

EINFÜHRUNG
SOFTWAREENTWICKLUNG
Hochschule Augsburg
Fakultät Informatik
Wintersemester 2014/2015

STEPHAN BATTEIGER
+49 (0) 8131 779979-1
stephan.batteiger@peerigon.com
peerigon.com

EINFÜHRUNG SOFTWAREENTWICKLUNG
Hochschule Augsburg, 2014

WIR SUCHEN DIE MIETWOHNUNGSVERMITTLUNG

2013/14 Herbst
Winter

DasTelefonbuch

Deutschland Alles in einem



Ärzte Augenoptik	A
Bauunter- nehmen Blumen	B
Cafés Computer Container	C
Dentallabor Druckereien	D
Elektronik Energie Ergotherapie	E
Farben Fliesen Fußpflege	F
Gaststätten Gläserien	G
Heilpraktiker Hörgeräte Hotels	H
Immobilien Ingenieur- büros	I
Justiz- behörden Juweliere	J
Kliniken Kosmetik	K
Logopädie Lohnsteuer	L
Medien- technik Metallbau	M
Nachhilfe Nagelstudio Notare	N

ANZAHL DER STUDENTEN IM RAUM?

VON LINKS OBEN NACH RECHTS UNTEN

1. Die erste Person in der linken hinteren Ecke steht auf, sagt laut EINS und setzt sich wieder hin.
2. Die Person daneben steht auf und addiert EINS dazu und sagt laut die neue Zahl.

ANZAHL DER STUDENTEN IM RAUM?

EINZELNE REIHEN

1. Jede Person, die LINKS in der Reihe sitzt, steht auf und sagt laut EINS und setzt sich wieder hin.
2. Die Person daneben steht auf und addiert EINS dazu und sagt laut die neue Zahl.
3. Wenn das Ergebnis der einzelnen Reihen feststeht, dann sagt die letzte Person in der letzten Reihe ihr Ergebnis LAUT und setzt sich wieder hin.
4. Die Person in der Reihe davor steht auf, addiert ihr Ergebnis dazu, sagt das neue Ergebnis laut und setzt sich wieder hin.

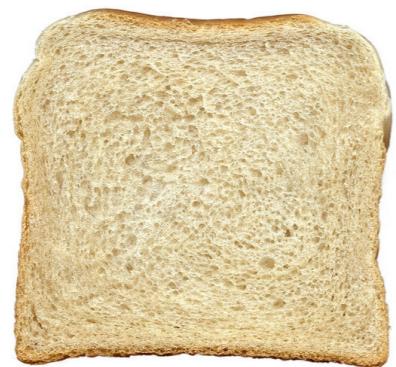
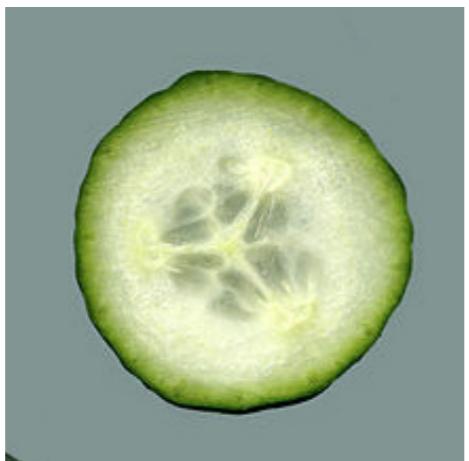
ANZAHL DER STUDENTEN IM RAUM?

ALLE ZUSAMMEN

1. Alle stehen einmal auf und denken an die Zahl 1.
2. Schließe dich mit jemand anderem zusammen, der steht, addiere die Zahlen zusammen und merke dir die neue Zahl.
3. Einer von euch sollte sich hinsetzen. Der andere soll mit Schritt zwei weitermachen.

DAS REZEPT

DARAUS...



... WIRD EIN SANDWICH:



DER ALGORITHMUS

„Ein Algorithmus ist eine endliche, genau definierte Handlungsvorschrift zur schrittweisen Lösung eines Problems oder einer Klasse von Problemen.“

vgl. GDI-Skript Prof. Schöler

EINFÜHRUNG
SOFTWAREENTWICKLUNG
Hochschule Augsburg
Fakultät Informatik
Wintersemester 2014/2015

STEPHAN BATTEIGER
+49 (0) 8131 779979-1
stephan.batteiger@peerigon.com
peerigon.com

JETZT LERNEN WIR PROGRAMMIEREN
... mit Processing

WAS IST PROCESSING

- grafische Programmiersprache
- schlanke Entwicklungsumgebung
- hat ihren Ursprung bei Java
- leichter Einstieg



WEITERFÜHRENDE INFORMATIONEN

ONLINE

<https://processing.org/>

<http://processing.org/reference/>

PRINT

Processing, O'reillys Basics, Erik Bartmann

<https://processing.org/>



[Cover](#)

[Download](#)

[Exhibition](#)

[Reference](#)

[Libraries](#)

[Tools](#)

[Environment](#)

[Tutorials](#)

[Examples](#)

[Books](#)

[Overview](#)

[People](#)

[Foundation](#)

[Shop](#)

[» Forum](#)

[» GitHub](#)

[» Issues](#)

[» Wiki](#)

[Download
Windows.](#)



[» Github](#)
[» Report Bug](#)
[» Wiki](#)
[» Supported](#)

[Stable Rel](#)

REILLY

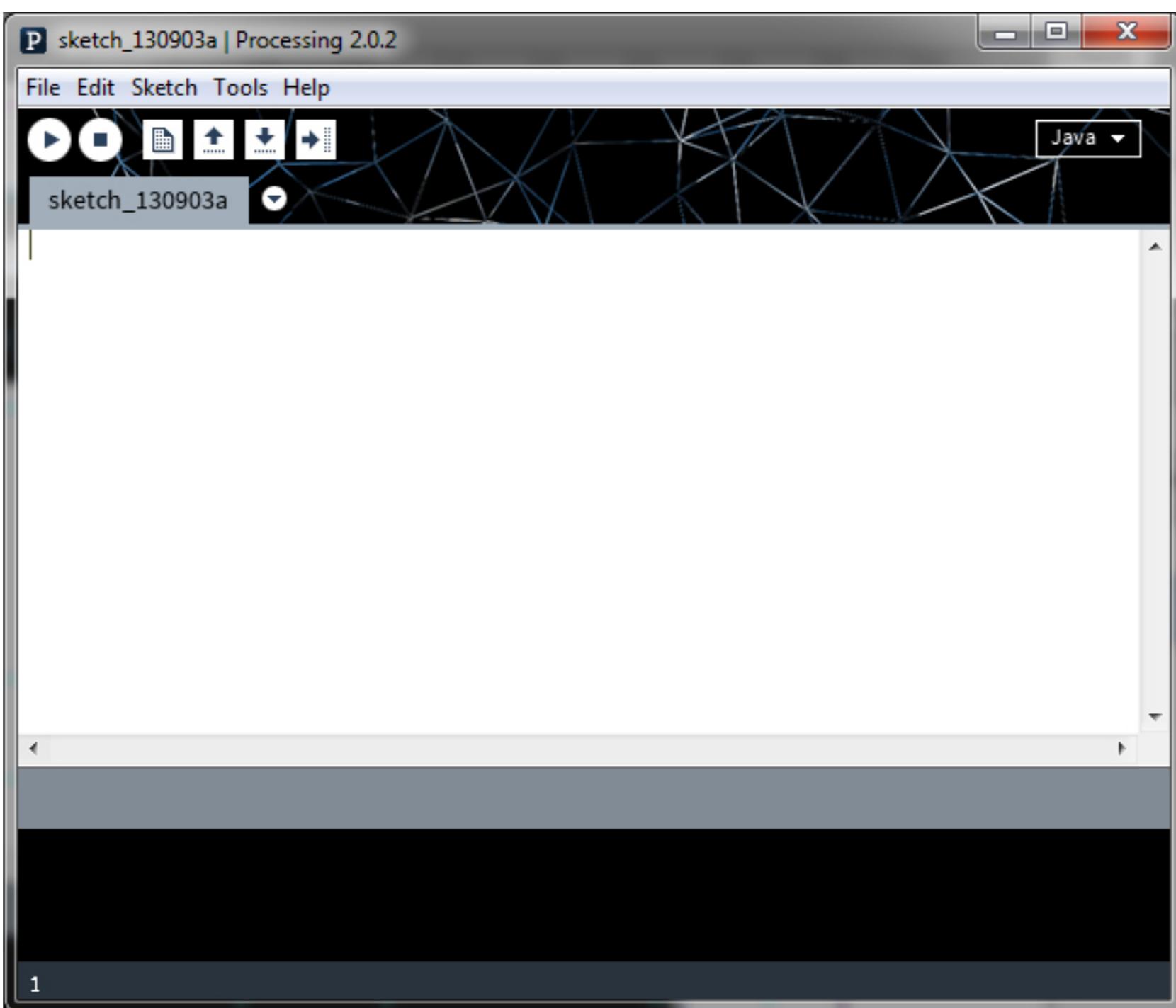
Processing



► Kreativ programmieren mit Processing

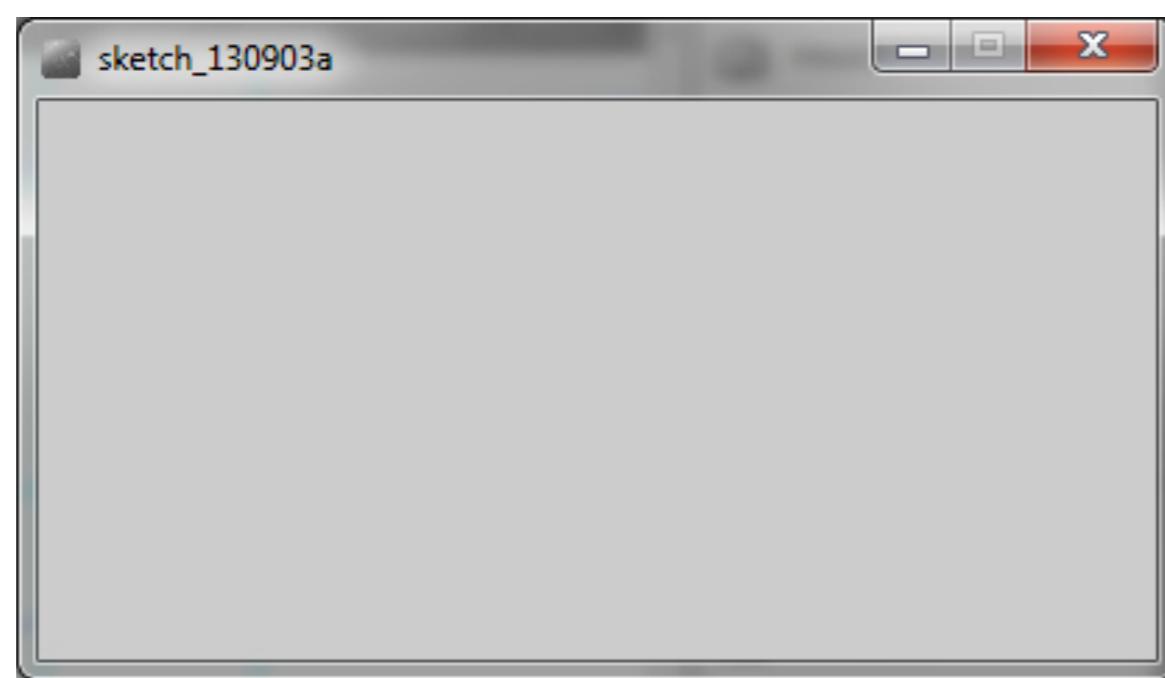
► Für Design, Künste und Spaß

DIE PROCESSING UI

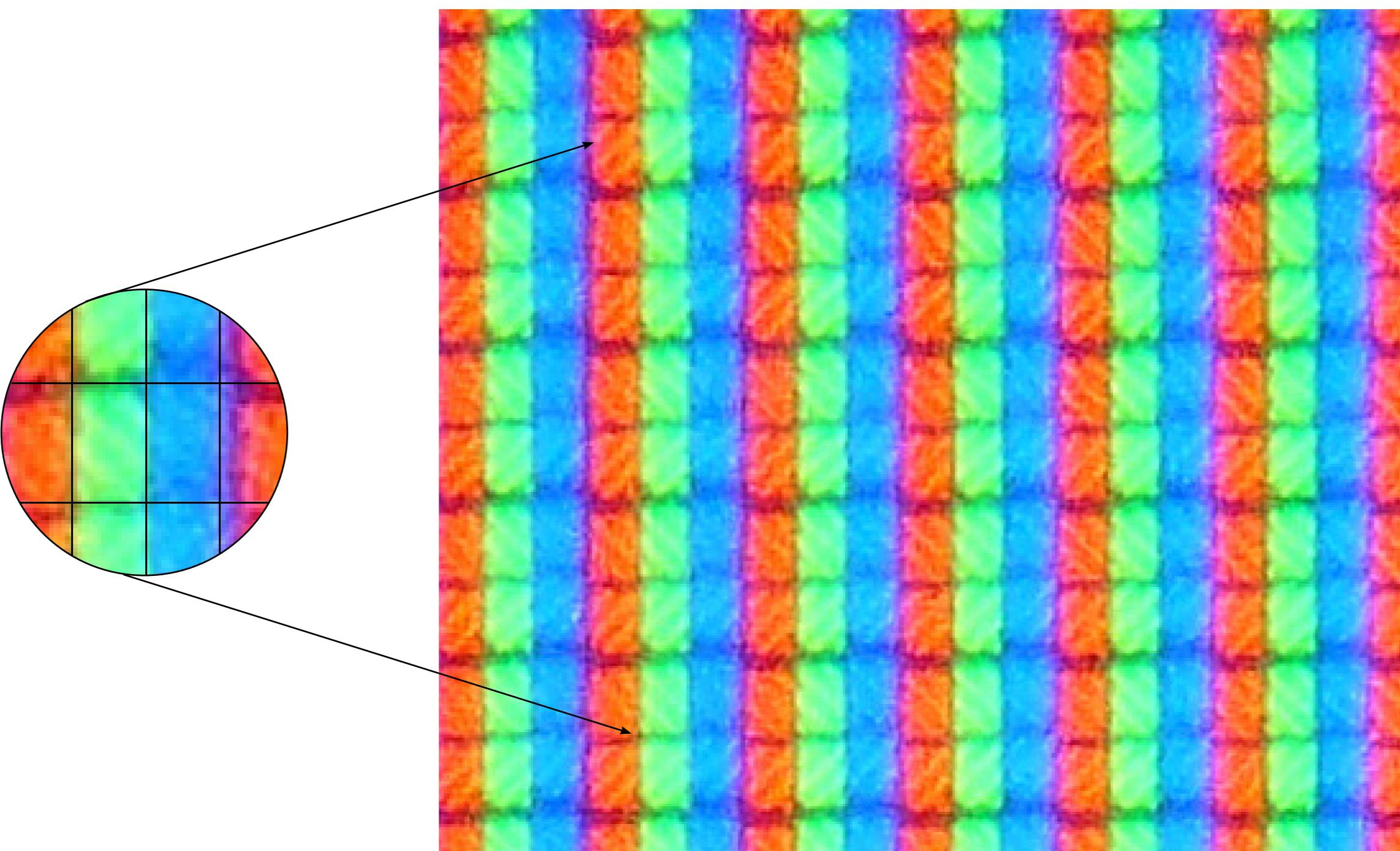


ZEICHENGRUNDLAGE

```
size(400, 200);  
befehlsname(parameterBreite, parameterHoehe);
```



WAS IST EIN PIXEL?



