# OOP
# Arrays
# Array Listen
# Klassen / Objekte
# 2D Transformationen
# PGraphics

# Objekt-orientierte Programmierung

1. Everything is an object,
2. Objects communicate by sending and receiving messages (in terms of objects),
3. Objects have their own memory (in terms of objects),
4. Every object is an instance of a class (which must be an object),
5. The class holds the shared behavior for its instances (in the form of objects in a program list),
6. To eval a program list, control is passed to the first object and the remainder is treated as its message

– Alan Kay Curtis (Erfinder "Smalltalk" und "object-oriented")

# Objekt-orientierte Programmierung

OOP to me means only messaging, local retention and protection and hiding of state-process, and extreme late-binding of all things.”



# Objekt-orientierte Programmierung

# Arrays

1991 1992 1999 2010 2019

# Arrays

| 1991 | 1992 | 1999 | 2010 | 2019 |
|---|---|---|---|---|
| 0 | 1 | 2 | 3 | 4 |

# Arrays

```
int[] myArray = {1991, 1992, 1999, 2010, 2019};
```



# Arrays

```
int[] myArray = {1991, 1992, 1999, 2010, 2019};
println(myArray[0]);
```

# Arrays

```
int[] myArray = {1991, 1992, 1999, 2010, 2019};
println(myArray[0]);
println(myArray[1]);
```

# Arrays

```
int[] myArray = {1991, 1992, 1999, 2010, 2019};
println(myArray[0]);
println(myArray[1]);

println(myArray.length);
```

# Arrays

```
int[] myArray = {1991, 1992, 1999, 2010, 2019};
println(myArray[0]);
println(myArray[1]);

println(myArray.length);
println(myArray[ myArray.length-1 ]);
```

# Arrays

```
int x0 = 50;
int x1 = 80;
int x2 = 20;
int x3 = 30;
int x4 = 70;
int x5 = 30;
int x6 = 20;
int x7 = 80;
int x8 = 100;
int x9 = 10;
```

```
int[] myX = {50, 80, 20, 30, 70,30, 20, 80, 100, 10};
```



# Arrays

```
int x0 = 50;
int x1 = 80;
int x2 = 20;
int x3 = 30;
int x4 = 70;
int x5 = 30;
int x6 = 20;
int x7 = 80;
int x8 = 100;
int x9 = 10;

fill(0);
rect(0, 0, x0, 8);
rect(0, 10, x1, 8);
rect(0, 20, x2, 8);
rect(0, 30, x3, 8);
rect(0, 40, x4, 8);
rect(0, 50, x5, 8);
rect(0, 60, x6, 8);
rect(0, 70, x7, 8);
rect(0, 80, x8, 8);
rect(0, 90, x9, 8);
```

# Arrays

```
int[] myX = {50, 80, 20, 30, 70,30, 20, 80, 100, 10};

fill(0);
for (int i = 0; i < myX.length; i++) {
  rect(0, i*10, myX[i], 8);
}
```

# Arrays

```
int[] myX = {50, 80, 20, 30, 70,30, 20, 80, 100, 10};

fill(0);
for (int i = 0; i < myX.leng
  rect(0, i*10, myX[i], 8);
}
```


# Arrays

```
int[] myArray = {1991, 1992, 1999, 2010, 2019}; // Deklarieren, erschaffen
und zuweisen
```

---

```
int[] myArray = new int[5]; // Deklarieren und Array erschaffen
myArray[0] = 1991; // Zuweisen
```

---

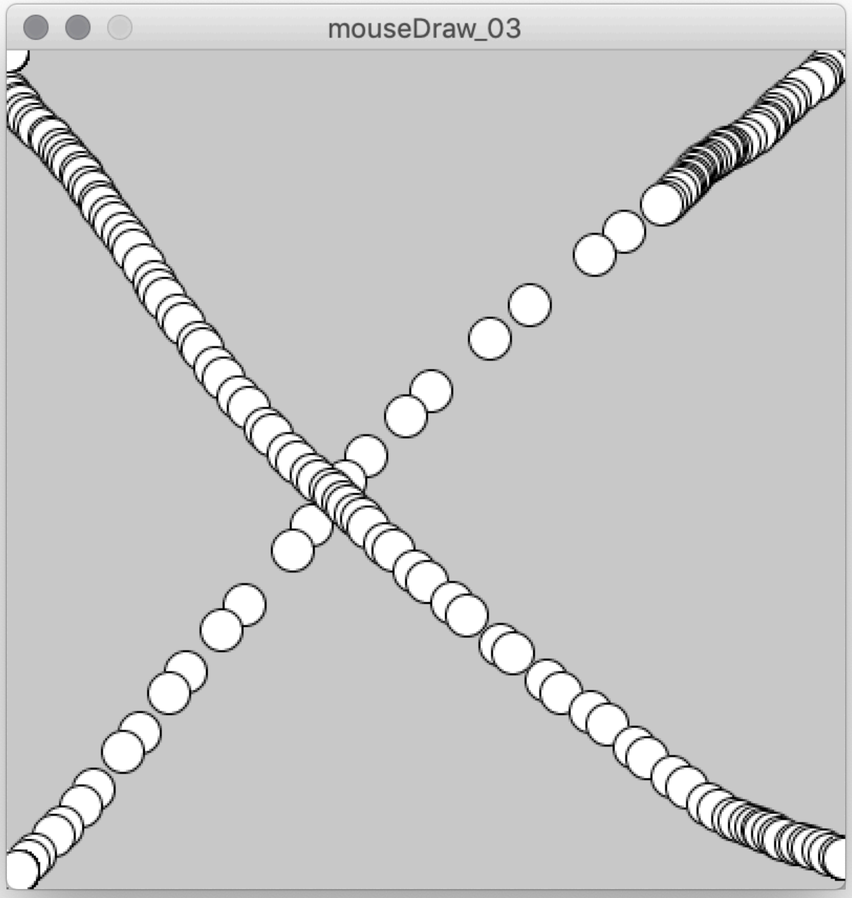
```
int[] myArray; // Deklarieren

myArray = new int[5]; // erschaffen
myArray[0] = 1991; // Zuweisen
```

# Arrays

```
void setup() {
  size(400, 400);
}

void draw() {
  circle(mouseX, mouseY, 20);
}
```

# mouseX, mouseY

```
void setup() {
  size(400, 400);
}

void draw() {
  circle(mouseX, mouseY,
}
```

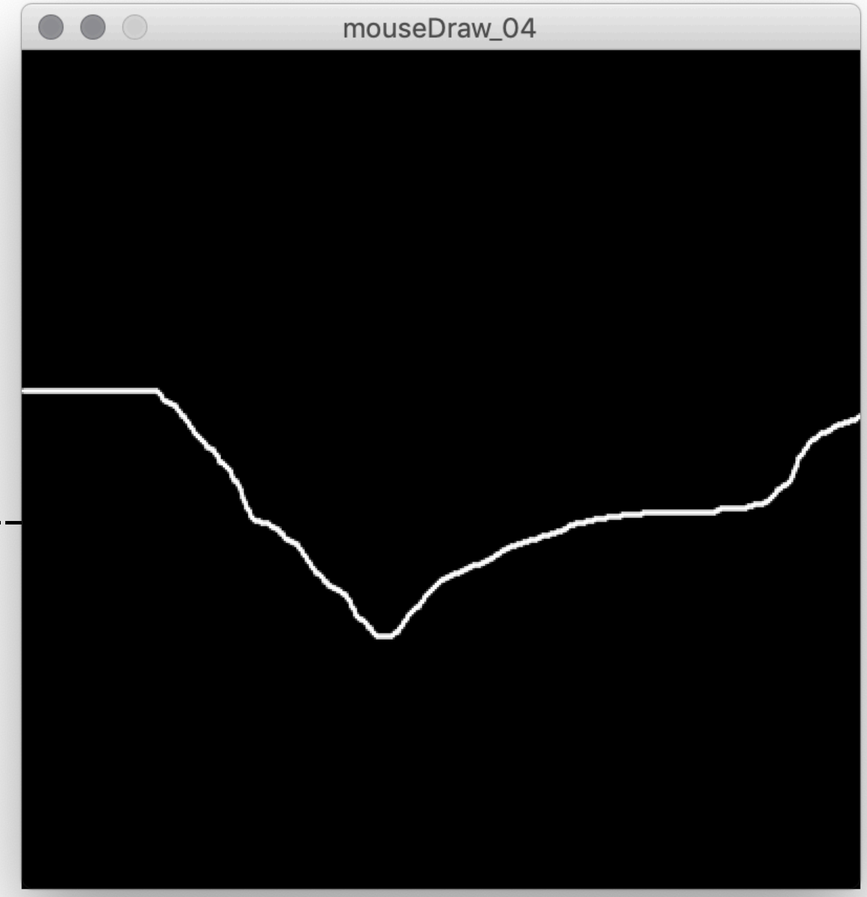



# mouseX, mouseY

```
int[] y;

void setup() {
  size(400, 400);
  y = new int[width];
  strokeWeight(2);
}

void draw() {
  background(0);
  for (int i = y.length-1; i > 0; i--) {
    y[i] = y[i-1];
  }

  y[0] = mouseY;
  stroke(255);
  for (int i = 1; i < y.length; i++) {
    line(i, y[i], i-1, y[i-1]);
  }
}
```

