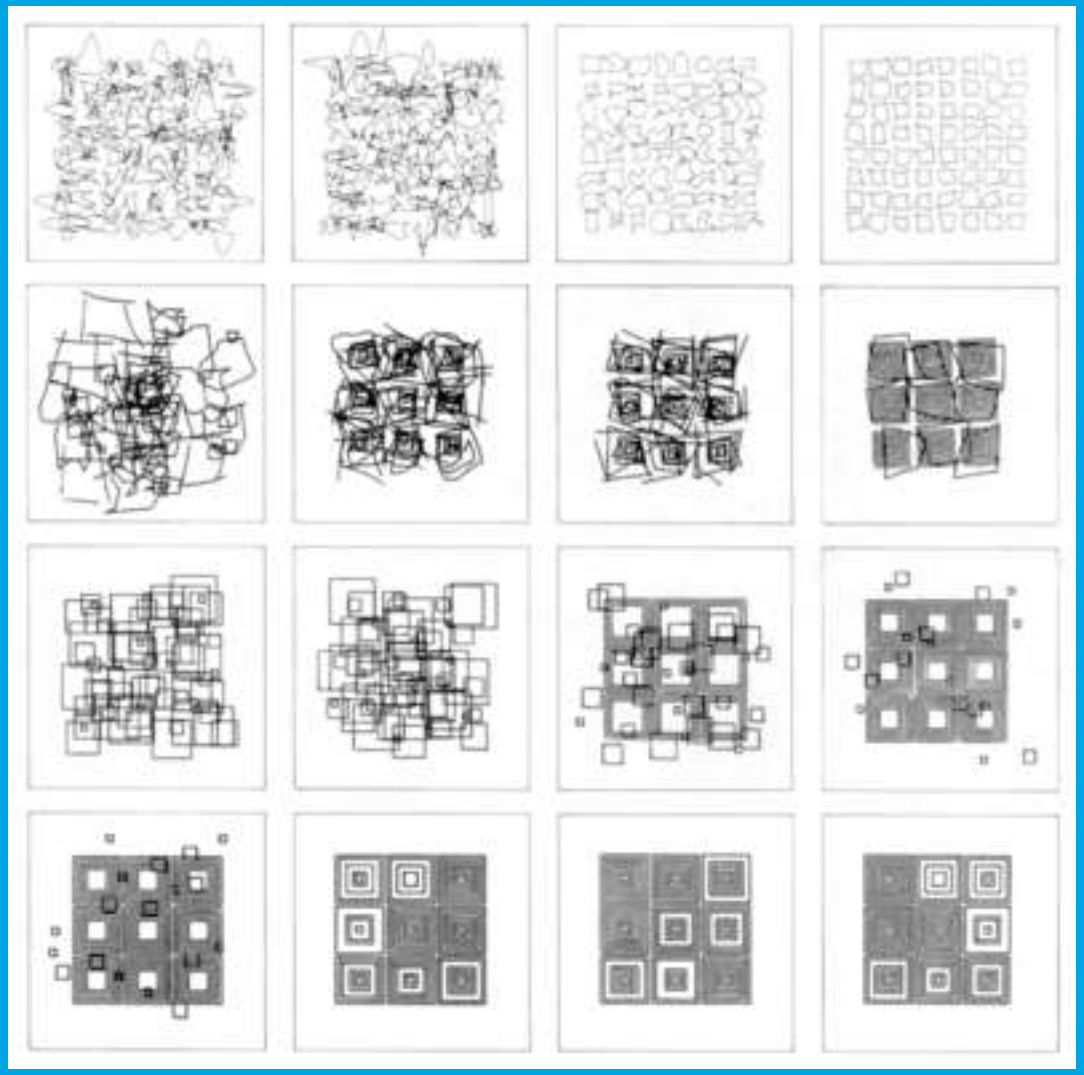
1924, Budapest

1942, Kunststudium Budapest