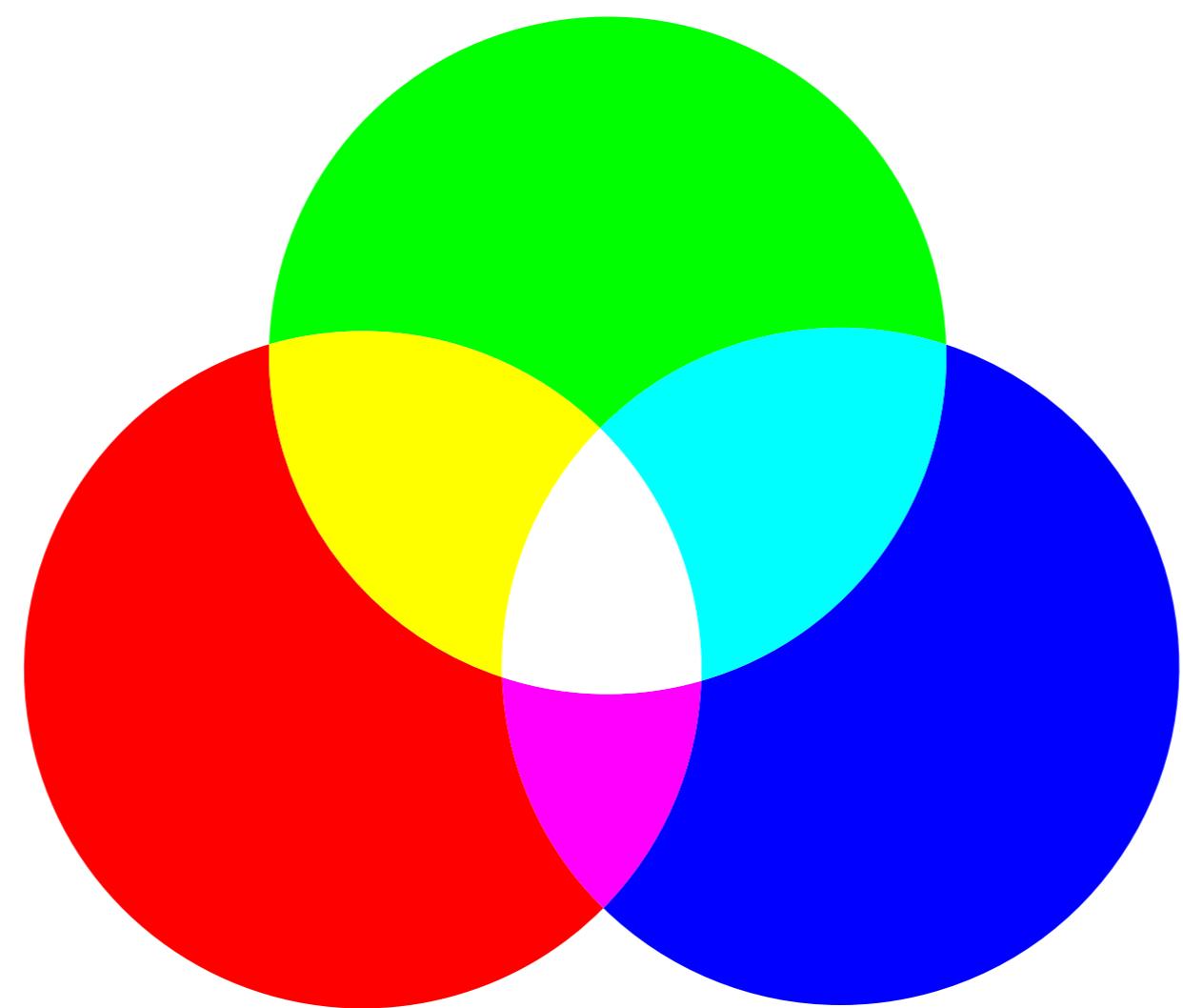
Quelle: Wikipedia

ÜBUNG: SIZE();

- Starte Processing
- Was passiert, wenn ich size(400,200); eingebe?
- Was passiert, wenn ich size(200,400); eingebe?
- Was passiert, wenn ich das Semikolon am Schluss weglasse?
- Für was dient dieses Semikolon?
- Was passiert, wenn ich siZe(400,200); eingebe?
- Was ist eine IDE?
- Warum ist in der IDE der Befehlsname farbig?
- Wann ist der Befehlsname nicht farbig?

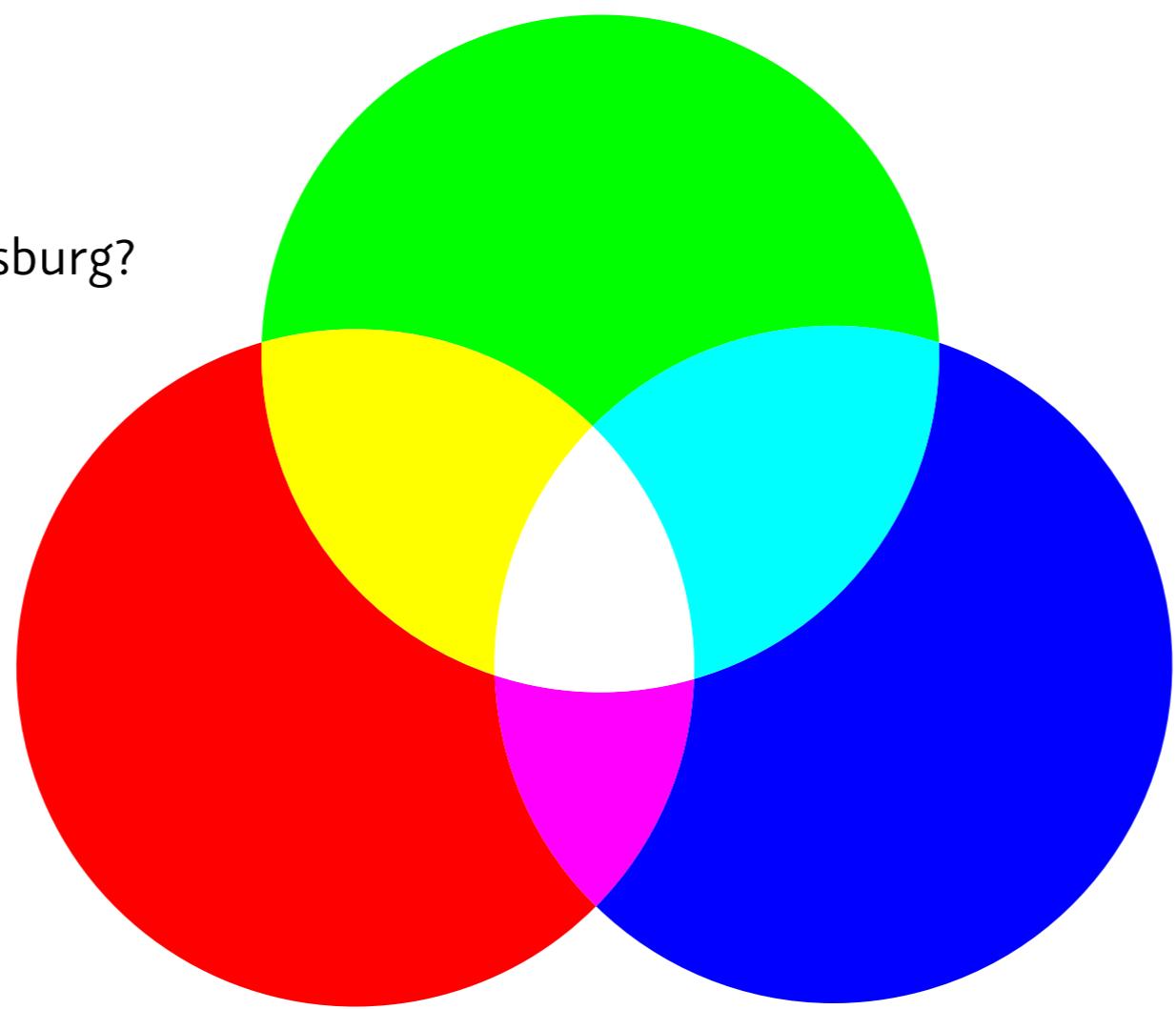
HINTERGRUNDFARBE - BACKGROUND-COLOR

```
background(125, 230, 85);  
background(rot, grün, blau);
```



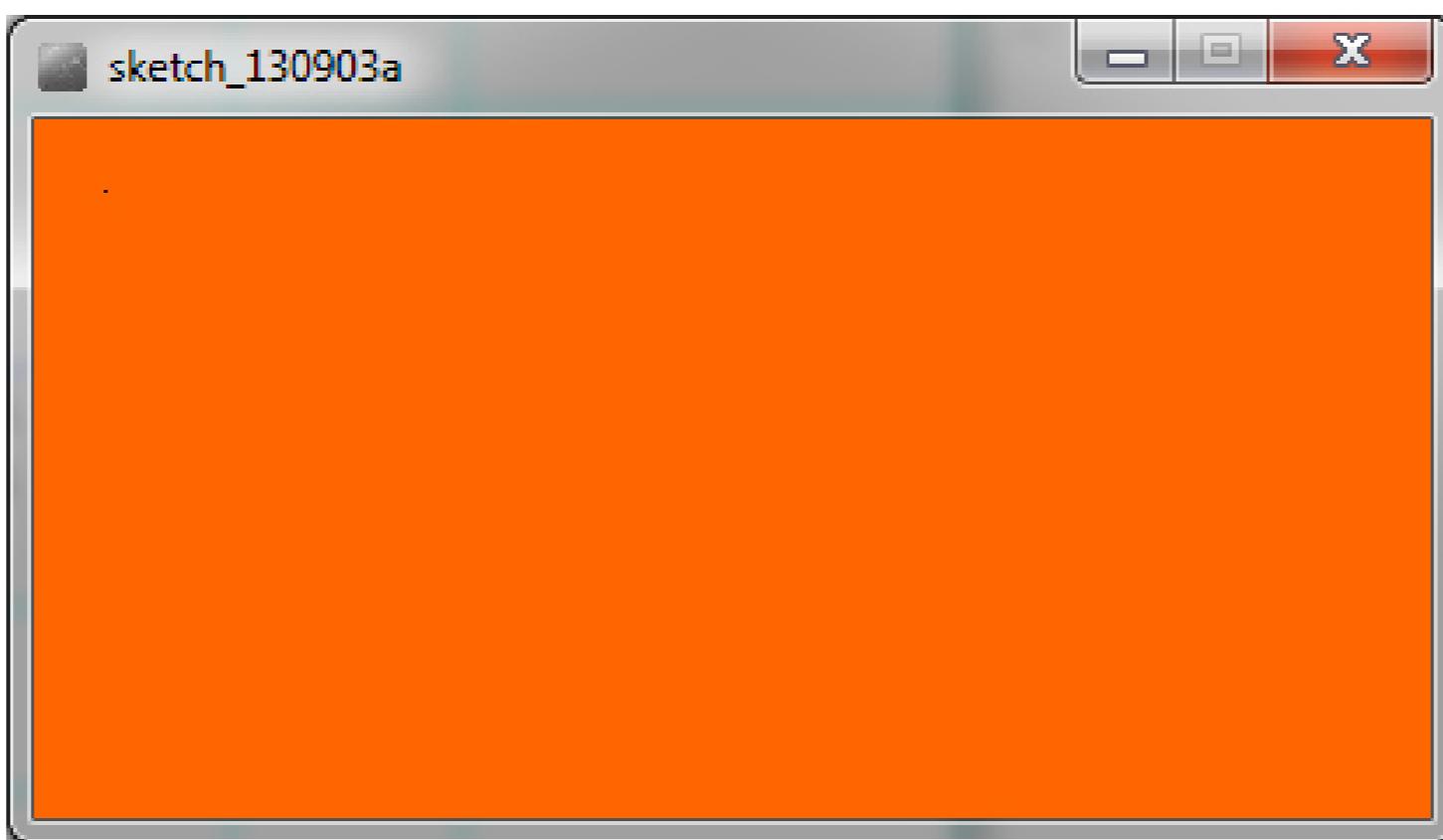
ÜBUNG: BACKGROUND();

- Was ist deine Lieblingsfarbe?
 - Was ist der RGB-Wert deiner Lieblingsfarbe?
 - Ändere den Hintergrund deiner Zeichenfläche!
-
- Was ist der offizielle RGB-Wert der Hochschule Augsburg?
 - Ändere den Hintergrund deiner Zeichenfläche!