# mouseY

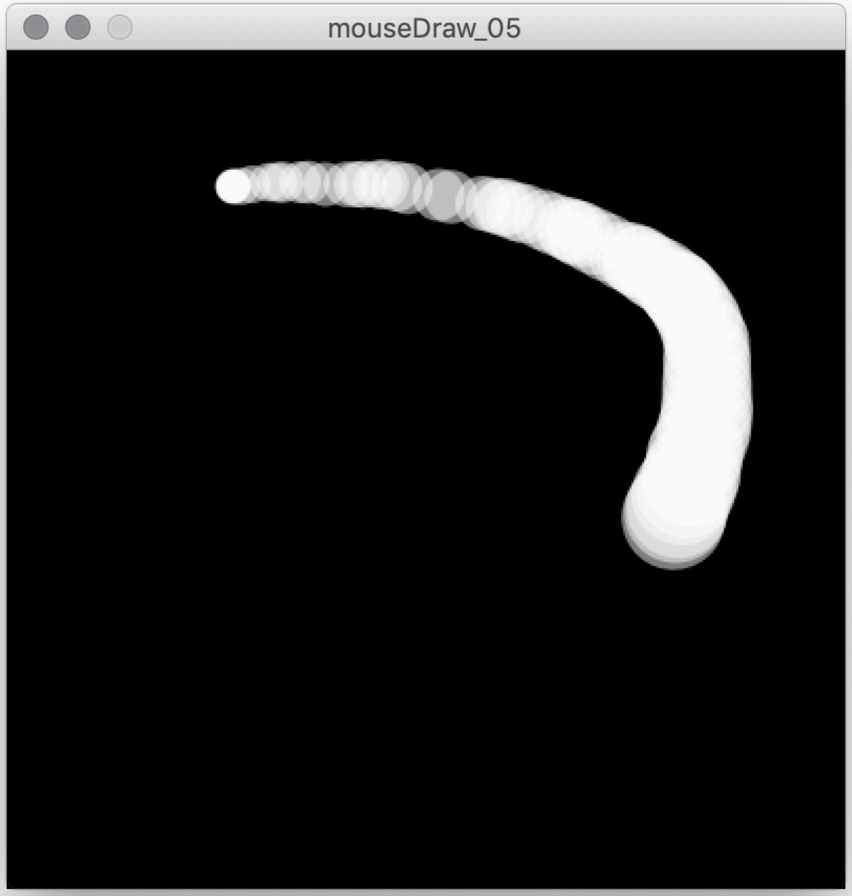
```
int[] y;

void setup() {
  size(400, 400);
  y = new int[width];
  strokeWeight(2);
}

void draw() {
  background(0);
  for (int i = y.length-1; i > 0; i--
    y[i] = y[i-1];
  }

  y[0] = mouseY;
  stroke(255);
  for (int i = 1; i < y.length; i++) {
    line(i, y[i], i-1, y[i-1]);
  }
}
```



```
int num = 100;
int[] x = new int[num];
int[] y = new int[num];

void setup() {
  size(400, 400);
  noStroke();
  fill(255, 127);
}

void draw() {
  background(0);

  for (int i = num-1; i > 0; i--) {
    x[i] = x[i-1];
    y[i] = y[i-1];
  }

  x[0] = mouseX;
  y[0] = mouseY;

  for (int i = 0; i < num; i++) {
    ellipse(x[i], y[i], i/2.0, i/2.0);
  }
}
```

# mouseX + mouseY

```
int num = 100;
int[] x = new int[num];
int[] y = new int[num];

void setup() {
  size(400, 400);
  noStroke();
  fill(255, 127);
}

void draw() {
  background(0);

  for (int i = num-1; i > 0; i--) {
    x[i] = x[i-1];
    y[i] = y[i-1];
  }

  x[0] = mouseX;
  y[0] = mouseY;

  for (int i = 0; i < num; i++) {
    ellipse(x[i], y[i], i/2.0, i/2.0);
  }
}
```




# mouseX + mouseY

# Erweitert den Code um weitere Formen.
# Zufällige Auswahl von neuen Formen?
# Linien die die versch. Formen verbinden?
# Wachstumsrichtung ändern?

# 15 Minuten Übung

Fingerübung

```
int num = 100;
int[] x = new int[num];
int[] y = new int[num];

void setup() {
  size(400, 400);
  noStroke();
  fill(255, 127);
}

void draw() {
  background(0);

  for (int i = num-1; i > 0; i--) {
    x[i] = x[i-1];
    y[i] = y[i-1];
  }

  x[0] = mouseX;
  y[0] = mouseY;

  for (int i = 0; i < num; i++) {
    ellipse(x[i], y[i], i/2.0, i/2.0);
  }
}
```

# Array von Objekten

```
class Ring {
  …
}
```



# Array von Objekten

```
class Ring {
  …
}

Ring[] rings;
int numRings = 50;
```

# Array von Objekten

```
class Ring {
  …
}

Ring[] rings;
int numRings = 50;

void setup() {
  size(600, 600);
  rings = new Ring[numRings];
  for (int i = 0; i < rings.length; i++) {
    rings[i] = new Ring();
  }
}
```

# Array von Objekten

```
…
void setup() {
  size(600, 600);
  rings = new Ring[numRings];
  for (int i = 0; i < rings.length; i++) {
    rings[i] = new Ring();
  }
}

void draw() {
  background(0);
  for (int i = 0; i < rings.length; i++) {
    rings[i].grow();
    rings[i].display();
  }
}
```

# Array von Objekten

```
…
void setup() {
  …
}

void draw() {
  background(255);
  for (int i = 0; i < rings.length; i++) {
    rings[i].grow();
    rings[i].display();
  }
}

void mousePressed() {
  …
}
```

# Array von Objekten

# ArrayList

# Array von Objekten = Array Liste

```
Ring[] rings = new Ring[numRings];
```

vs.

```
ArrayList<Ring> rings = new ArrayList<Ring>();
```

# ArrayList

```
ArrayList<Ring> rings = new ArrayList<Ring>();

rings.add(new Ring());
```

# ArrayList

```
ArrayList<Ring> rings = new ArrayList<Ring>();

rings.add(new Ring());

Ring r = rings.get(0);
```

# ArrayList

```
ArrayList<Ring> rings = new ArrayList<Ring>();

rings.add(new Ring());

Ring r = rings.get(0);

println( rings.size() );
```

# ArrayList

```
ArrayList<Ring> rings = new ArrayList<Ring>();

rings.add(new Ring());

Ring r = rings.get(0);

println( rings.size() );

for (int i = 0; i < rings.size(); i++) {
  Ring ring = rings.get(i);
  ring.grow();
  ring.display();
}
```

# ArrayList

# Klassen

# Klassen (Objekte)

**Aufwachen**

**Kaffee / Tee trinken**

**Frühstücken**

**Zur Arbeit oder Uni fahren**

Klassen (Objekte)

**Aufwachen**

**Kaffee / Tee trinken**

**Frühstücken**

**Zur Arbeit oder Uni fahren**

Klassen (Objekte)

**Attribute**
**– Haarfarbe**
**– Augenfarbe**
**– Körpergröße**

**Funktionen**
**– Fahrrad fahren**
**– Parkour**
**– Kochen**

Klassen (Objekte)

**Attribute**
**– Haarfarbe**
**– Augenfarbe**
**– Körpergröße**

**Funktionen**
**– Fahrrad fahren**
**– Parkour**
**– Kochen**

Klassen (Objekte)

```
// Globale Variablen

void setup() {
}

void draw() {
}
```



# Klassen (Objekte)

```
// Globale Variablen
color c = color(0);
float x = 0;
float y = 100;
float speed = 1;

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

void draw() {
  background(255);
  move();
  display();
}
```

```
void move() {
  x = x + speed;
  if (x > width) {
    x = 0;
  }
}

void display() {
  fill(c);
  rect(x,y,30,10);
}
```



# Klassen (Objekte)

```
// Globale Variablen
color c = color(0);
float x = 0;
float y = 100;
float speed = 1;

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

void draw() {
  background(255);
  move();
  display();
}
```

```
void move() {
  x = x + speed;
  if (x > width) {
    x = 0;
  }
}

void display() {
  fill(c);
  rect(x,y,30,10);
}
```



# Klassen (Objekte)

```
Car myCar;

void setup()  {
  myCar = new Car();
}

void draw()  {
  background(255);
  myCar.drive();
  myCar.display();
}
```

# Klassen (Objekte)

```
Car myCar;

void setup()  {
  myCar = new Car();
}

void draw()  {
  background(255);
  myCar.drive();
  myCar.display();
}
```

```
class Car { // Klassenname
  // Unsere Daten
  color c;
  float xpos;
  float ypos;
  float xspeed;

  Car() { // Konstruktor
    c = color(255);
    xpos = width/2;
    ypos = height/2;
    xspeed = 1;
  }

  void display() {
    rectMode(CENTER);
    fill(c);
  }

  void move() {
    xpos = xpos + xspeed;
    if(xpos > width) {
      xpos = 0;
    }
  }
}
```

# Klassen (Objekte)

```
Car myCar;

void setup()  {
  myCar = new Car();
}

void draw()  {
  background(255);
  myCar.drive();
  myCar.display();
}
```

```
class Car { // Klassenname
  // Unsere Daten
  color c;
  float xpos;
  float ypos;
  float xspeed;

  Car() { // Konstruktor
    c = color(255);
    xpos = width/2;
    ypos = height/2;
    xspeed = 1;
  }

  void display() {
    rectMode(CENTER);
    fill(c);
  }

  void move() {
    xpos = xpos + xspeed;
    if(xpos > width) {
      xpos = 0;
    }
  }
}
```

# Klassen (Objekte)

```
Car myCar;

void setup()  {
  myCar = new Car();
}

void draw()  {
  background(255);
  myCar.drive();
  myCar.display();
}
```

```
class Car { // Klassenname
  // Unsere Daten
  color c;
  float xpos;
  float ypos;
  float xspeed;

  Car() { // Konstruktor
    c = color(255);
    xpos = width/2;
    ypos = height/2;
    xspeed = 1;
  }

  void display() {
    rectMode(CENTER);
    fill(c);
  }

  void move() {
    xpos = xpos + xspeed;
    if(xpos > width) {
      xpos = 0;
    }
  }
}
```

# Klassen (Objekte)

```
Car myCar;

void setup()  {
  myCar = new Car();
}

void draw()  {
  background(255);
  myCar.drive();
  myCar.display();
}
```

```
class Car { // Klassenname
  // Unsere Daten
  color c;
  float xpos;
  float ypos;
  float xspeed;

  Car() { // Konstruktor
    c = color(255);
    xpos = width/2;
    ypos = height/2;
    xspeed = 1;
  }

  void display() {
    rectMode(CENTER);
    fill(c);
  }

  void move() {
    xpos = xpos + xspeed;
    if(xpos > width) {
      xpos = 0;
    }
  }
}
```