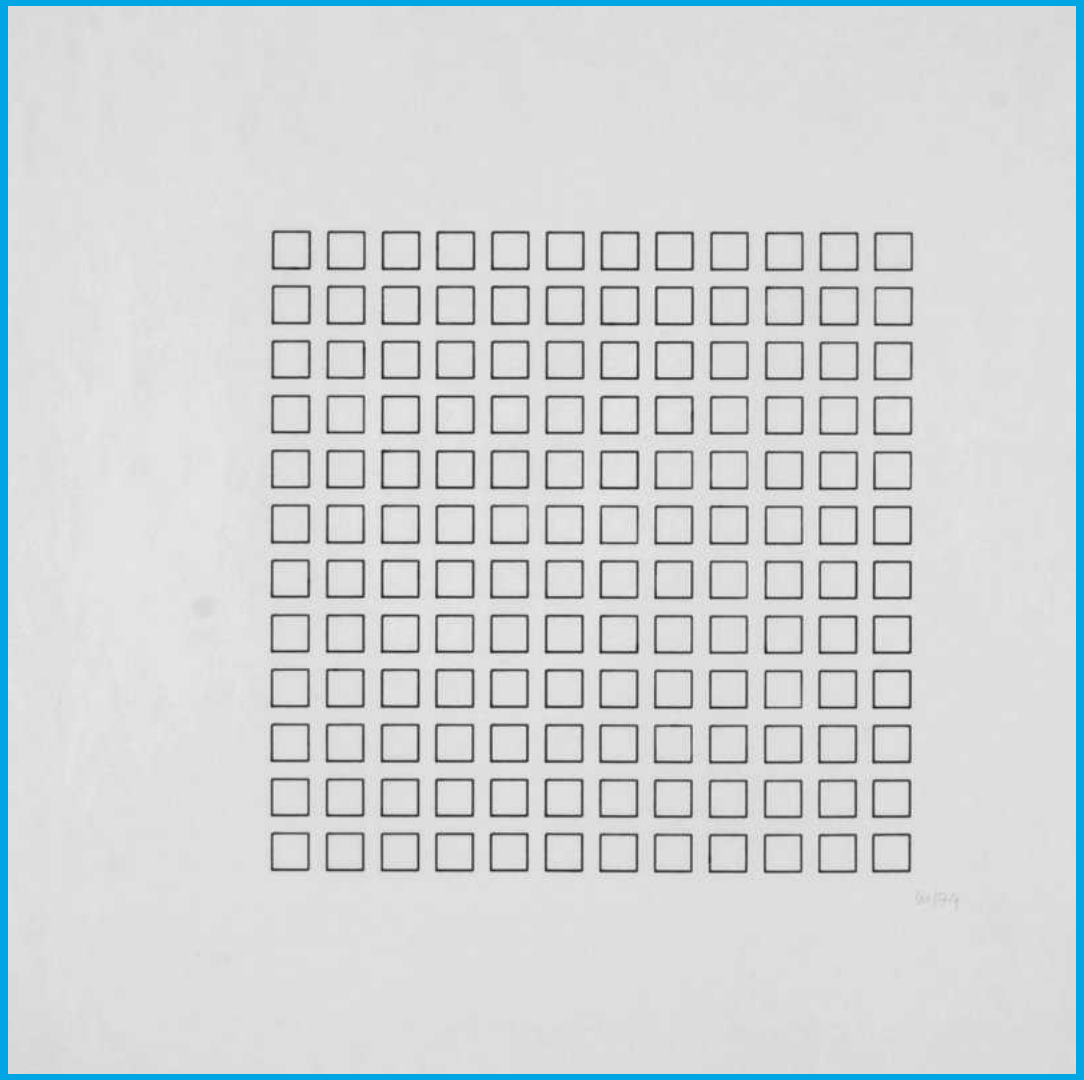
arbeitet seit 1968 nur am Computer

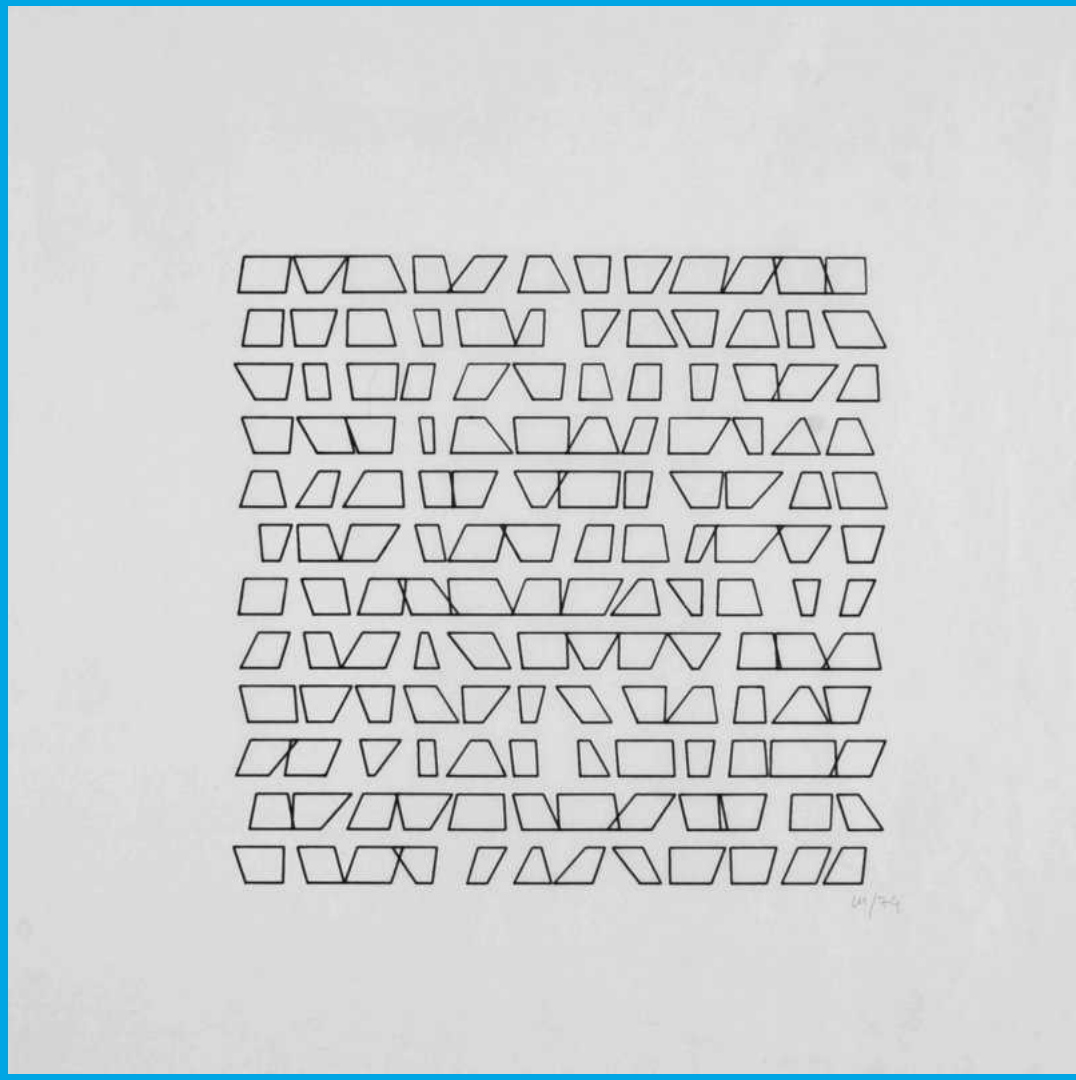