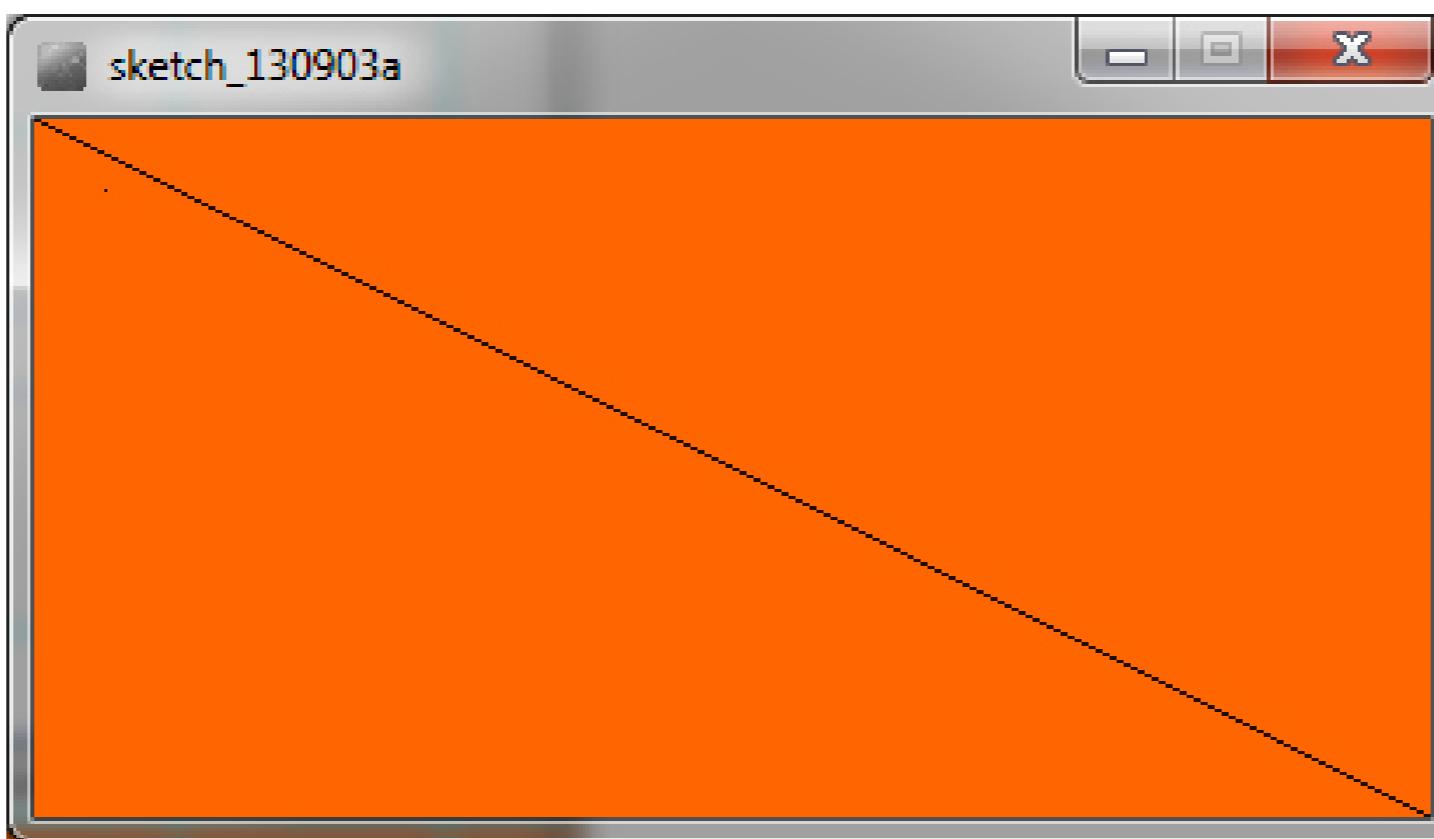
POINT();

```
point(20,20);  
point(x,y);
```



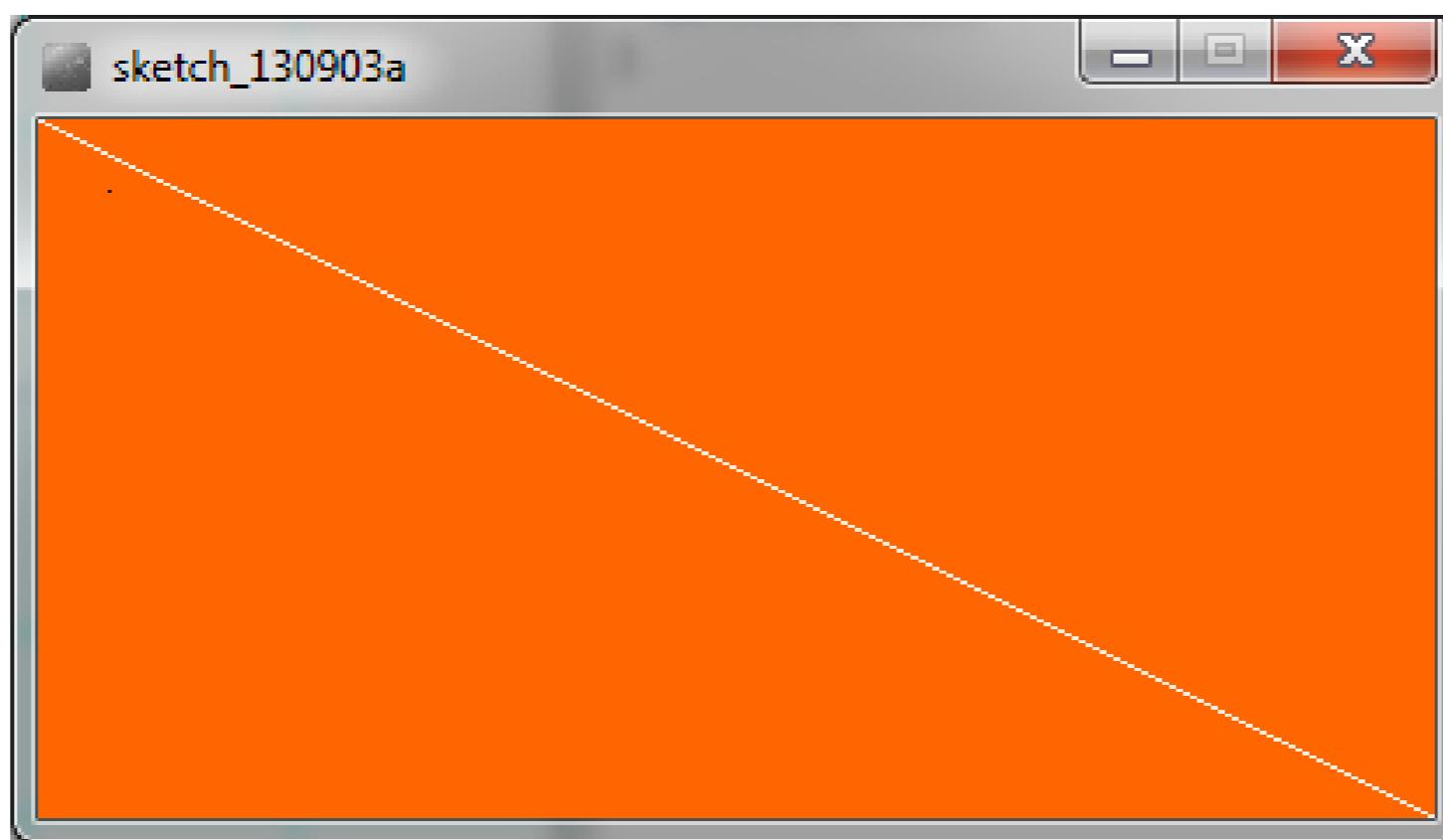
ÜBUNG: POINT();

- Probiere `point(20,20);` aus!
- Versuche diese Linie nachzubauen!



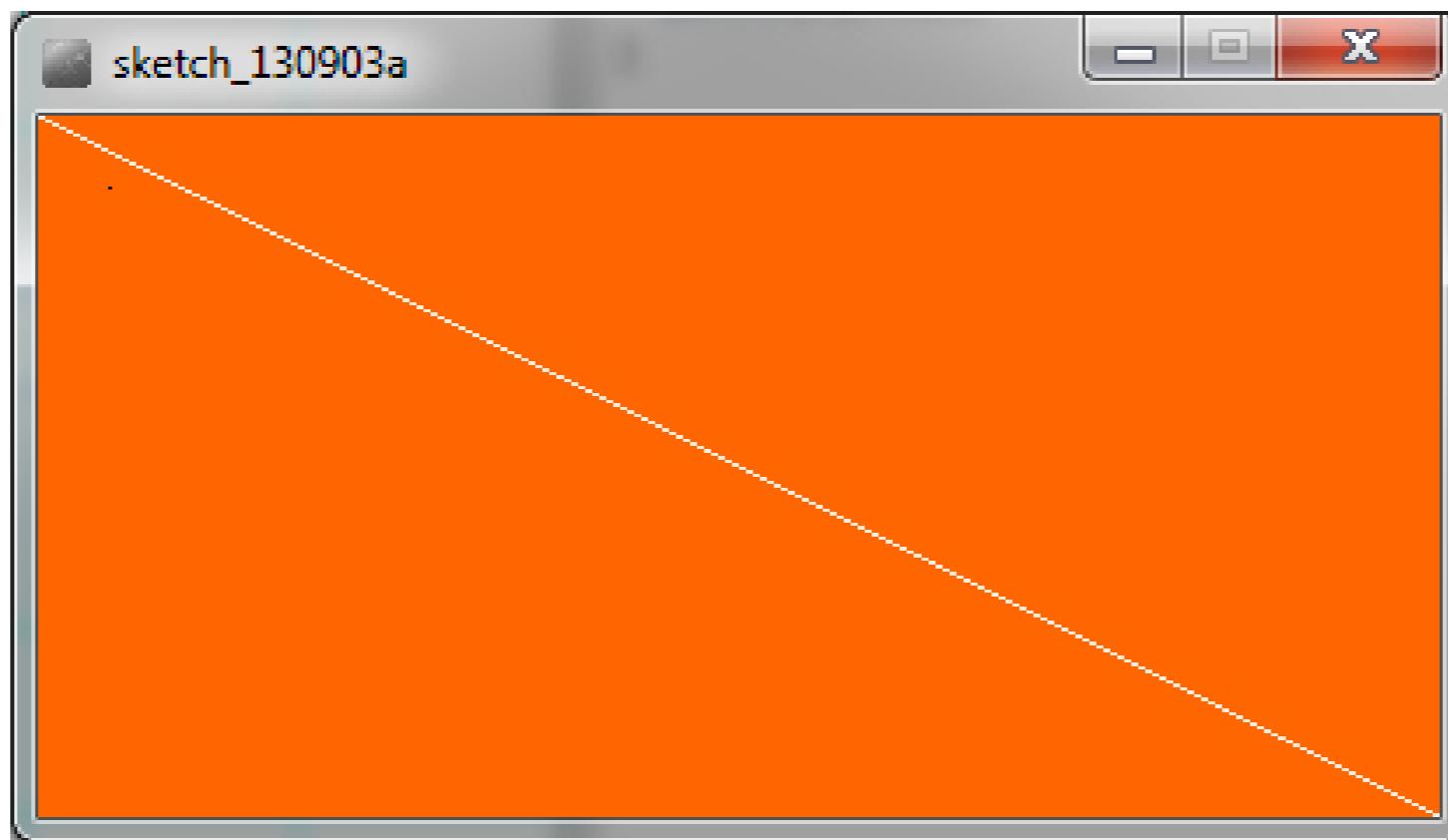
LINE();

```
line(x1, y1, x2, y2);
line(0, 0, 100, 100);
```



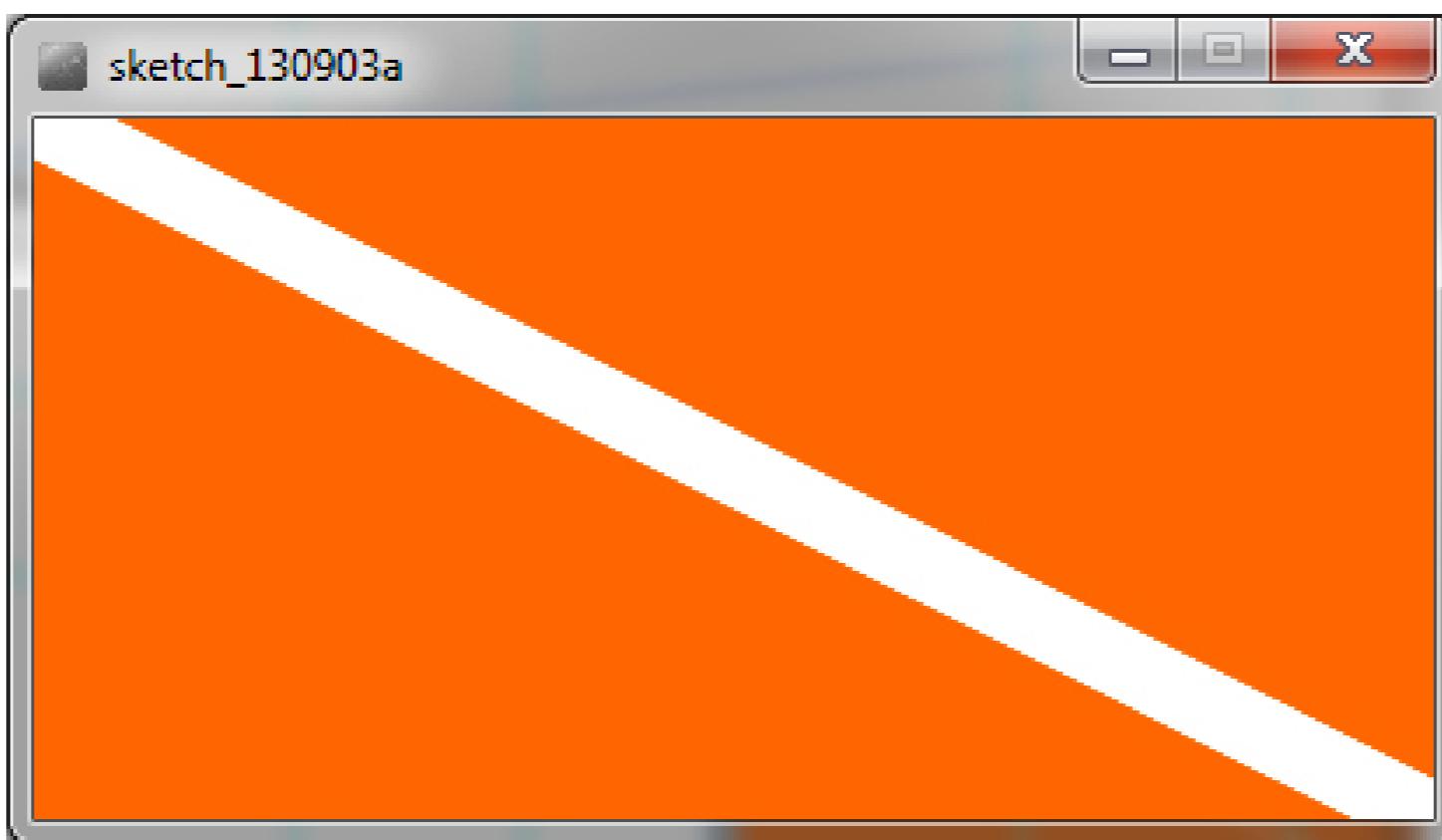
ZEICHENFARBE: STROKE();

```
stroke(255,255,255);  
stroke(rot, grün, blau);
```