# Klassen (Objekte)

```
Car myCar;
Car myCar2;

void setup()  {
  myCar = new Car(0, 100, 2);
  myCar2 = new Car(20, 20, 1);
}

void draw()  {
  background(255);
  myCar.drive();
  myCar.display();
  myCar2.drive();
  myCar2.display();
}
```

```
class Car { // Klassenname
  // Unsere Daten
  color c;
  float xpos;
  float ypos;
  float xspeed;

  Car(float _xp, float _yp, float _xs) {
    c = color(255);
    xpos = _xp;
    ypos = _yp;
    xspeed = _xs;
  }

  void display() {
    rectMode(CENTER);
    fill(c);
  }

  void move() {
    …
  }
}
```

# Klassen (Objekte)

# Matrix
# +
# Style

Stapel (Stack)

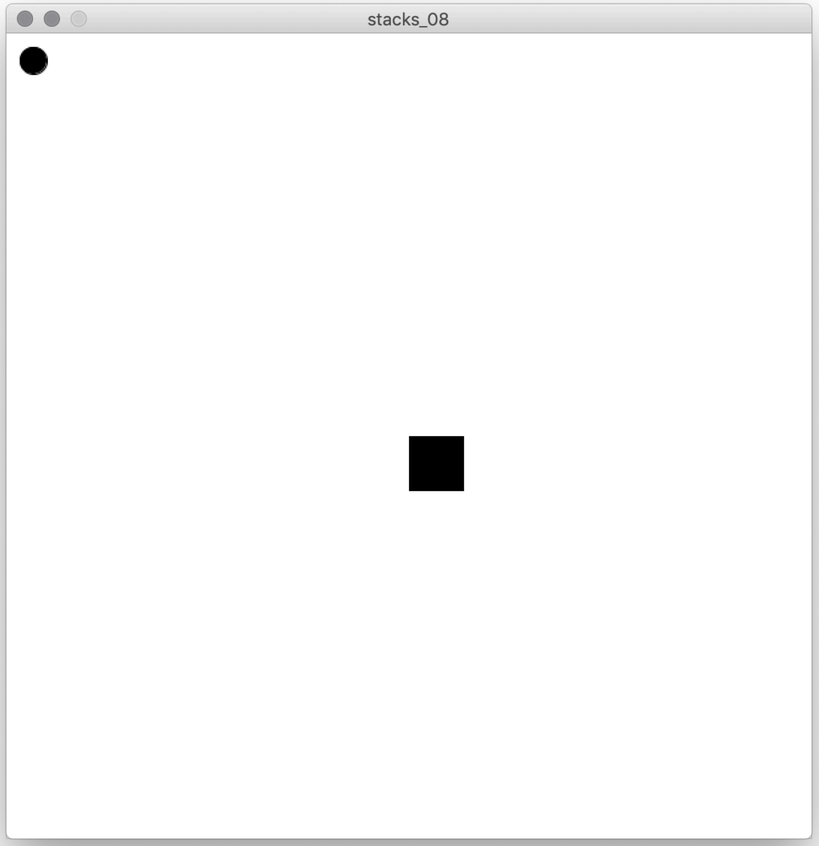
```
void setup() {
  size(600, 600);
}

void draw() {
  background(255);
  rect(width/2, height/2, 40, 40);
  fill(0);
  circle(20, 20, 20);
}
```

# Stapel (Stack)

```
void setup() {
  size(600, 600);
}

void draw() {
  background(255);
  rect(width/2, height/2, 40, 40);
  fill(0);
  circle(20, 20, 20);
}
```

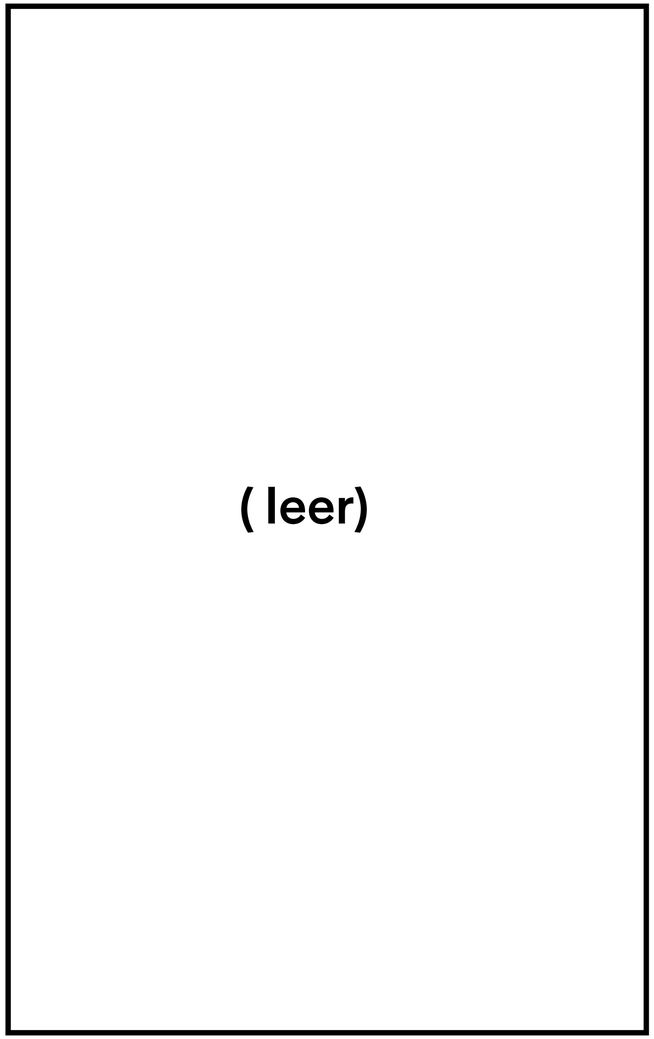
# Stapel (Stack)

```
void setup() {
  size(600, 600);
}

void draw() {
  background(255);
  rect(width/2, height/2,
  fill(0);
  circle(20, 20, 20);
}
```

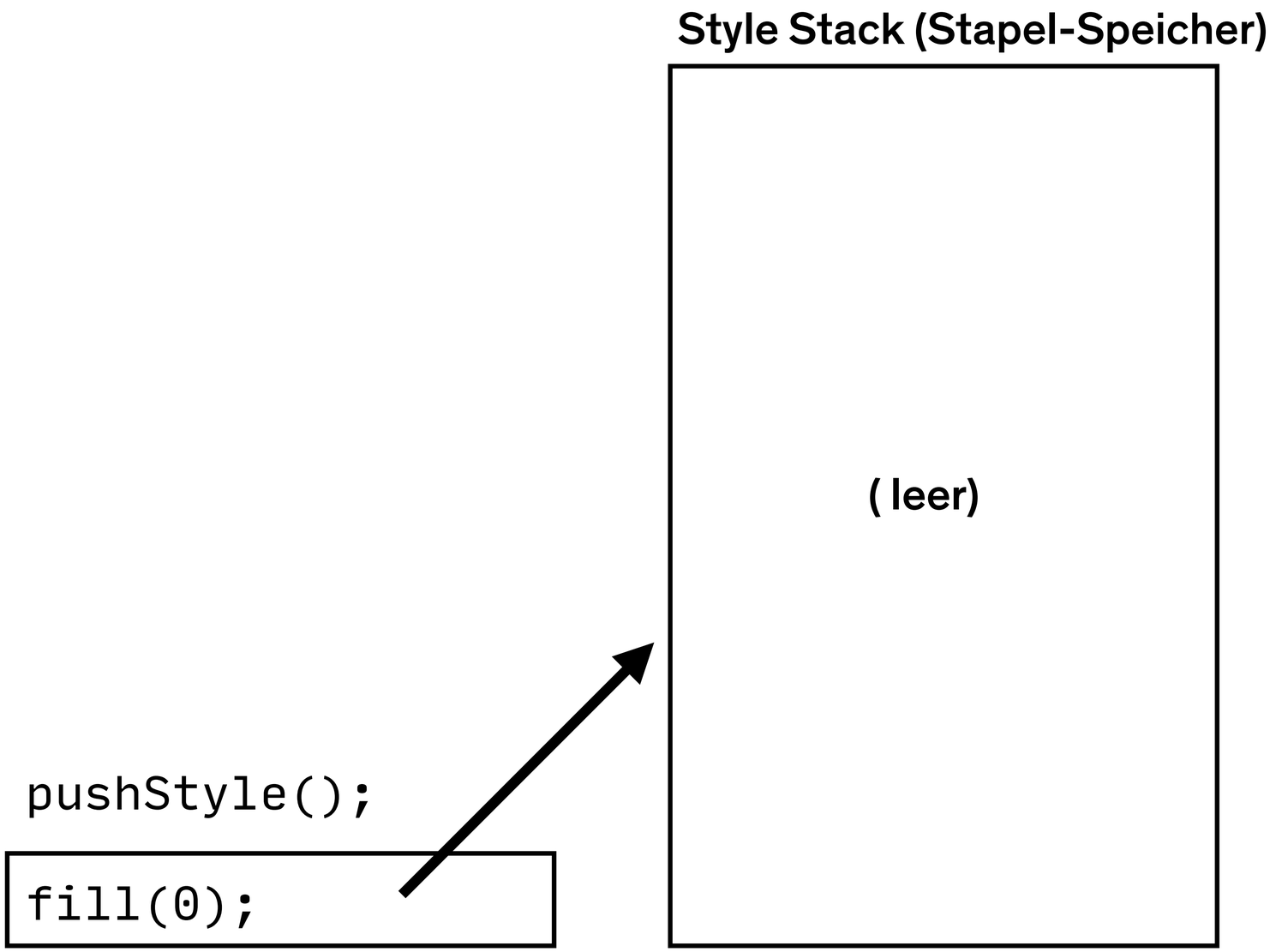



# Stapel (Stack)

Style Stack (Stapel-Speicher)