1% Unordnung, 1976

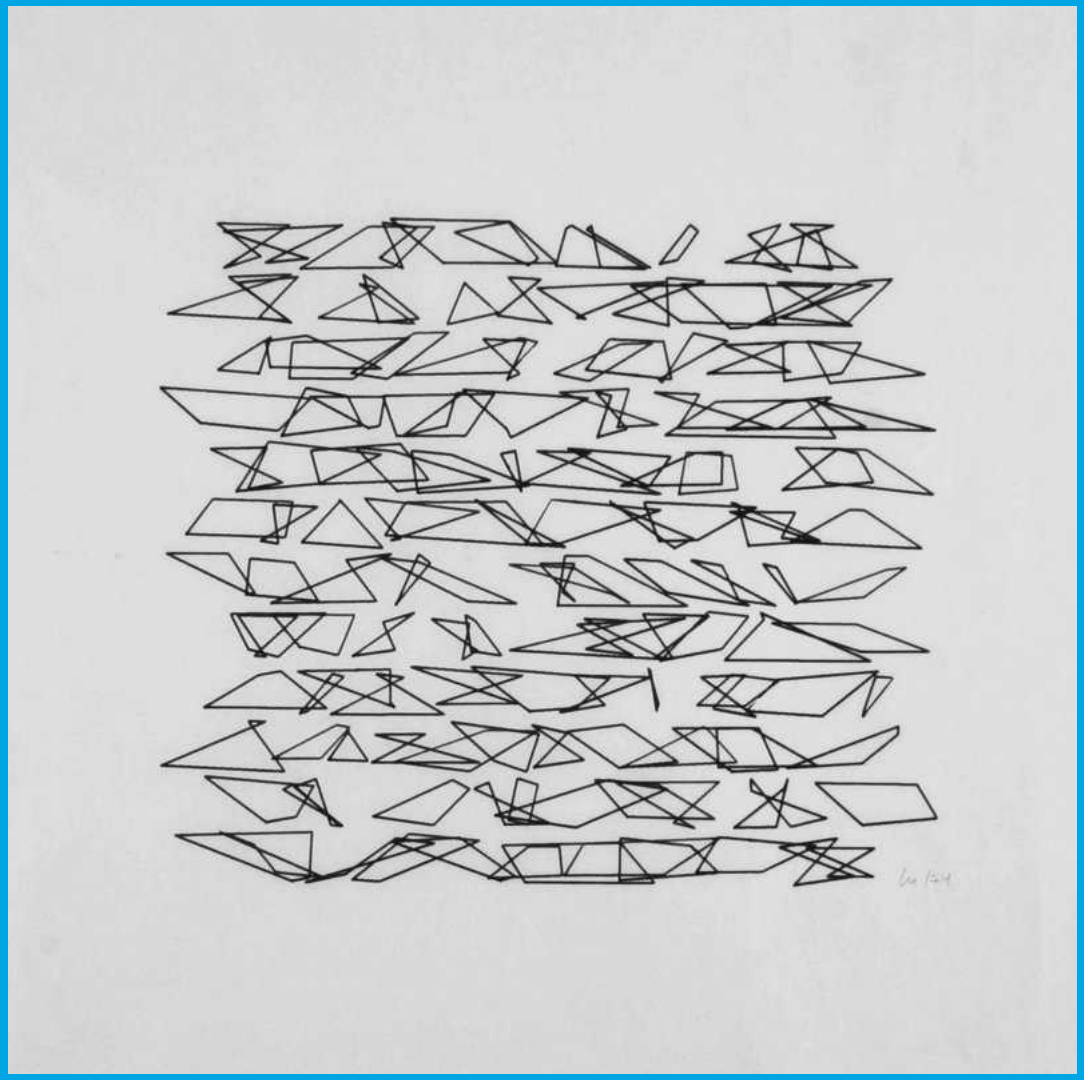


Squares, 1974