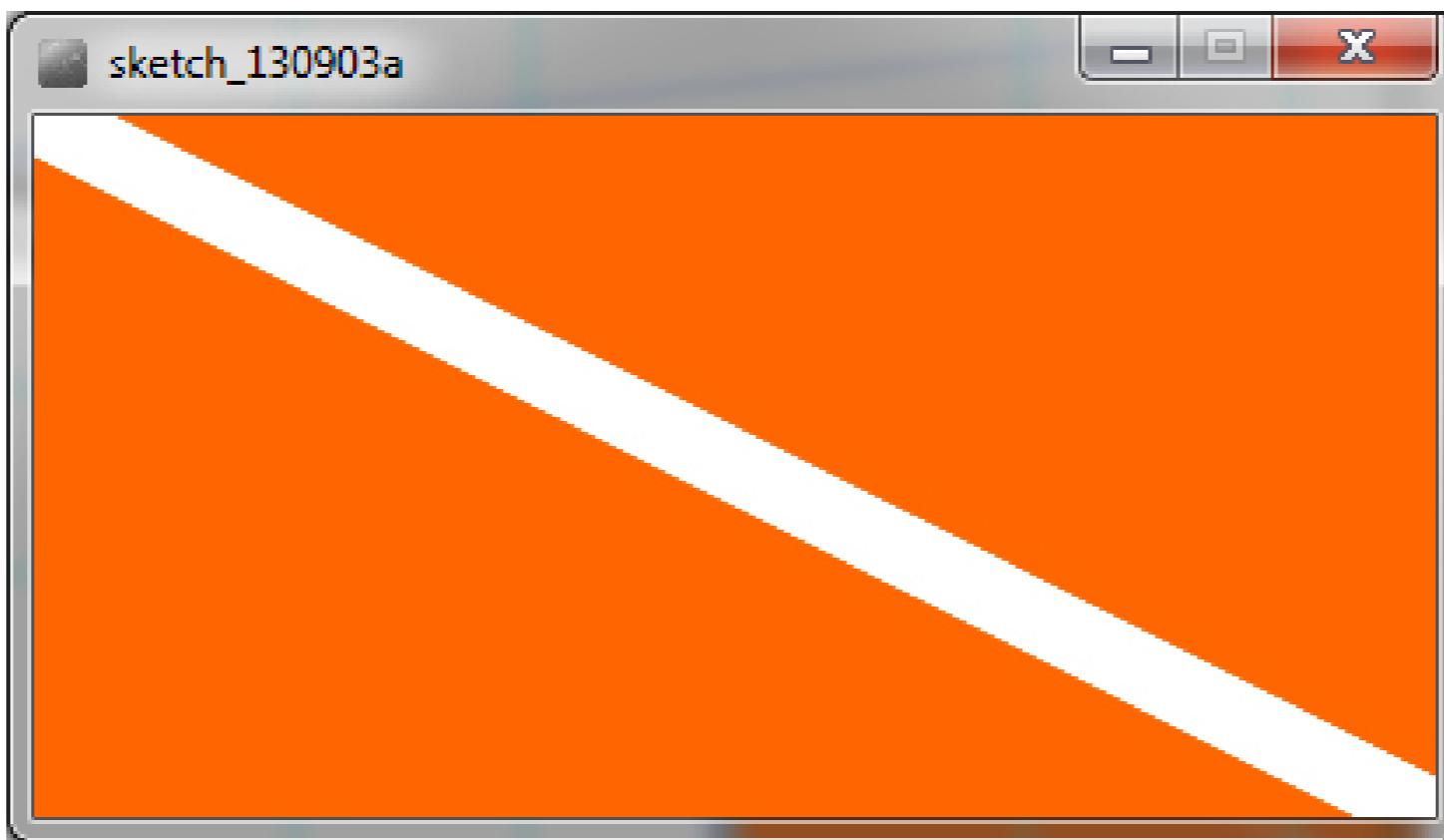
PUNKTGRÖSSE: STROKEWEIGHT();

```
strokeWeight(20);  
strokeWeight(anzahlPixel);
```



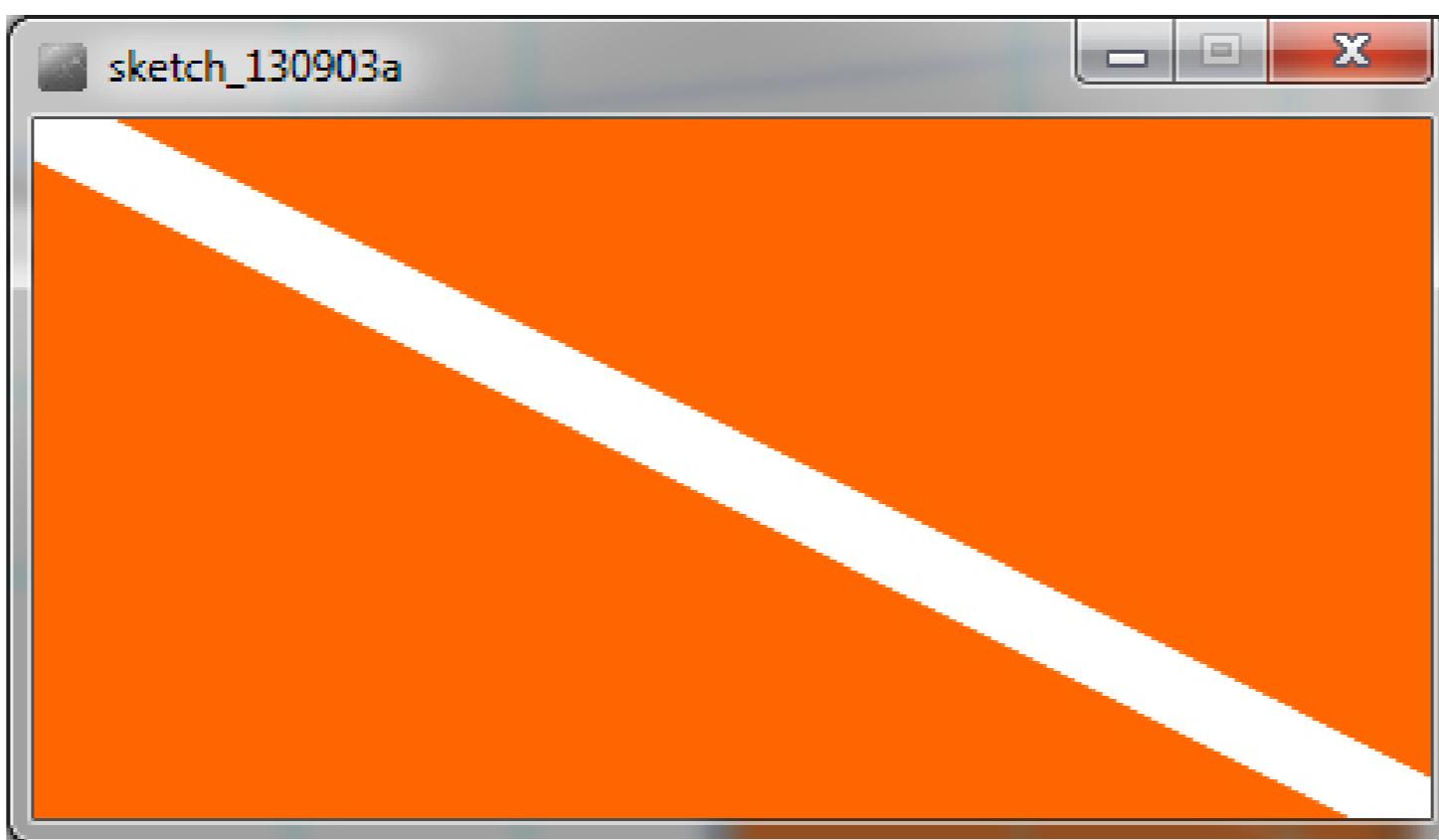
PUNKTGRÖSSE: STROKEWEIGHT();

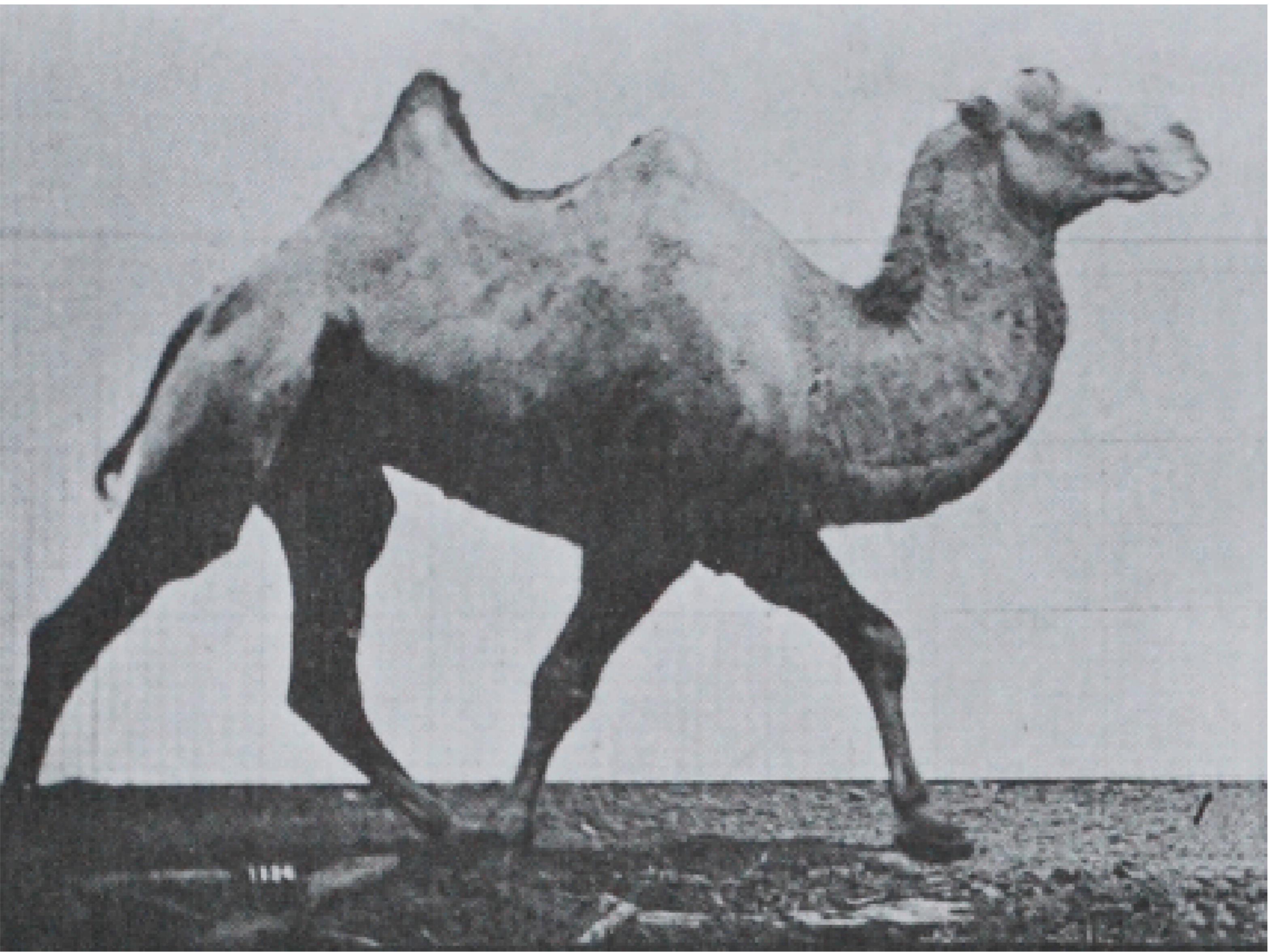
```
strokeWeight(20);  
strokeWeight(anzahlPixel);
```



ÜBUNG: STROKEWEIGHT(); & LINE();