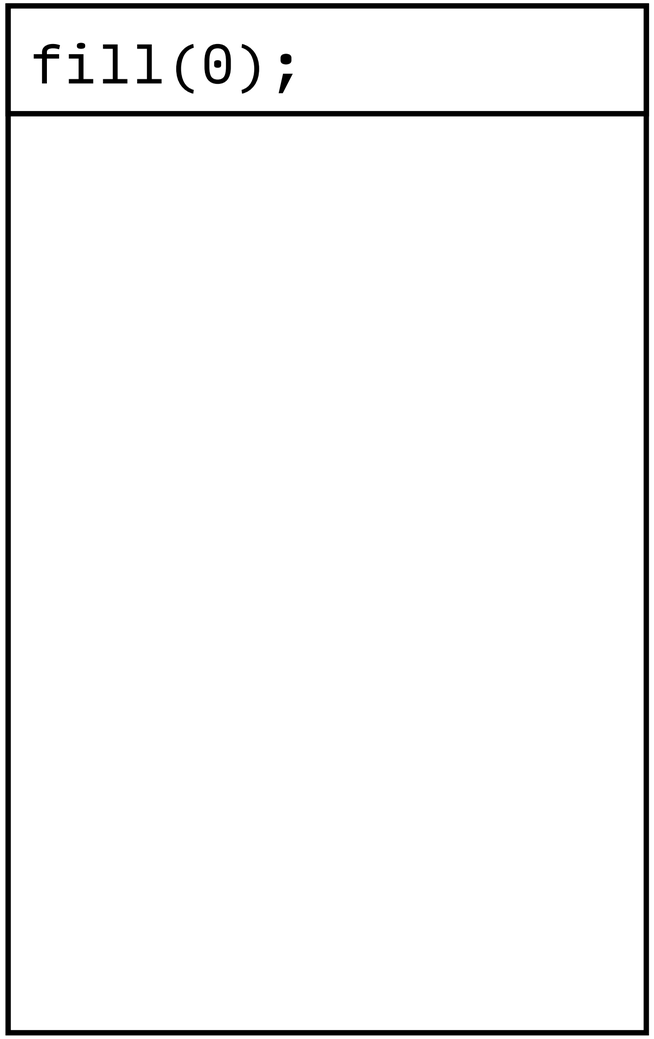



# Stapel (Stack)

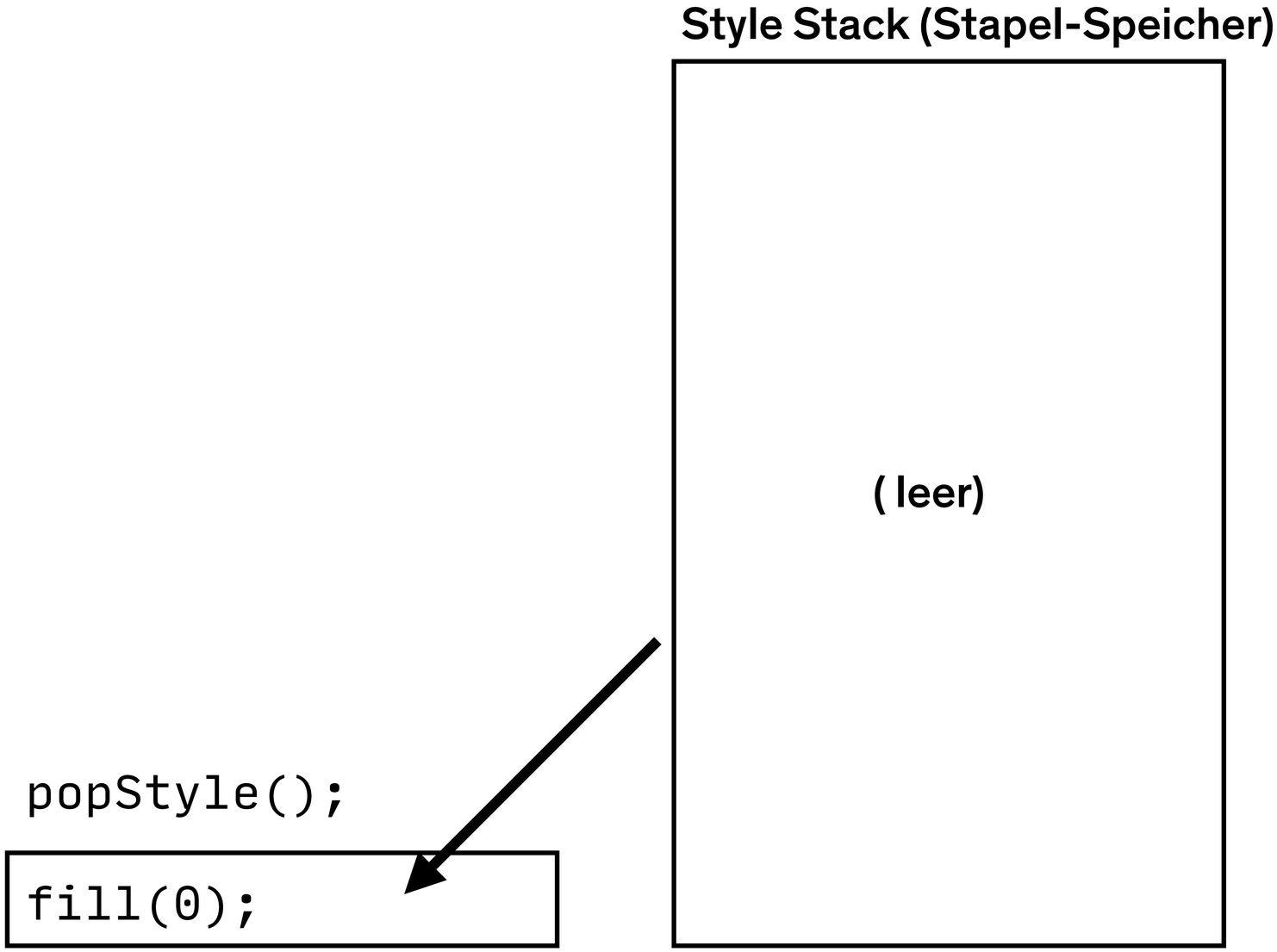



# Stapel (Stack)

**Style Stack (Stapel-Speicher)**




# Stapel (Stack)




# Stapel (Stack)

Style Stack (Stapel-Speicher)

( leer)

```
fill(0);
```

# Stapel (Stack)

Style Stack (Stapel-Speicher)

( leer)

```
fill(0);
```

# Stapel (Stack)

```
void setup() {
  size(600, 600);
}

void draw() {
  background(255);
  rect(width/2, height/2, 40, 40);
  pushStyle();
  fill(0);
  circle(20, 20, 20);
  popStyle();
}
```

# Stapel (Stack)

```
void setup() {
  size(400, 400);
}

void draw() {
  rect(20, 20, 40, 40);
}
```



# 2D Transformationen

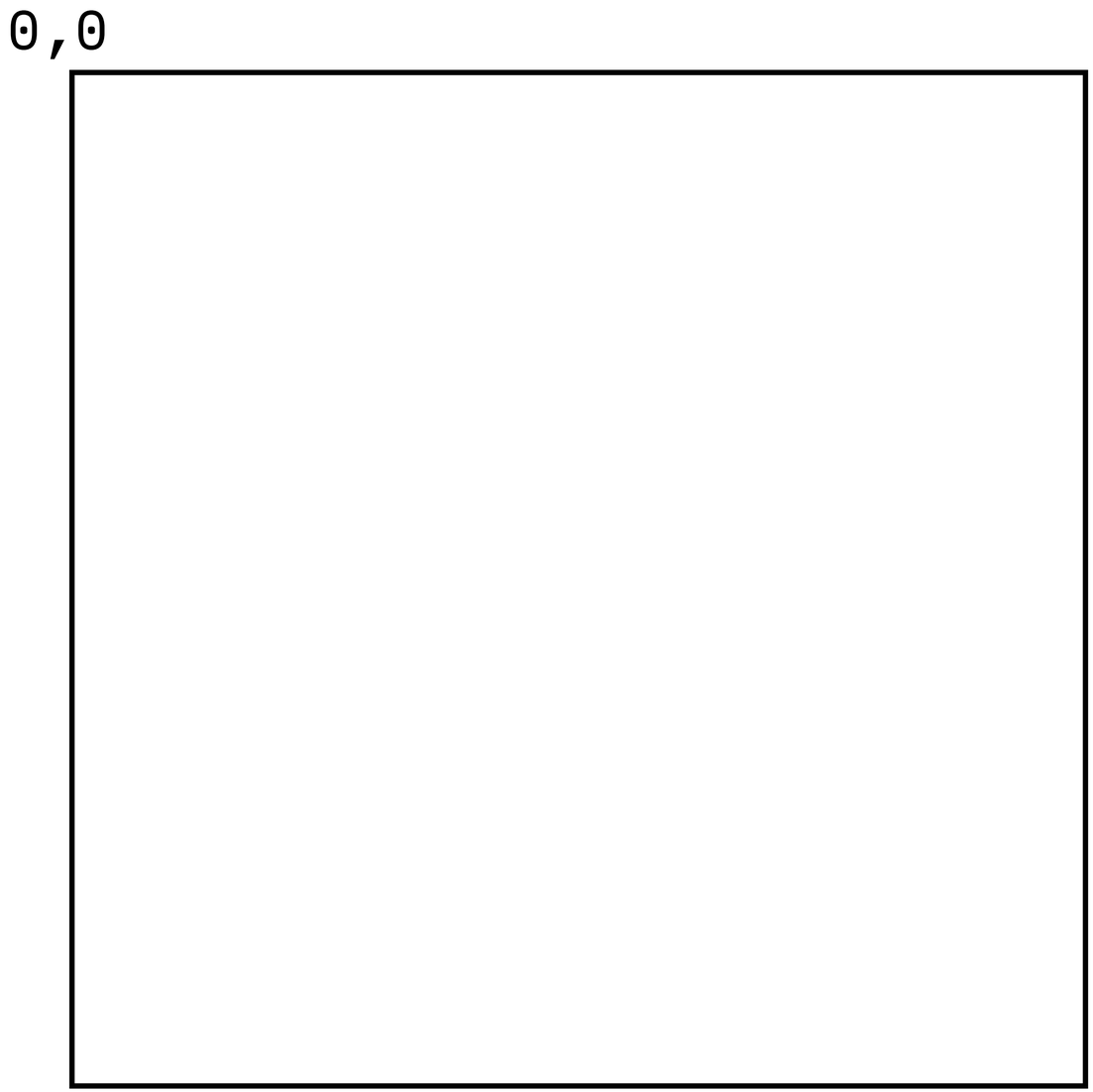
```
void setup() {
  size(400, 400);
}

void draw() {
  translate(20, 20);
  rect(0, 0, 40, 40);
}
```