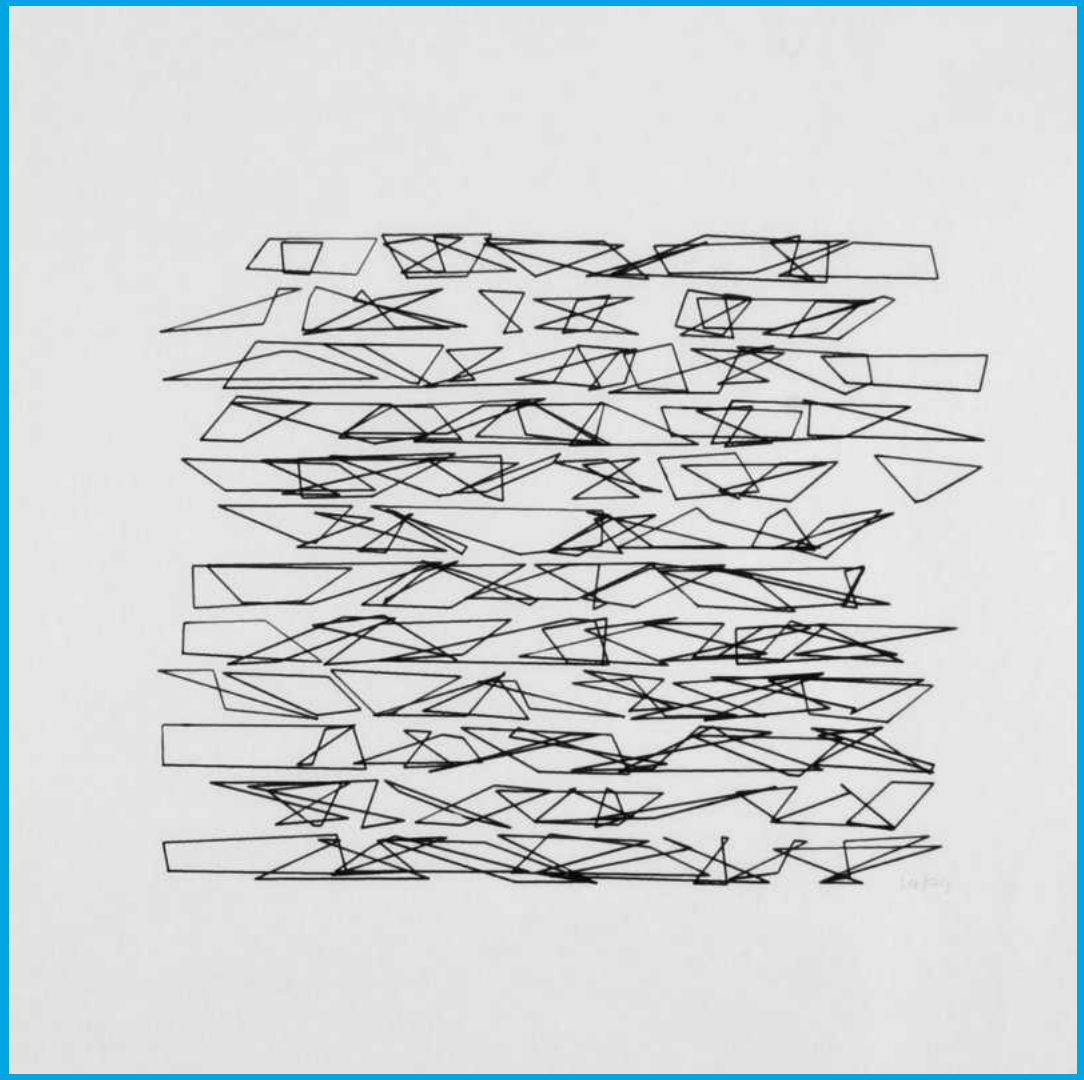




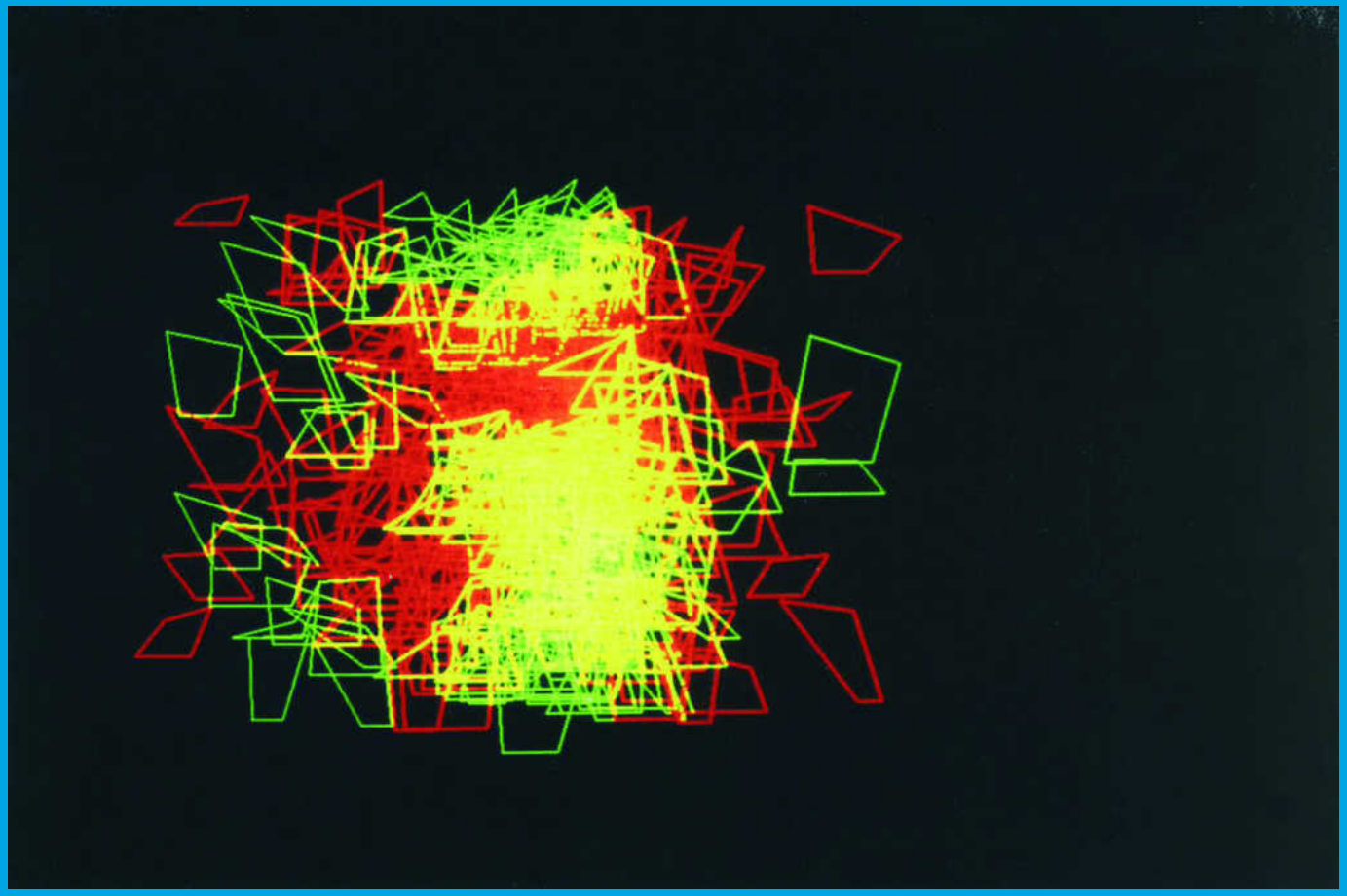
144 Trapézes, 1974



144 Trapézes, 1974



25 Squares, 1990



Square Structures, 1986, screenshot

http://freeartbureau.org/fab_activity/the-computer-from-hell/

http://dada.compart-bremen.de/

http://digitalartmuseum.org/index.htm