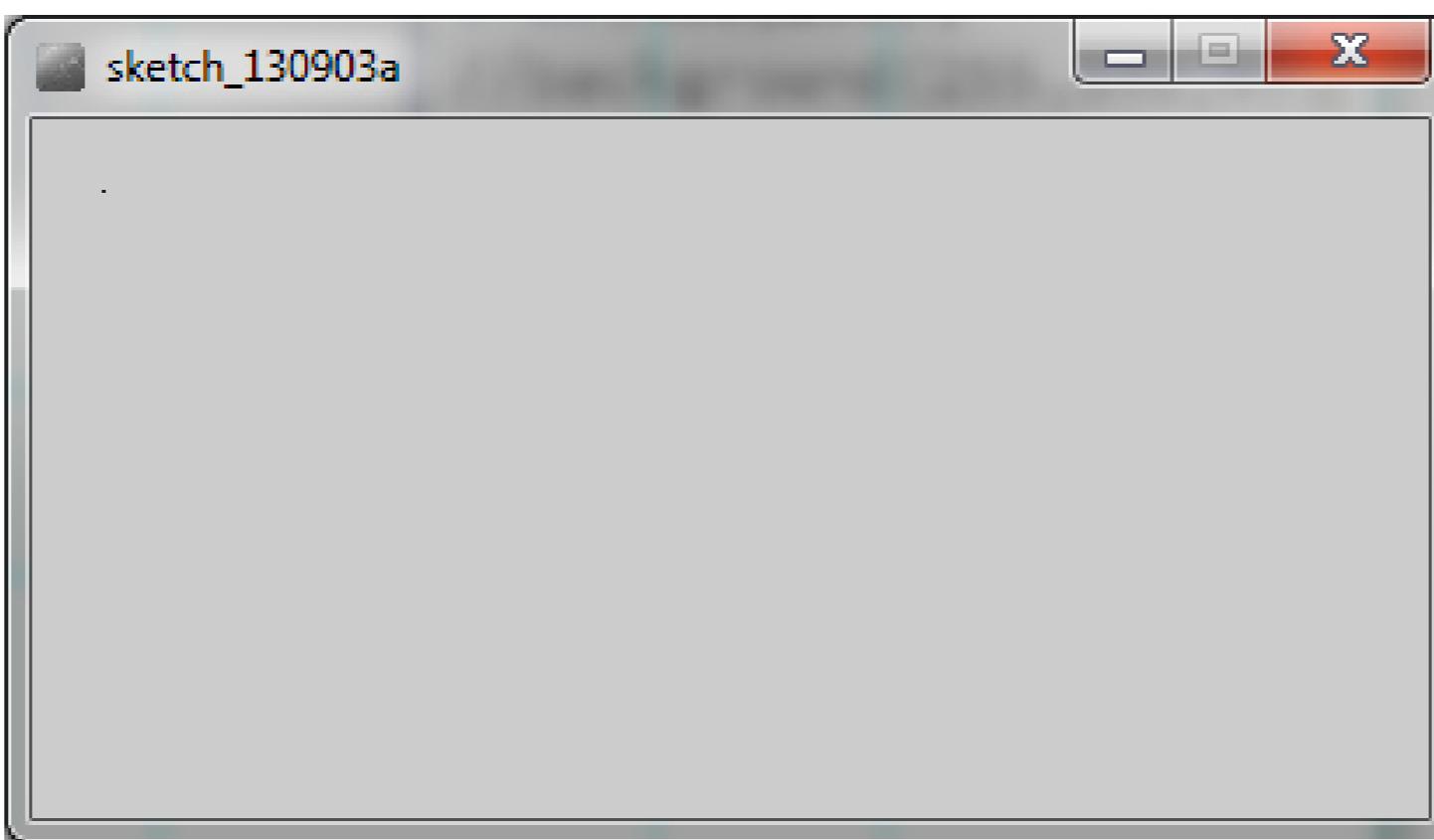
- Teste `strokeWeight(20);`!
- Was bedeutet: “CamelCase”?





Quelle: Wikipedia

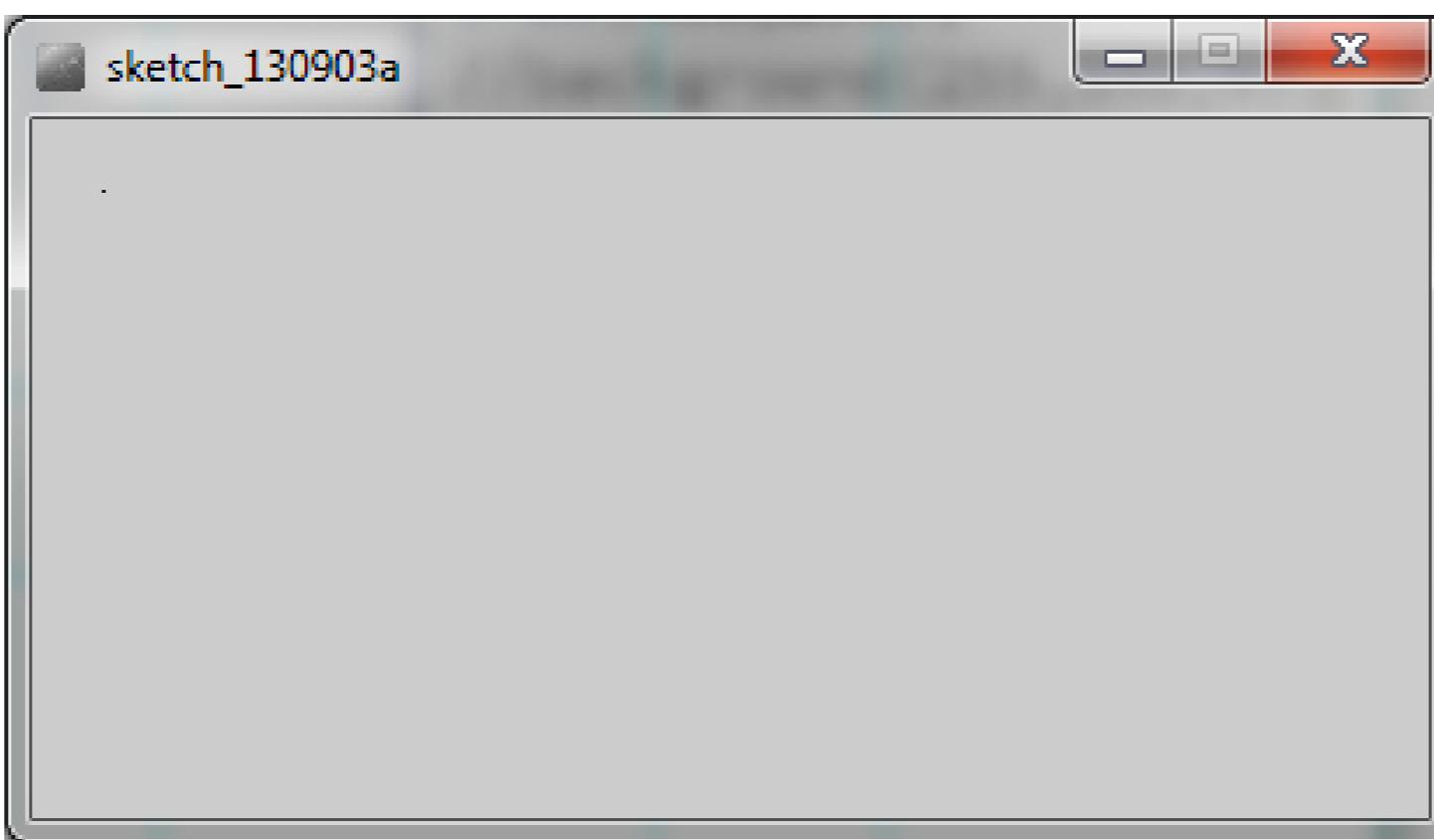
// KOMMENTAR



```
size(400,200);  
//background(255,102,0);
```

// ÜBUNG: KOMMENTAR

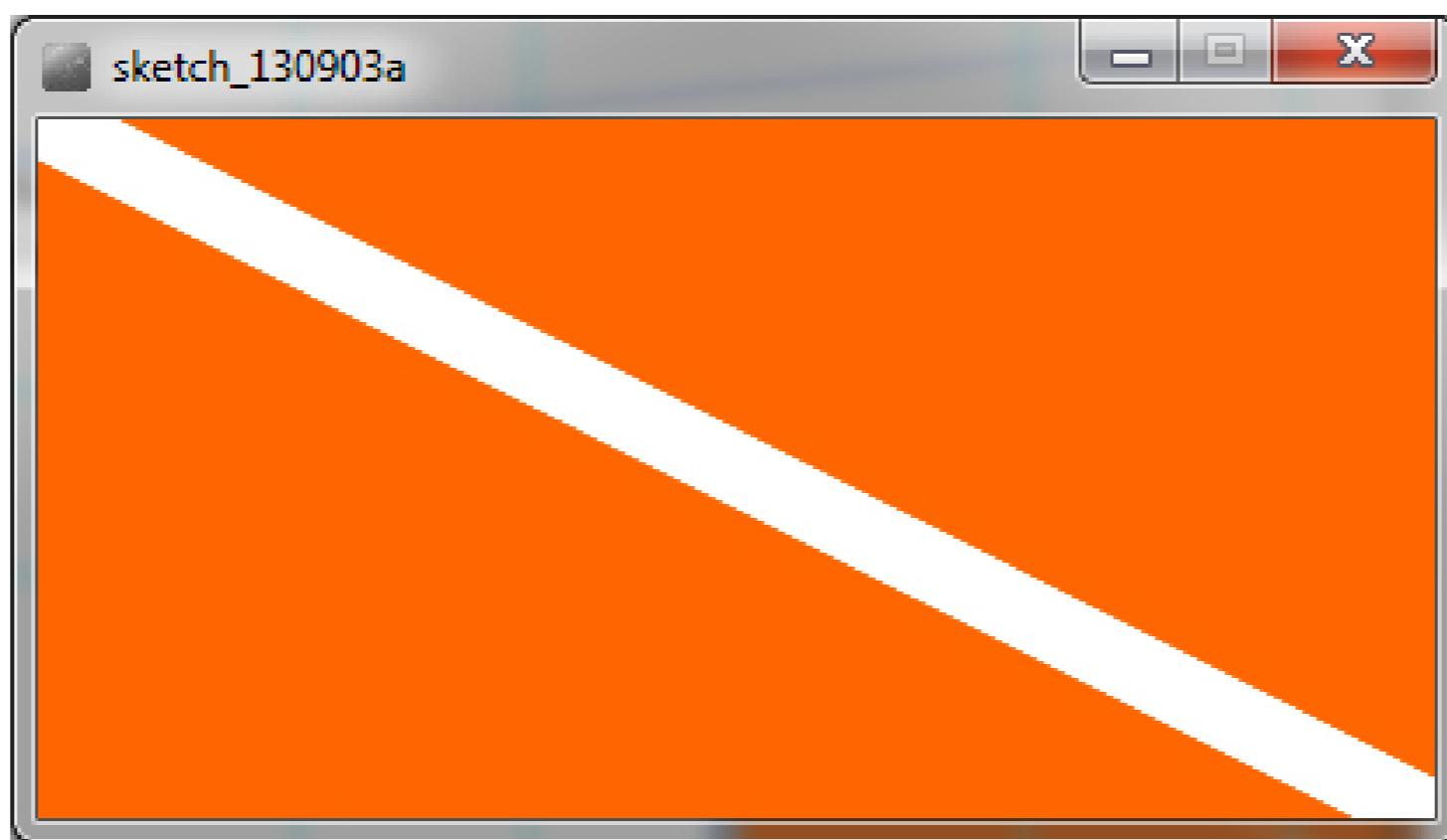
- – Warum schreibt man Kommentare?
- – Kommentiere die Hintergrundfarbe aus!
- – Wann und wie setzt man /* */ ein?



```
size(400,200);
//background(255,102,0);
```

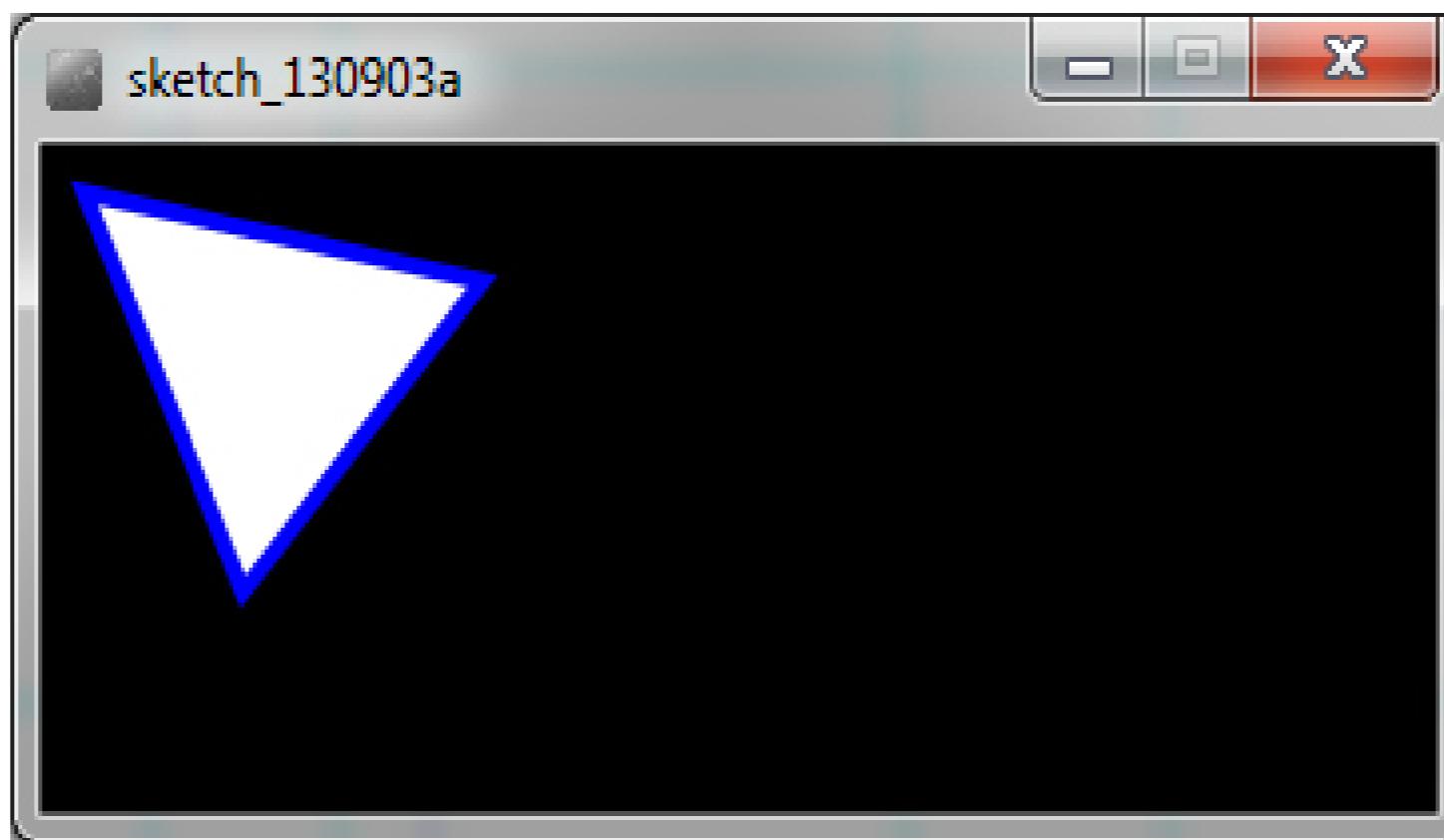
LINIE: LINE();

```
line(2,6,7,1);  
line(punktX1, punktY1, punktX2, punktY2);
```