# 2D Transformationen

```
void setup() {
  size(400, 400);
}

void draw() {
  translate(20, 20);
  rect(0, 0, 40, 40);
}
```

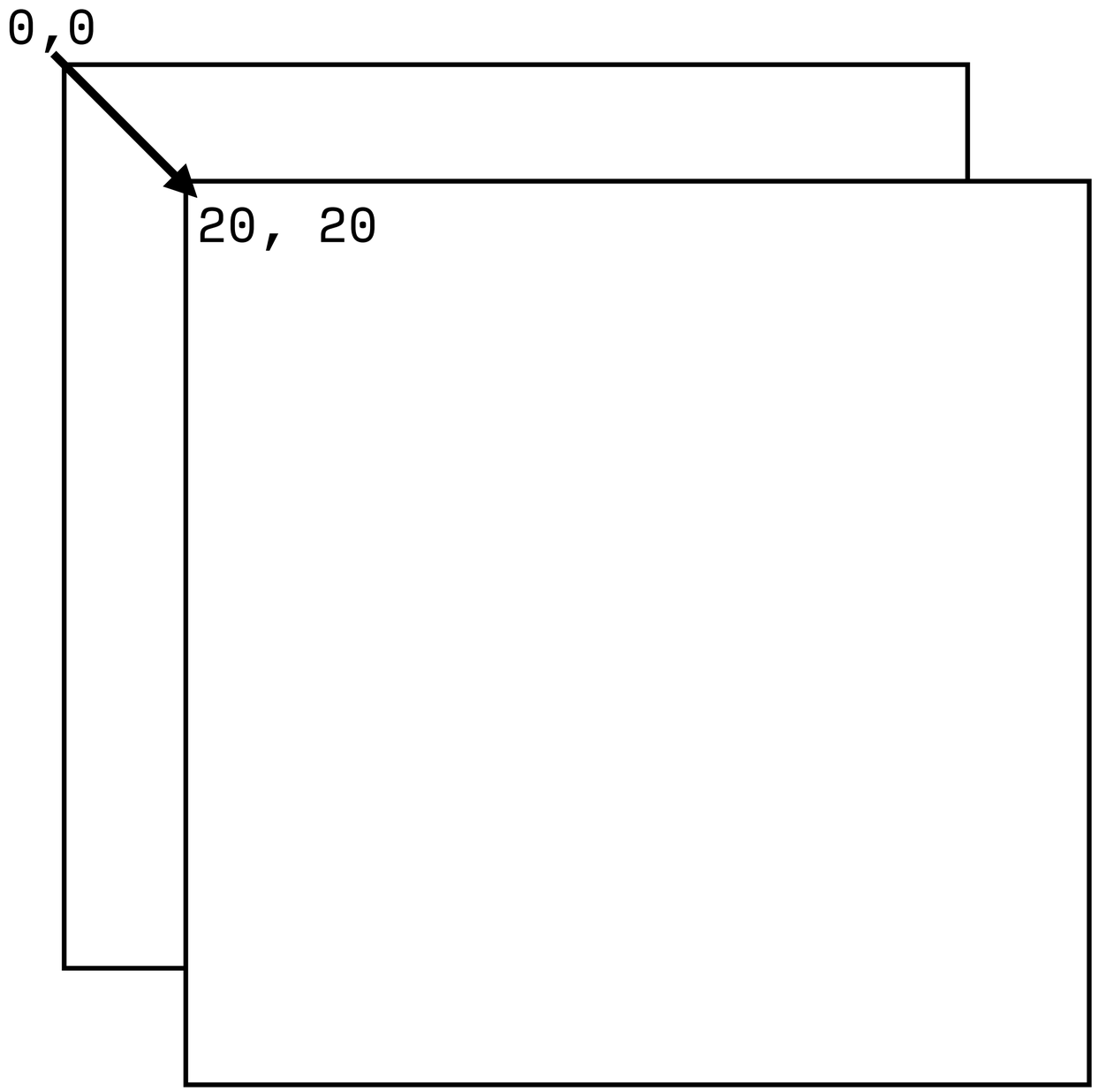
# 2D Transformationen

```
void setup() {
  size(400, 400);
}

void draw() {
  translate(20, 20);
  rect(0, 0, 40, 40);
}
```



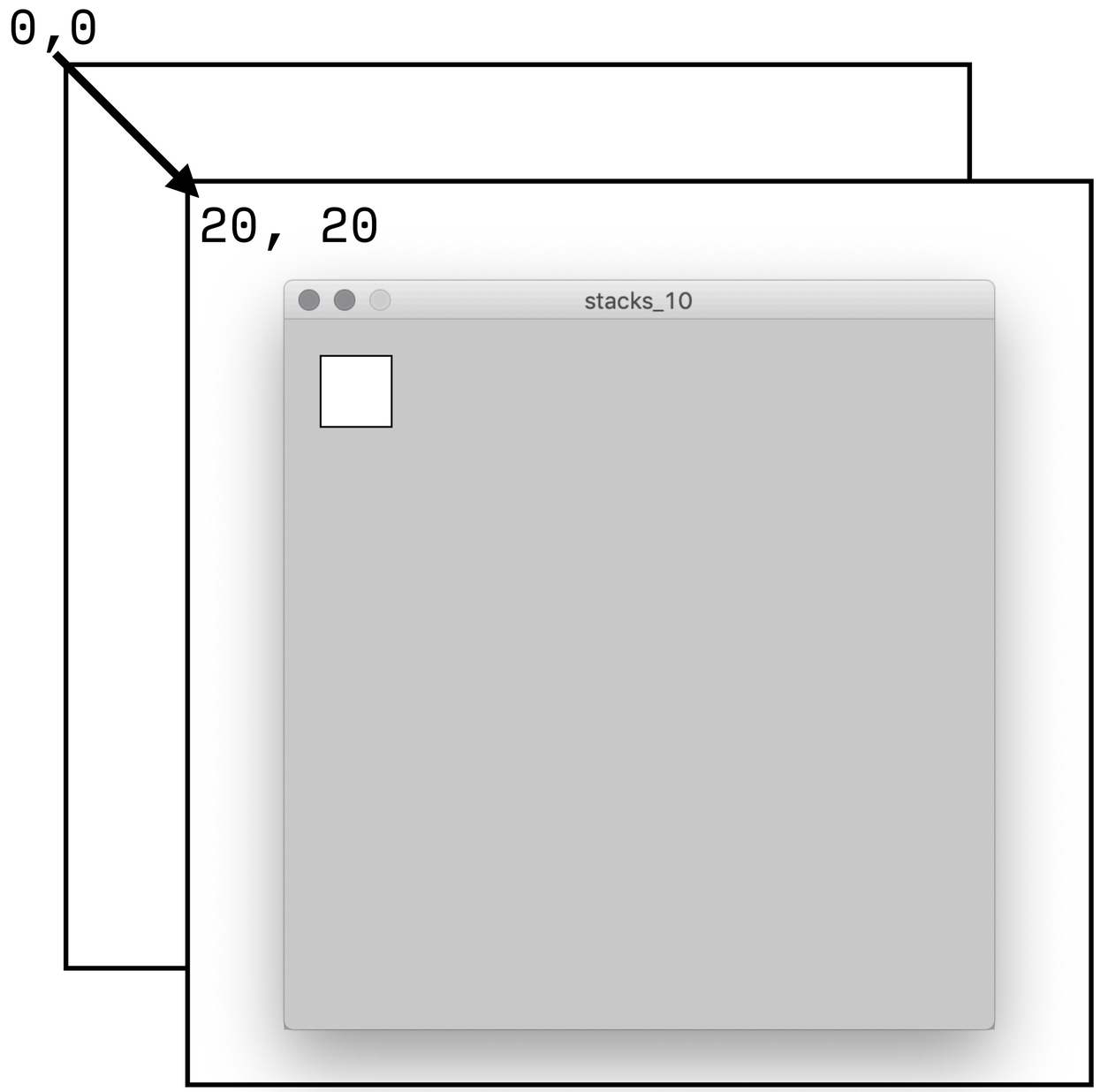

# 2D Transformationen

```
void setup() {
  size(400, 400);
}

void draw() {
  translate(20, 20);
  rect(0, 0, 40, 40);
}
```



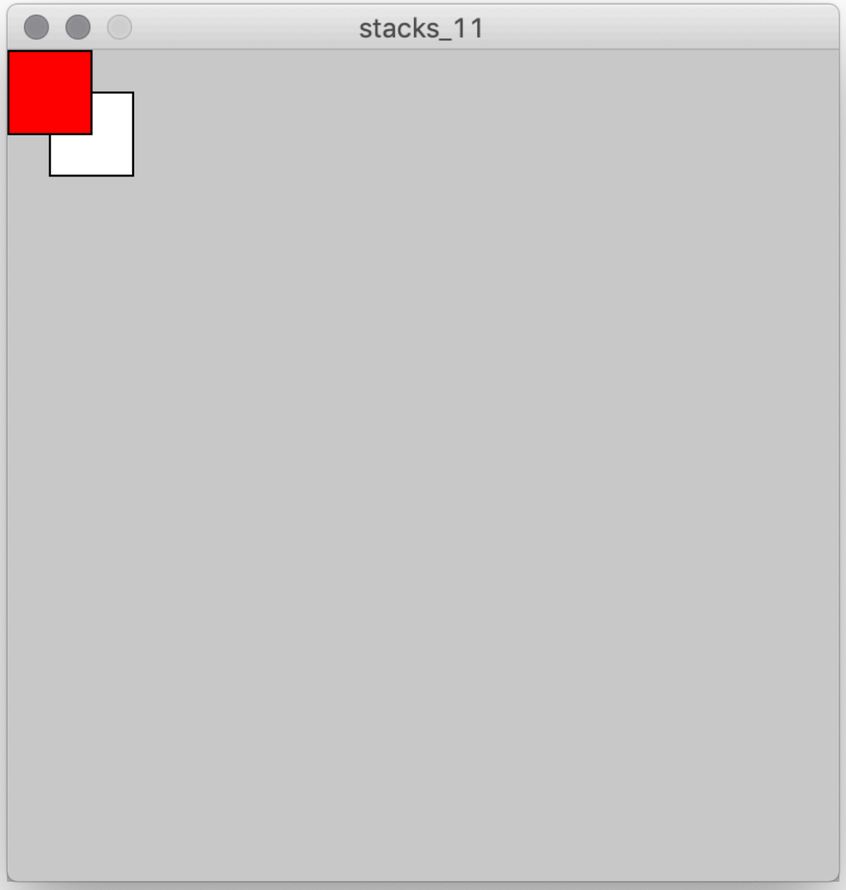

# 2D Transformationen

```
void setup() {
  size(400, 400);
}

void draw() {
  translate(20, 20);
  rect(0, 0, 40, 40);
}
```




# 2D Transformationen

```
void setup() {
  size(400, 400);
}

void draw() {
  pushMatrix();
  translate(20, 20);
  rect(0, 0, 40, 40);
  popMatrix();
  pushStyle();
  fill(255, 0, 0);
  rect(0,0, 40, 40);
  popStyle();
}
```




# 2D Transformationen

```
void translate(posX, posY);
void rotate(angle);
void scale(percentage);
```



# 2D Transformationen

```
void setup() {
  size(200, 200);
  background(255);
  smooth();
  fill(192);
  noStroke();
  rect(40, 40, 40, 40);

  pushMatrix();
  rotate(radians(45));
  fill(0);
  rect(40, 40, 40, 40);
  popMatrix();
}
```

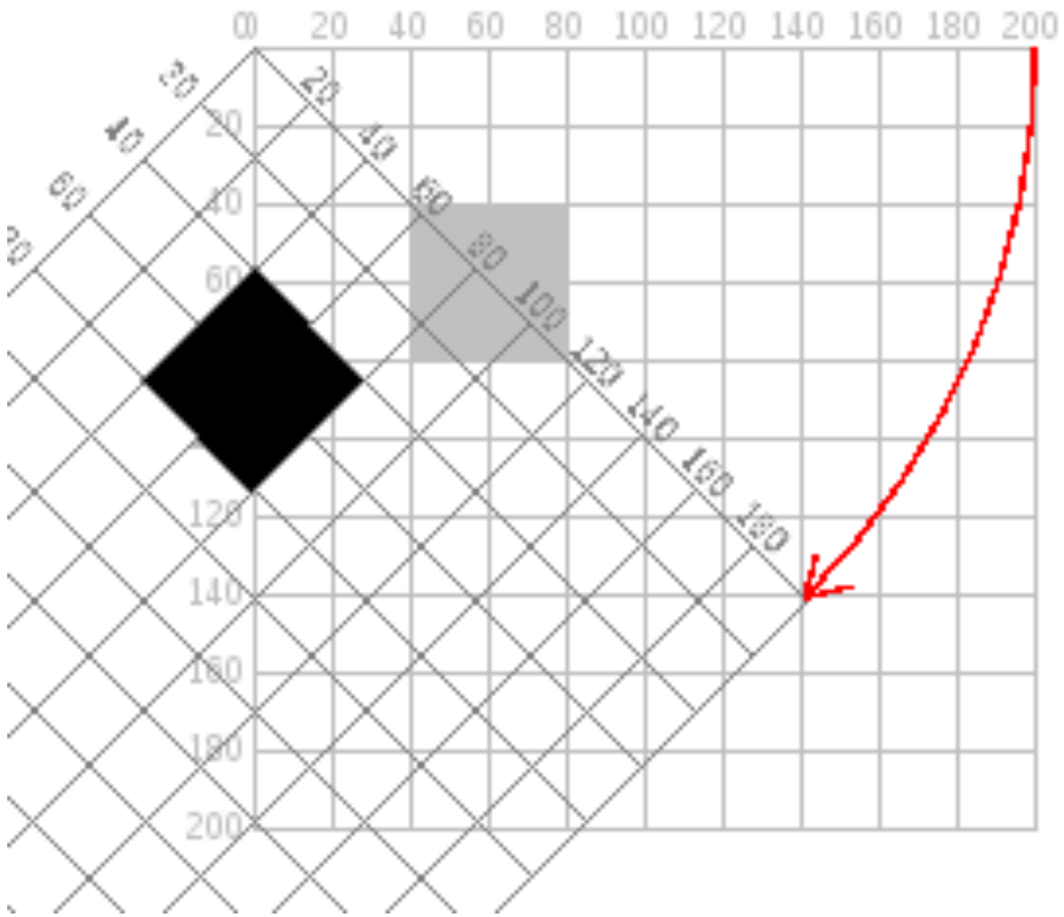
# 2D Transformationen

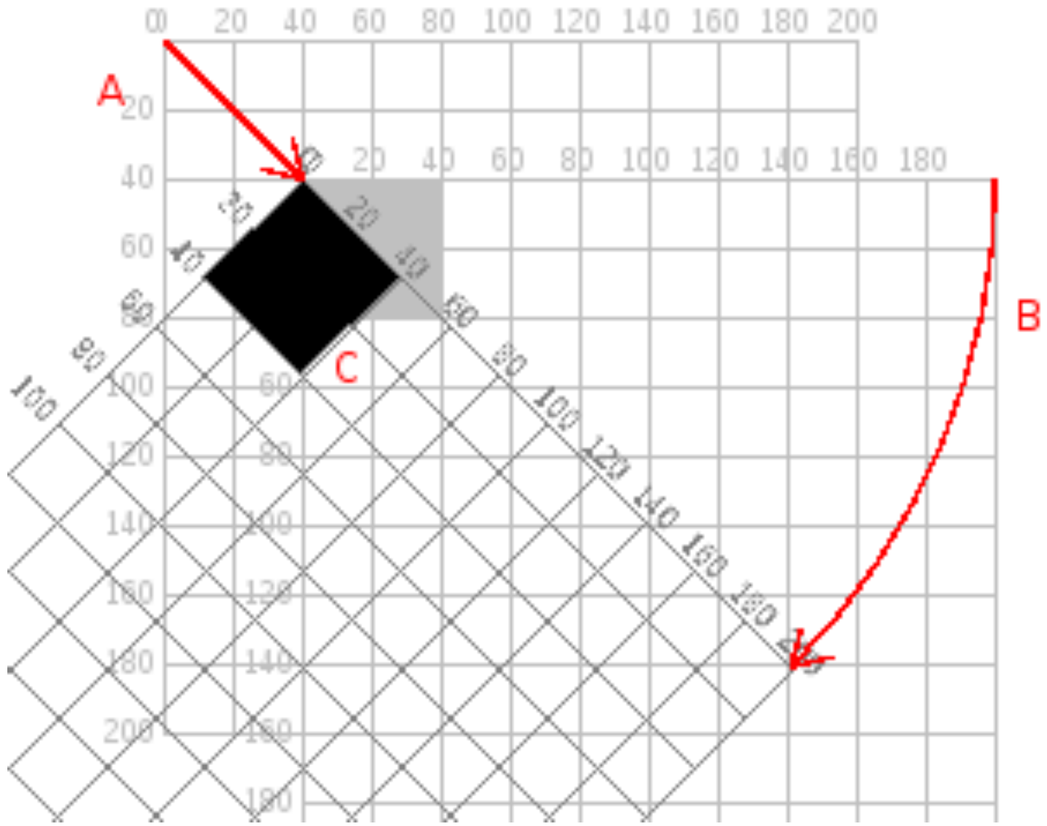
```
void setup() {
  size(200, 200);
  background(255);
  smooth();
  fill(192);
  noStroke();
  rect(40, 40, 40, 40);

  pushMatrix();
  rotate(radians(45));
  fill(0);
  rect(40, 40, 40, 40);
  popMatrix();
}
```

# 2D Transformationen

```
void setup() {
  size(200, 200);
  background(255);
  smooth();
  fill(192);
  noStroke();
  rect(40, 40, 40, 40);

  pushMatrix();
  translate(40, 40);
  rotate(radians(45));
  fill(0);
  rect(0, 0, 40, 40);
  popMatrix();
}
```

# 2D Transformationen






# 2D Transformationen

# PGraphics


# PGraphics

```
PGraphics pg;

void setup() {
  size(600, 600);
  pg = createGraphics(200, 200);

  pg.beginDraw();
  …
  pg.endDraw();

}

void draw() {
  background(255);
  image(pg, width/2, height/2);
}
```

# PGraphics

```
PGraphics pg;

void setup() {
  size(600, 600);
  pg = createGraphics(200, 200);

  pg.beginDraw();
  …
  pg.endDraw();

}

void draw() {
  background(255);
  image(pg, width/2, height/2);
}
```

# PGraphics

```
PGraphics pg;

void setup() {
  size(600, 600);
  pg = createGraphics(200, 200);

  pg.beginDraw();
  …
  pg.endDraw();

}

void draw() {
  background(255);
  image(pg, width/2, height/2);
}
```

# PGraphics

```
PImage p;

void setup() {
  size(600, 600);
  p = loadImage("coolStuff.jpg");
}

void draw() {
  background(255);
  image(p, width/2, height/2);
}
```