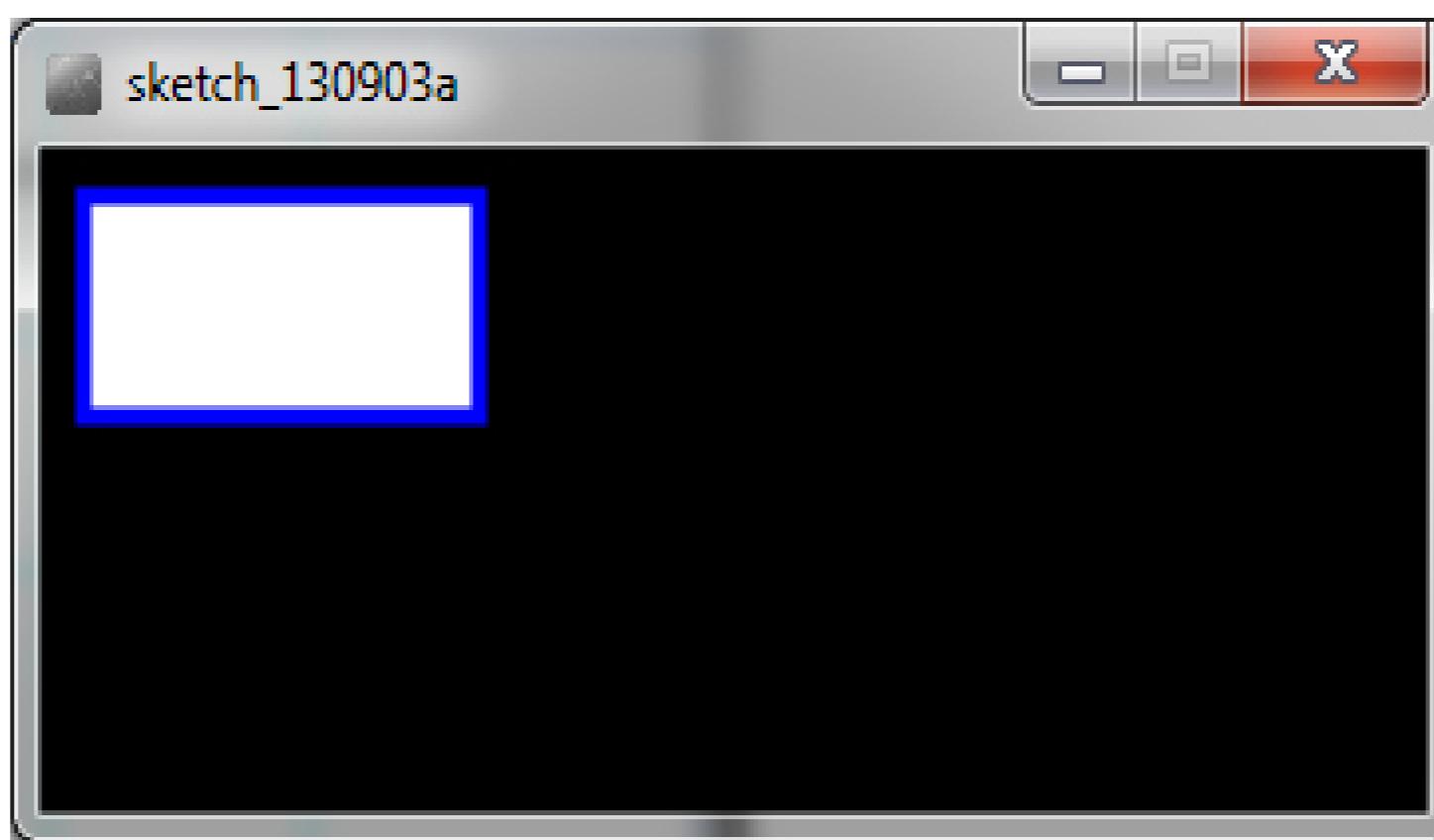
DREIECK: TRIANGLE();

```
triangle(10,10,110,30,50,100);  
triangle(x1,y1,x2,y2,x3,y3);
```



RECHTECK: RECT();

```
rect(10,10,100,50);  
rect(x1,y1,breite,hoehe);
```



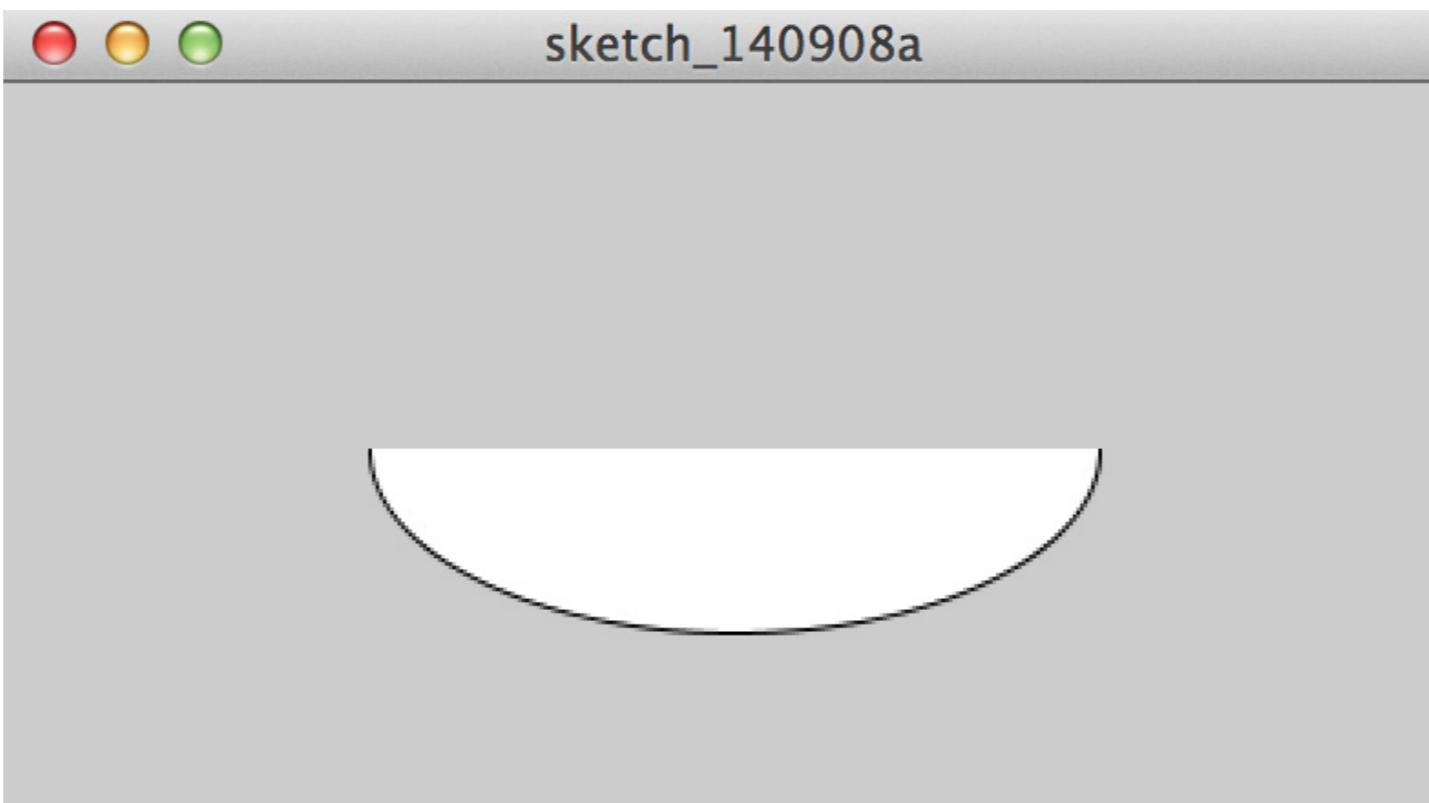
ELLIPSE: ELLIPSE();

```
ellipse(100,60,150,80);  
ellipse(x,y,breite,hoehe);
```



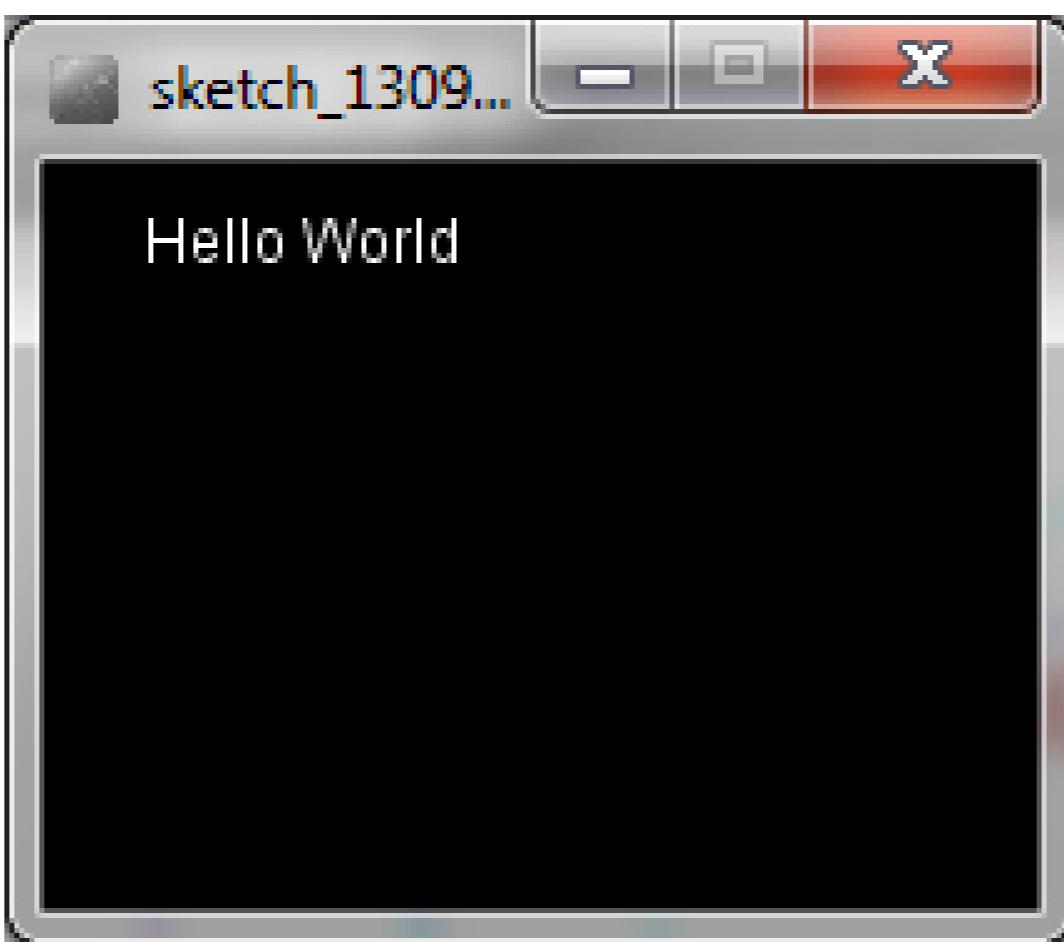
HALBKREIS: ARC();

```
arc(x, y, breite, hoehe, start, stop);
arc(200, 200, 200, 100, 0, PI*2);
```



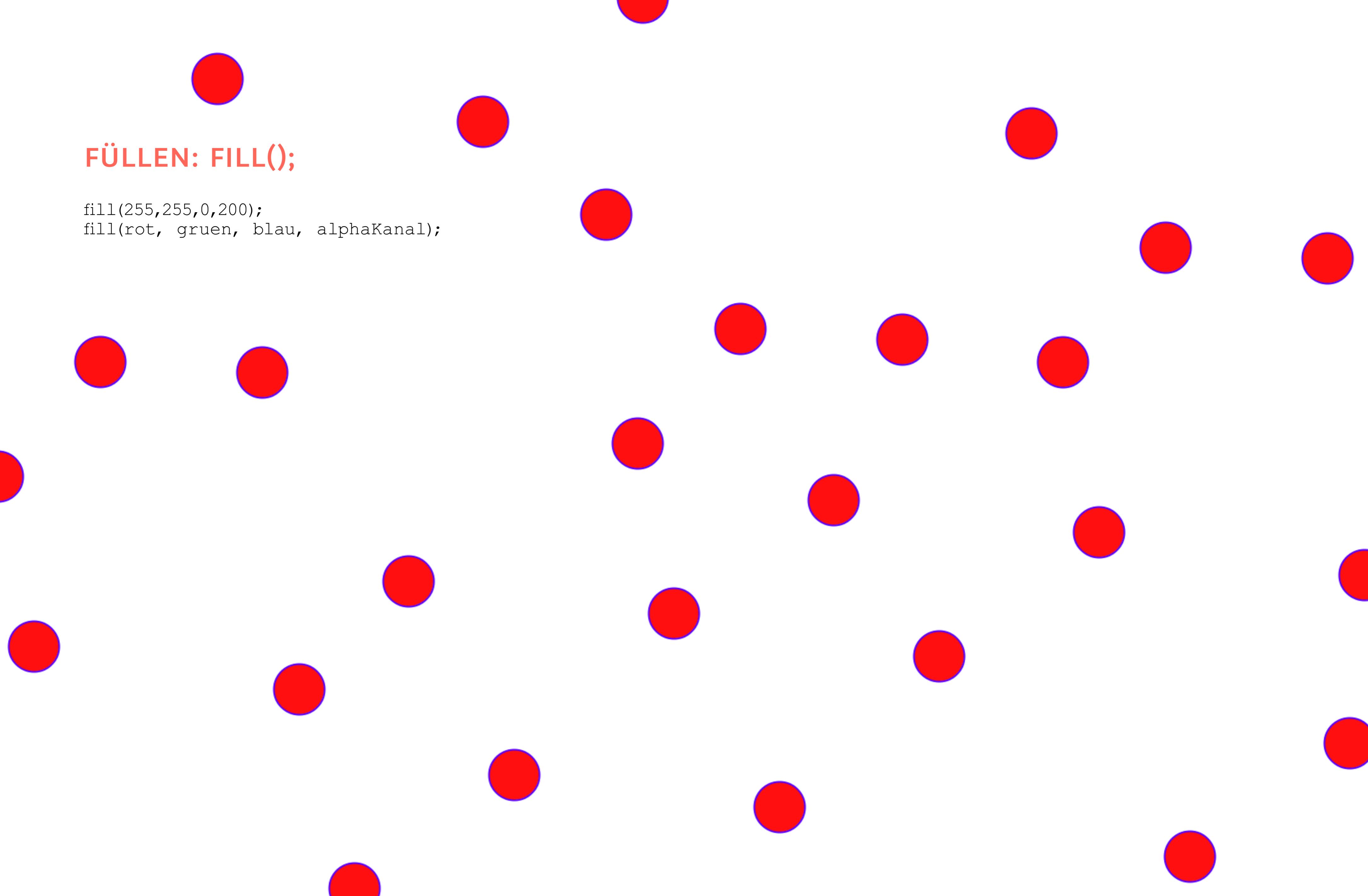
TEXT: TEXT();

```
text("Hello World",20,20);
```



FÜLLEN: FILL();

```
fill(255,255,0,200);  
fill(rot, gruen, blau, alphaKanal);
```



ÜBUNG: MALE DAS HS AUGSBURG LOGO ...