# PGraphics

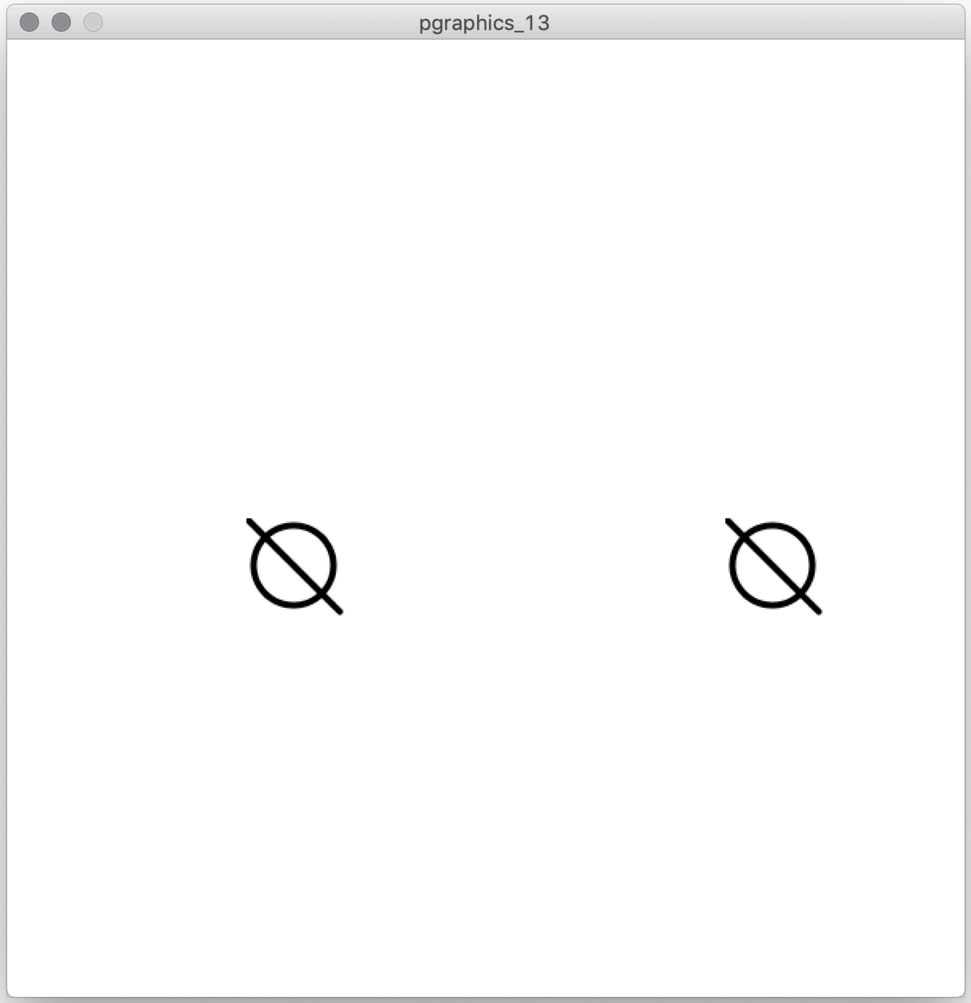
```
PGraphics pg;

void setup() {
  size(600, 600);
  pg = createGraphics(200, 200);

  pg.beginDraw();
  pg.ellipseMode(CORNER);
  pg.noFill();
  pg.strokeWeight(4);
  pg.circle(4,4, 50); // korrektur für strokeWeight
  pg.line(0,0, 58, 58); // und hier
  pg.endDraw();

}

void draw() {
  background(255);
  image(pg, width/2, height/2);
  image(pg, width/2, 50);
}
```

# PGraphics

```
PGraphics pg;
void setup() {
  size(600, 600);
  pg = createGraphics(200, 200);

  pg.beginDraw();
  pg.ellipseMode(CORNER);
  pg.noFill();
  pg.strokeWeight(4);
  pg.circle(4,4, 50); // korrektur für strokeWeight
  pg.line(0,0, 58, 58); // und hier
  pg.endDraw();
}

void draw() {
  background(255);
  image(pg, width/2 + 150, height/2);
  image(pg, width/2 - 150, height/2);
}
```

# PGraphics

```
PGraphics pg;
void setup() {
  size(600, 600);
  pg = createGraphics(200, 200);

  pg.beginDraw();
  pg.ellipseMode(CORNER);
  pg.noFill();
  pg.strokeWeight(4);
  pg.circle(4,4, 50); // korrektur für strokeWeight
  pg.line(0,0, 58, 58); // und hier
  pg.endDraw();
}

void draw() {
  background(255);
  image(pg, width/2 + 150, height/2);
  image(pg, width/2 - 150, height/2);
}
```




# PGraphics

**Nehmt ein Raster von der letzten Aufgabe und erweitert es:**
**– Erstellt min. 3 Klassen mit Funktionalität**
**– benutzt ArrayListen**
**– vermeidet es auf den Canvas direkt zu zeichnen (PGraphics)**
**– bringt Bewegung in's Spiel (s. Auto, mouseX/ mouseY – was gibt es noch?)**
**- Wie kann das Raster aufgebrochen werden?**

Aufgabe 3 – Bis 15.11.2019

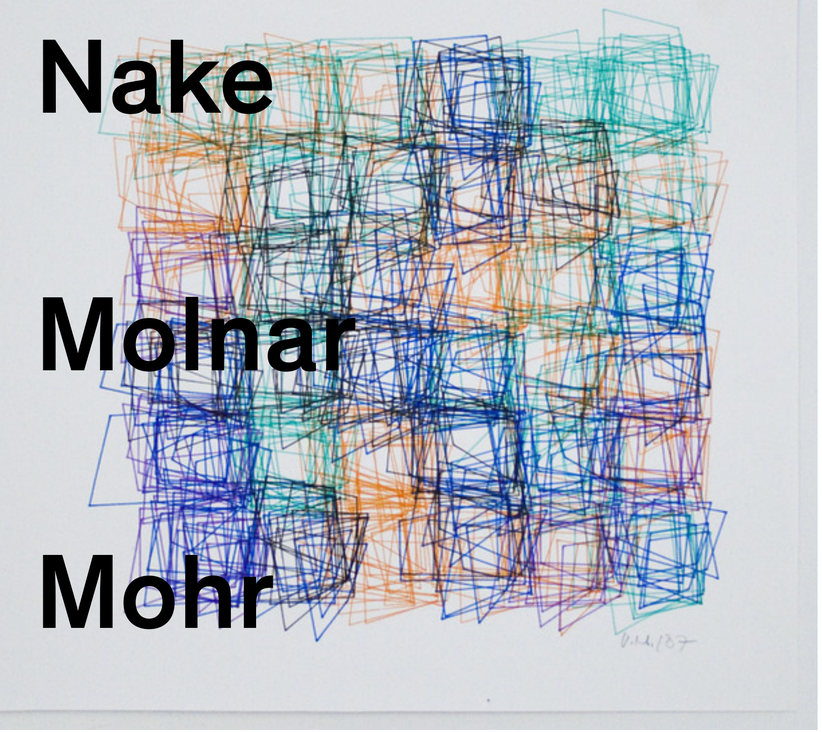
**Dokumentiert die Ausgabe + Code (alles als Zip)**

**Benutzt auch das Tutorium von Elias.**

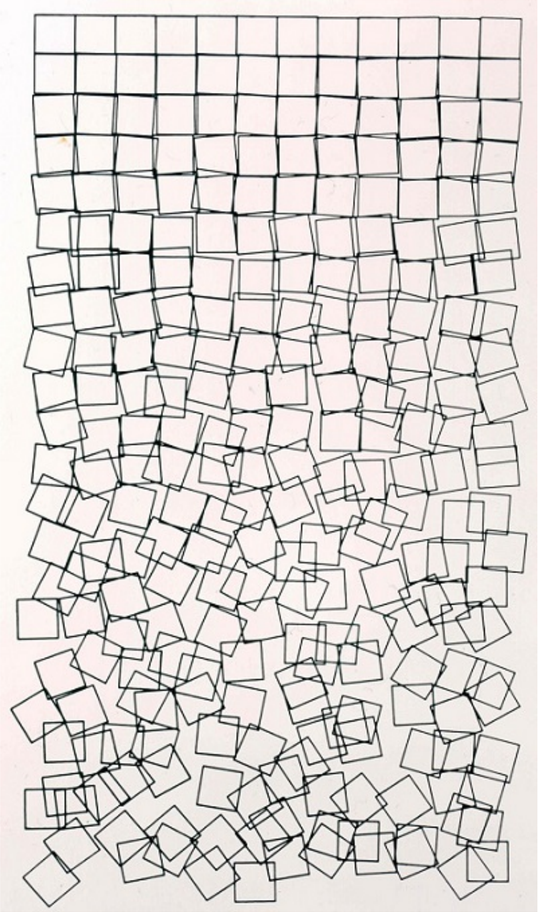
**https://processing.org/reference/**

Aufgabe 3 – Bis 15.11.2019

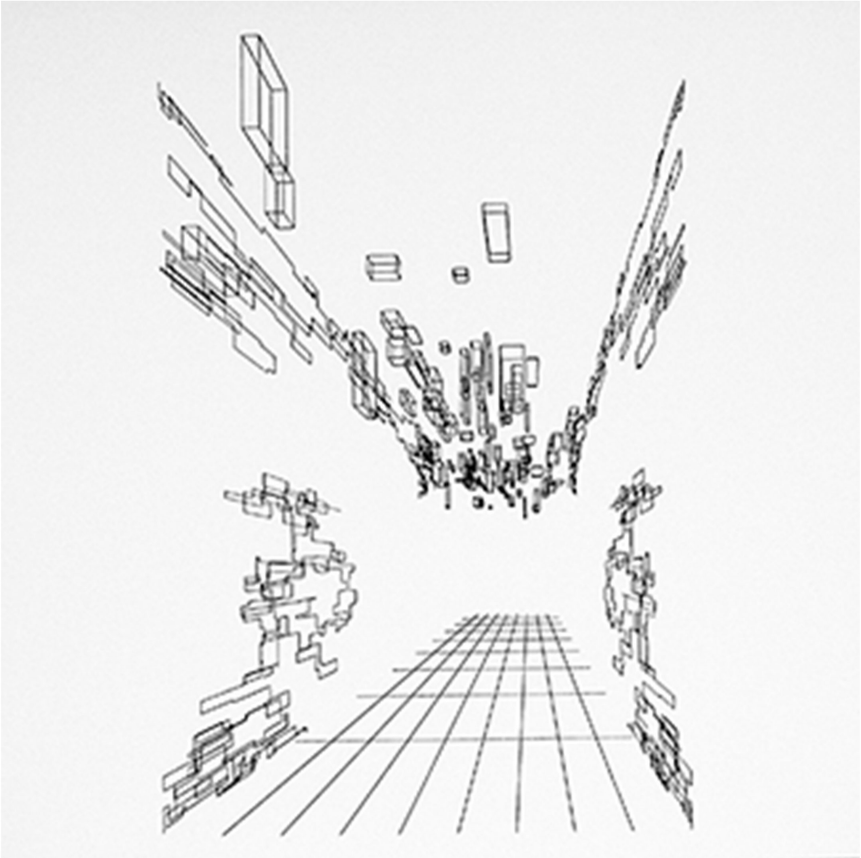
**Nake**



**Molnar**



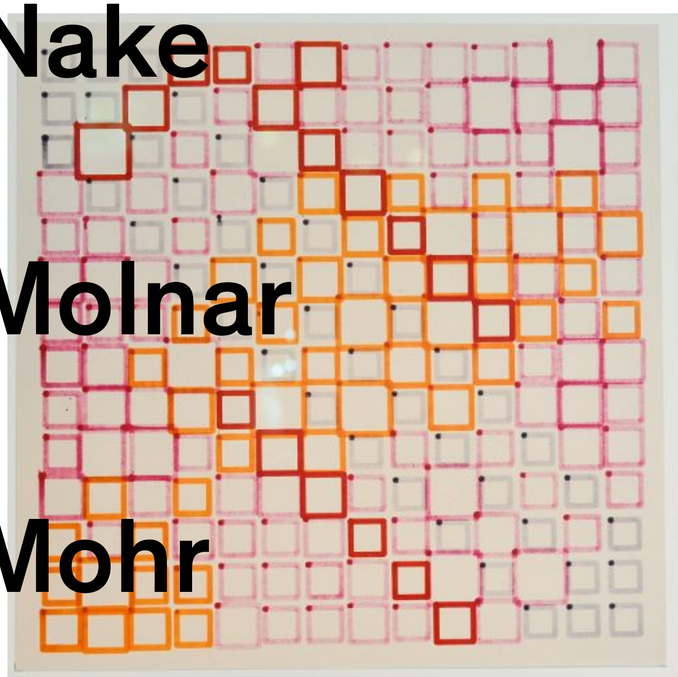
**Mohr**



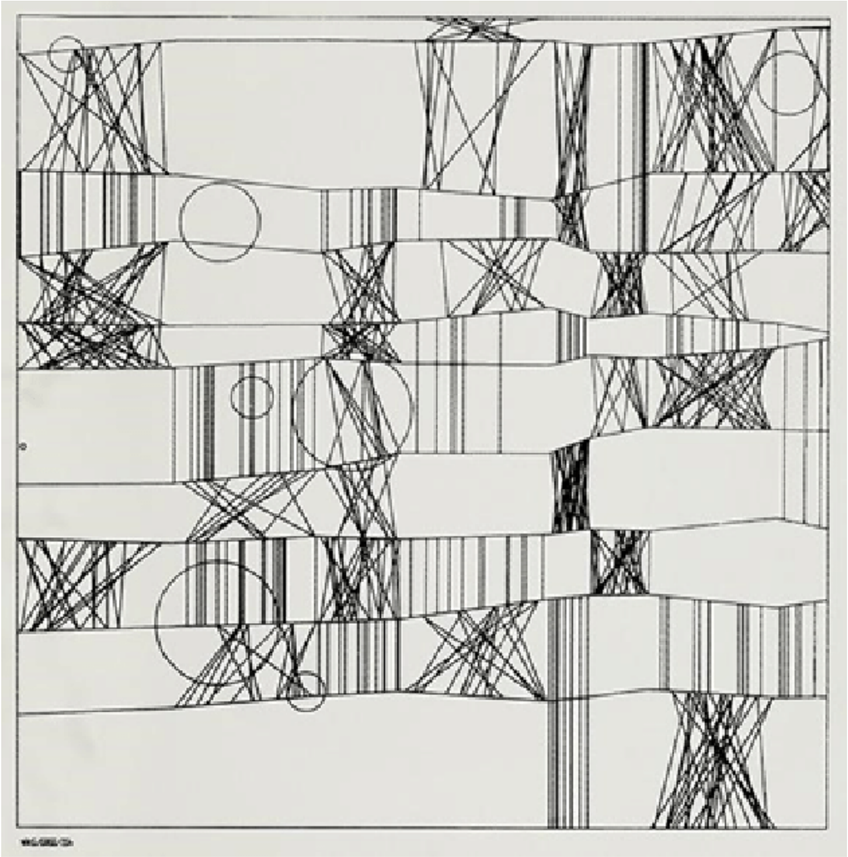
**Nees**

# Aufgabe 3 – Bis 15.11.2019

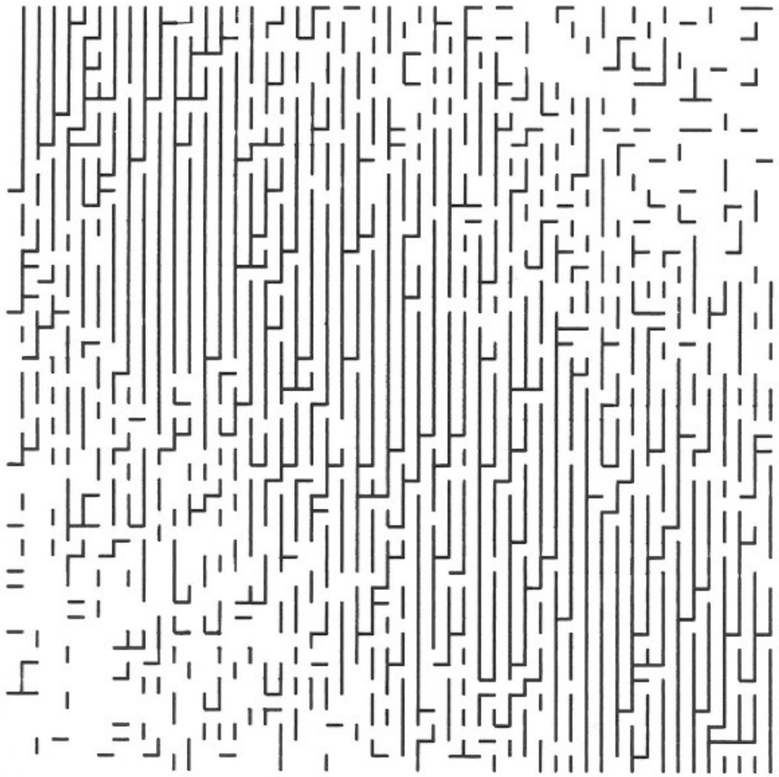
**Nake**



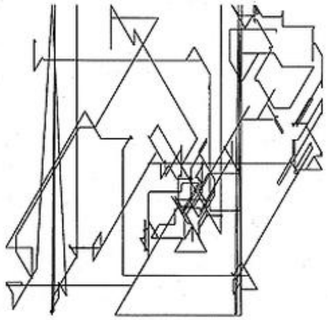
**Molnar**



**Mohr**



**Nees**



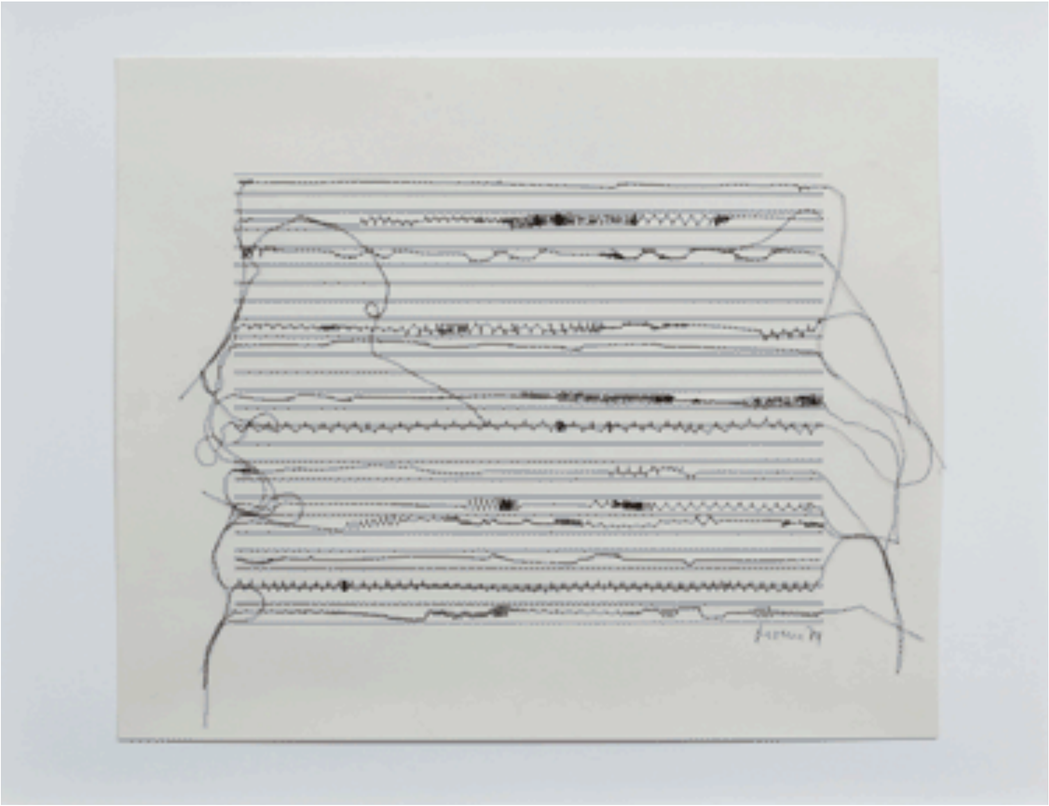
# Aufgabe 3 – Bis 15.11.2019

**Nake**

**Molnar**

**Mohr**

**Nees**



Aufgabe 3 – Bis 15.11.2019