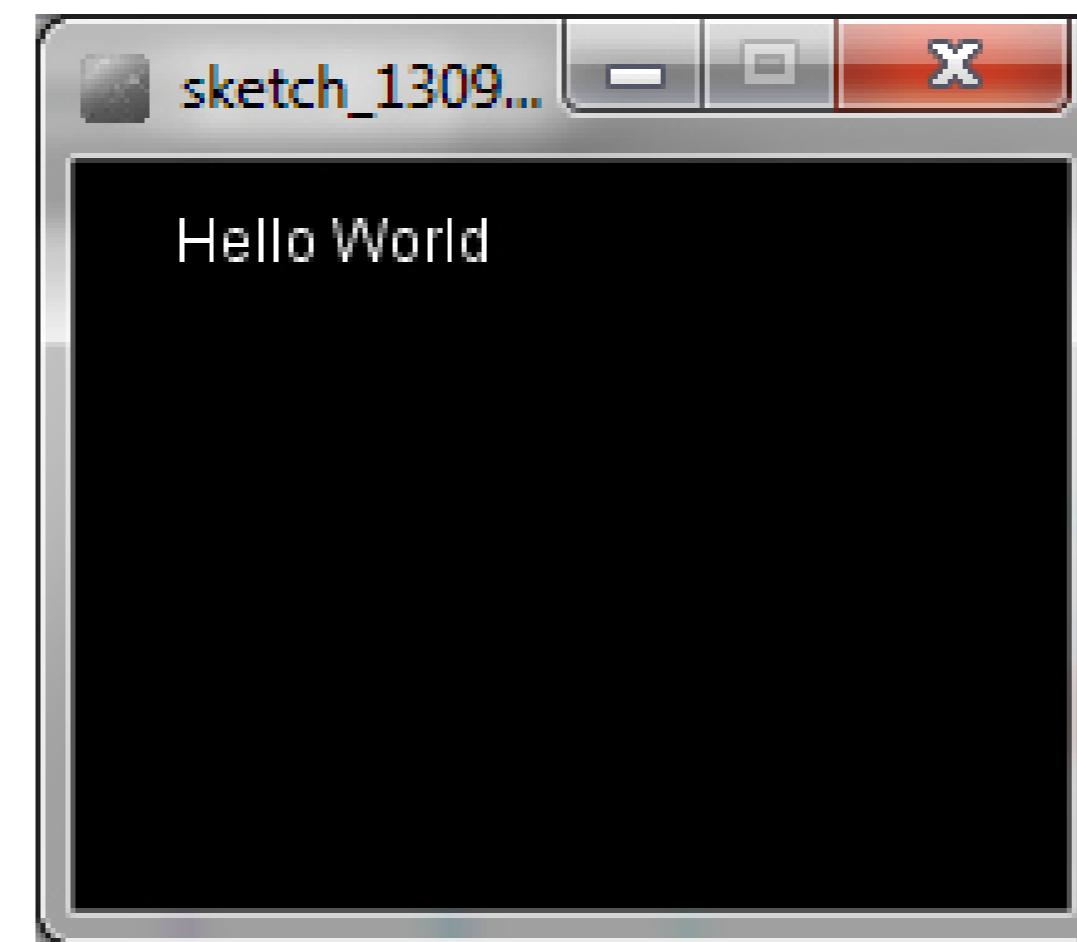
... und setze deinen Namen darunter!



```
point(x,y); stroke(rot, grün, blau); strokeWeight(anzahlPixel); line(punktX1, punktY1, punktX2, punktY2); triangle(x1,y1,x2,y2,x3,y3);
rect(x1,y1,breite,hoehe); ellipse(x,y,breite,hoehe); text("Hello World",20,20); fill(rot, gruen, blau, alphaKanal); background(rot, grün, blau);
arc(x, y, breite, hoehe, start, stop);
```

METHODEN / FUNKTIONEN

$$V(x) = (a + b) \left[\frac{\sum_{i=0}^{x-1} \left(\frac{1-p}{p} \right)^i}{\sum_{i=0}^{a+b-1} \left(\frac{1-p}{p} \right)^i} \right]$$

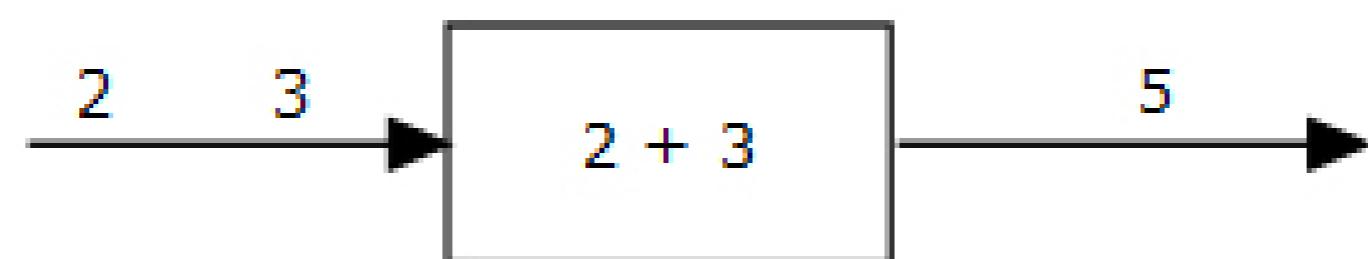
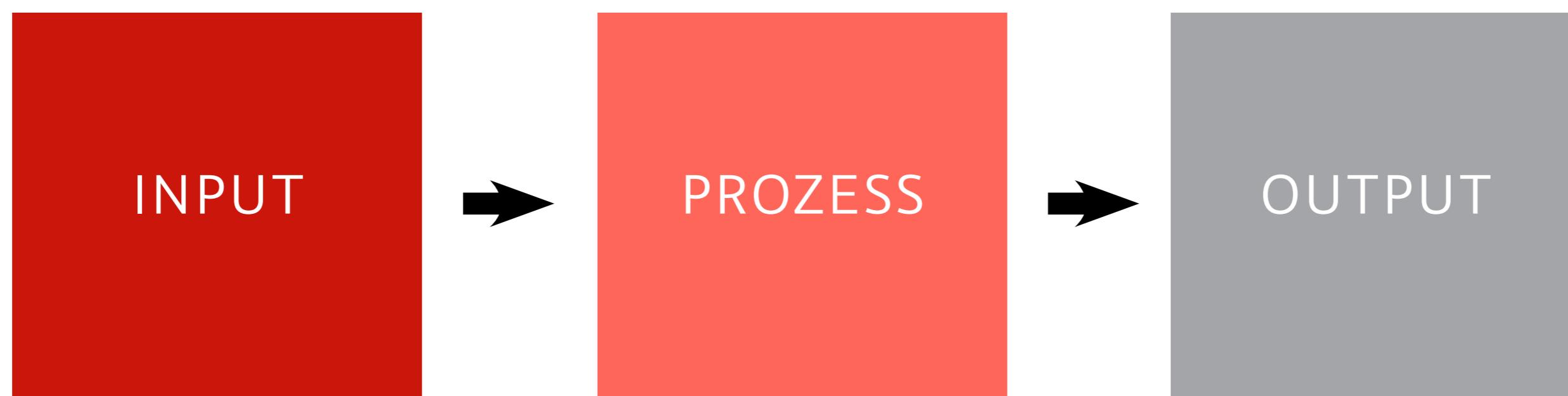


```
text("Hello World",20,20);
```

METHODEN / FUNKTIONEN



METHODEN / FUNKTIONEN



METHODEN / FUNKTIONEN



```
int funktionName(int input1, int input2) {  
    // mein code  
    return input1 * input2;  
}  
  
void setup() {  
    println(funktionName(2,3));  
}
```

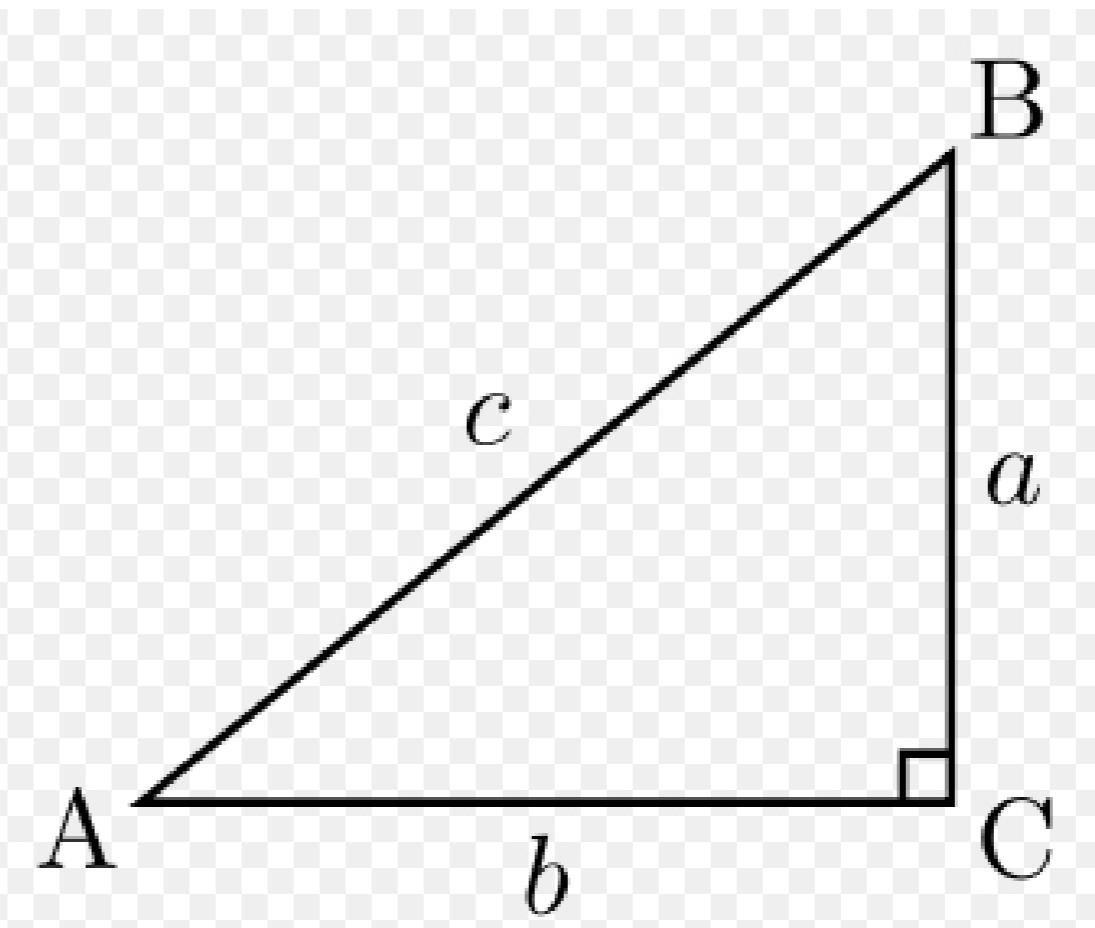
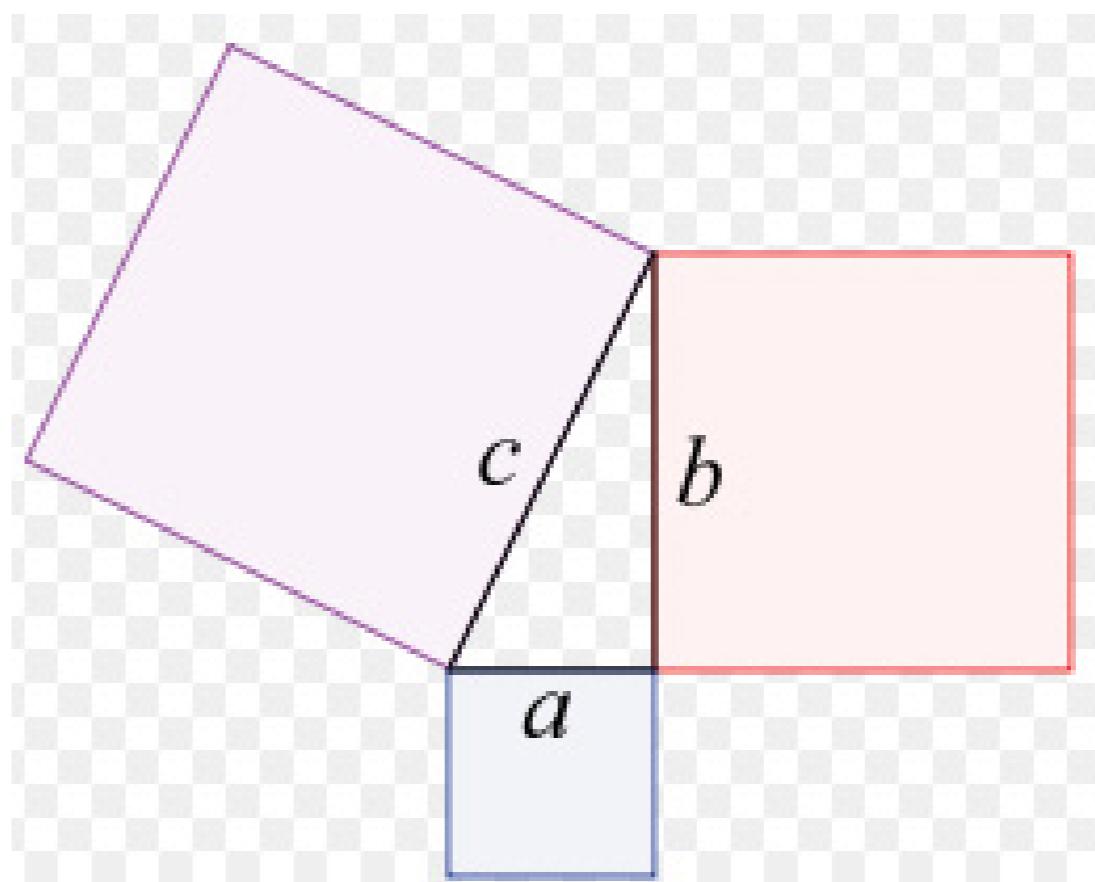
SETUP() & DRAW()

= dt. leer

```
void setup() {  
    /* Dieser Block wird nur einmal zu Beginn der  
    Anwendung aufgerufen */  
}  
  
void draw() {}  
// Dieser Block wird kontinuierlich ausgeführt  
}
```

RETURN

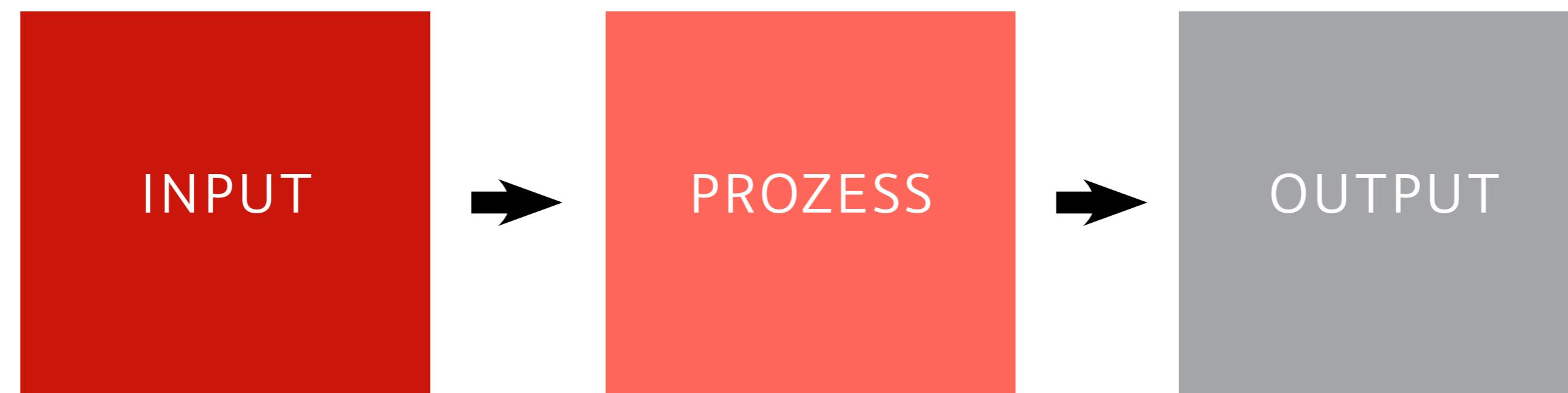
```
void setup() {  
    background(0);  
    size(200, 150);  
    text("22 + 42 = " + satzDesPythagoras(2, 4), 10, 20);  
}  
  
float satzDesPythagoras(int a, int b) {  
    sqrt(a*a+b*b); // ohne return  
    return sqrt(a*a+b*b); // mit return  
}
```



ÜBUNG: METHODEN / FUNKTIONEN

```
int funktionName(int input1, int input2) {  
    // mein code  
    return input1 * input2;  
}  
  
void setup() {  
    println(funktionName(2,3));  
}
```

- Eine Funktion die zwei Zahlen addiert und das Ergebnis per Return zurück gibt.
- Rufe die Addition-Funktion mit 2 und 3 auf.
- Eine Funktion die zwei Zahlen subtrahiert und das Ergebnis auf dem Bildschirm ausgibt.
- Rufe die Subtraktions-Funktion mit 6 und 4 auf.



CONSOLE OUTPUT: PRINTLN(); & PRINT();

The image shows a screenshot of the Processing 2.2.1 software interface. The title bar reads "sketch_140908a | Processing 2.2.1". The code editor contains the following Java code:

```
console.log("Hello World");
println("Hello world!");
print("Bist Du ");
println("Startklar?");
println("Bist Du ");
print("Startklar?");
```

The processing window is black and displays the following text in white:

```
Hello world!
Bist Du Startklar?
Bist Du
Startklar?
```

On the right side of the image, there are three lines of text corresponding to the different print methods:

- `alert("hello world");`
- `echo hello World`
- `7`

VARIABLEN

SINN UND
ZWECK?

7

10



8

11



9



12

DATENTYPEN

7



8

9

Datentyp	Länge [Byte]	Wertebereich	Standardwert
int	2	-2147483648 bis 2147483647	0
float	4	+/- 3,40282347*10 ³⁸	0.0
boolean	1	true, false	false
char	2	Alle Unicodezeichen	
String	x*char	Zeichenkette	

ÜBUNG: DATEN UND DATENTYPEN

Welche Daten haben welchen Typ?

- Alter? "22"
- PI? "3.14159265359"
- An/Aus?
- Anzahl der Studenten?
- "a", "b", "c"
- "abc"
- Durchschnittliche Anzahl von Kindern pro Frau?
- Ergebnis von 1:2
- Geburtsdatum?
- Donnerstag, 1. Januar 1970 00:00

Datentyp	Wertebereich
int	-2147483648 bis 2147483647
float	+/- 3,40282347*10 ³⁸
boolean	true, false
char	Alle Unicodezeichen
String	Zeichenkette

VARIABLENNAMEN

WO FINDE ICH

MEINE DATEN WIEDER?

7



8

- sprechend, z.B. "anzahlStudenten" und nicht "n"
- erster Buchstabe eines Variablenamens wird klein geschrieben. Ansonsten CamelCase
- Ungültige Variablennamen: null, true, boolean

10

11



9

12

SYNTAX

datentyp variablenName;
variablenName = zugewiesenerWert;

7

8

10

11

9

12

SYNTAX

datentyp variablenName;
variablenName = zugewiesenerWert;

oder

datentyp variablenName = zugewiesenerWert;

10

7

8

11



9

12

SYNTAX

datentyp variablenName;
variablenName = zugewiesenerWert;

oder

datentyp variablenName = zugewiesenerWert;

```
int anzahlStudenten = 32;
float pi = 3.14;
boolean istStudent = true;
String vorlesungName = "Startklar 2014";
char euroZeichen = "€";
```

10

11

12

SICHTBARKEIT VON VARIABLEN

```
// Globale Variablen
int posX = 20;
int posY = 20;

void setup() {
    // Lokale Variablen
    int breiteFenster = 500;
    int hoeheFenster = 350;
    size(breiteFenster, hoeheFenster);
    frameRate(30);
}

void draw() {
    background(255, 102, 0);
    ellipse(posX,posY,40,40);
    println("PositionX: " + posX + " PositionY: " + posY);
    posX = posX + 1;
    posY = posY + 1;
}
```

ÜBUNG: SICHTBARKEIT VON VARIABLEN

```
// Globale Variablen
int posX = 20;
int posY = 20;

void setup() {
    // Lokale Variablen
    int breiteFenster = 500;
    int hoeheFenster = 350;
    size(breiteFenster, hoeheFenster);
    frameRate(30);
}

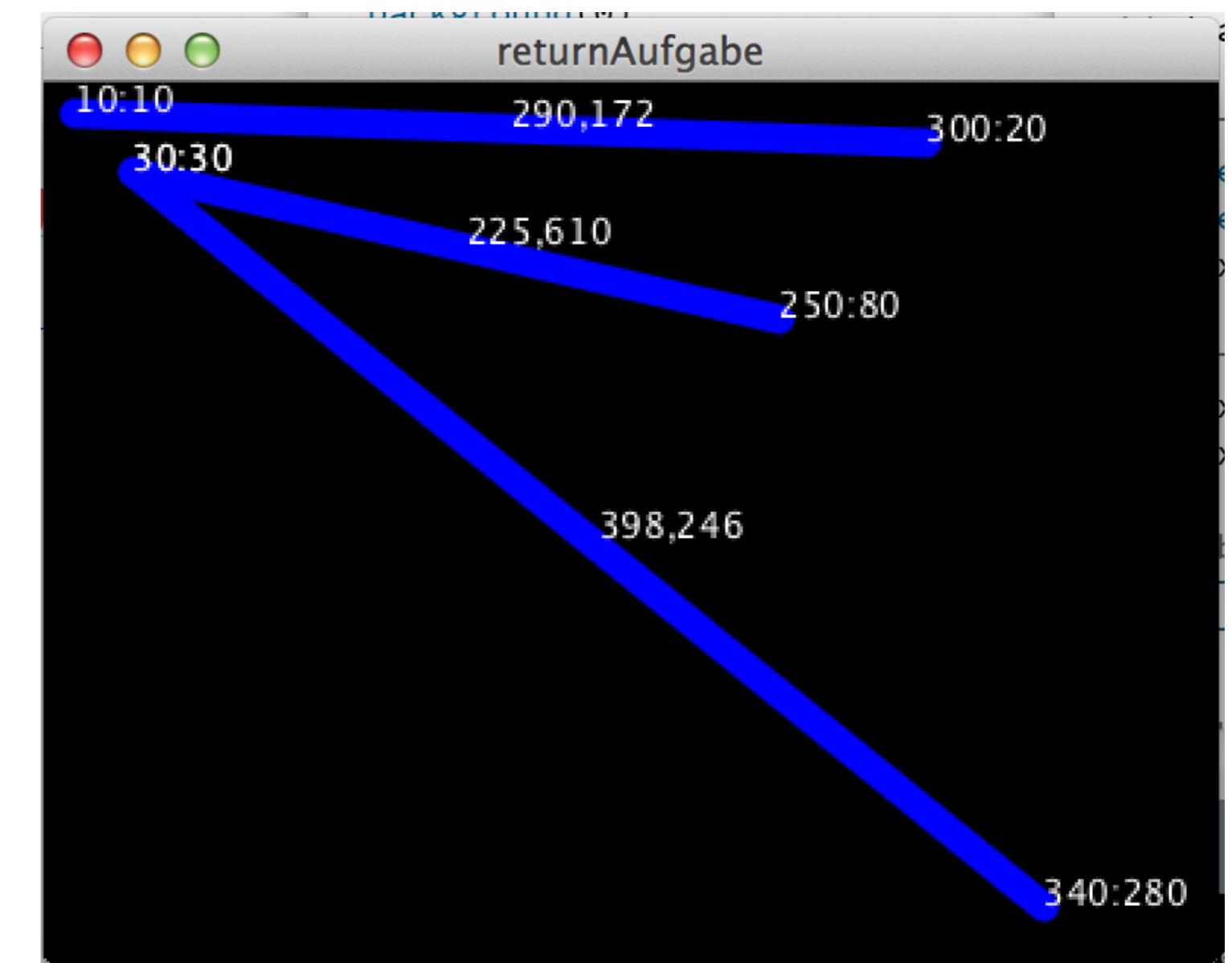
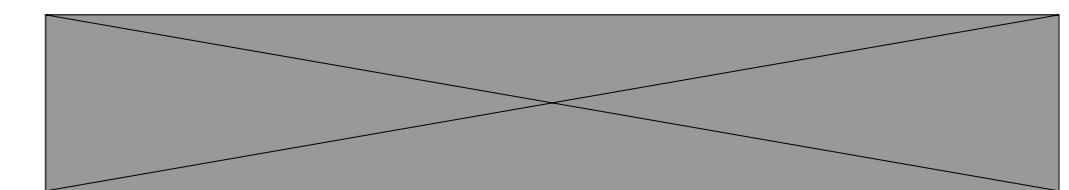
void draw() {
    background(255, 102, 0);
    ellipse(posX, posY, 40, 40);
    println("PositionX: " + posX + " PositionY: " + posY);
    posX = posX + 1;
    posY = posY + 1;
}
```

- Was passiert, wenn Du "posX = posX +1;" durch "posX++;" ersetzt?
- Was passiert, wenn Du "posY = posY +1;" durch "posY--;" ersetzt?
- Was passiert, wenn Du den Parameter von frameRate auf 1 setzt?
- Was passiert, wenn Du int von posX in float änderst?
- Was passiert, wenn Du int von posY in boolean änderst?
- Was passiert, wenn Du die globalen Variablen (die ersten drei Zeilen) löscht?
- Was passiert, wenn Du die globalen Variablen in die setup-Methode verschiebst?
- Was passiert, wenn Du die globalen Variablen unter die draw-Funktion schreibst?
- Was passiert, wenn Du die globalen Variablen am Anfang in die draw-Funktion verschiebst?

ÜBUNG: METHODEN

- Schreibe eine Funktion, die die Distanz/Länge von x_1, y_1, x_2 und y_2 berechnet und den Wert per Return zurückgibt.
- Schreibe eine Funktion, die den Mittelpunkt zwischen Punkt1 und Punkt2 zurückgibt.
- Schreibe eine Funktion, die anhand der Koordinaten x_1, y_1, x_2 und y_2 ...
 - eine blaue 10px breite Linie zeichnet
 - die Koordinaten im Fenster ausgibt
 - die Koordinaten in der Console ausgibt
 - die Länge im Fenster ausgibt
 - und die Länge in der Console ausgibt
- Rufe diese Funktion mit folgenden Werten auf:
 - $x_1: 10, y_1: 10, x_2: 300, y_2: 20$
 - $x_1: 30, y_1: 30, x_2: 250, y_2: 80$
 - $x_1: 30, y_1: 30, x_2: 340, y_2: 280$
- Für Schnelle: gib jeder Linie eine andere Farbe.

SQRT

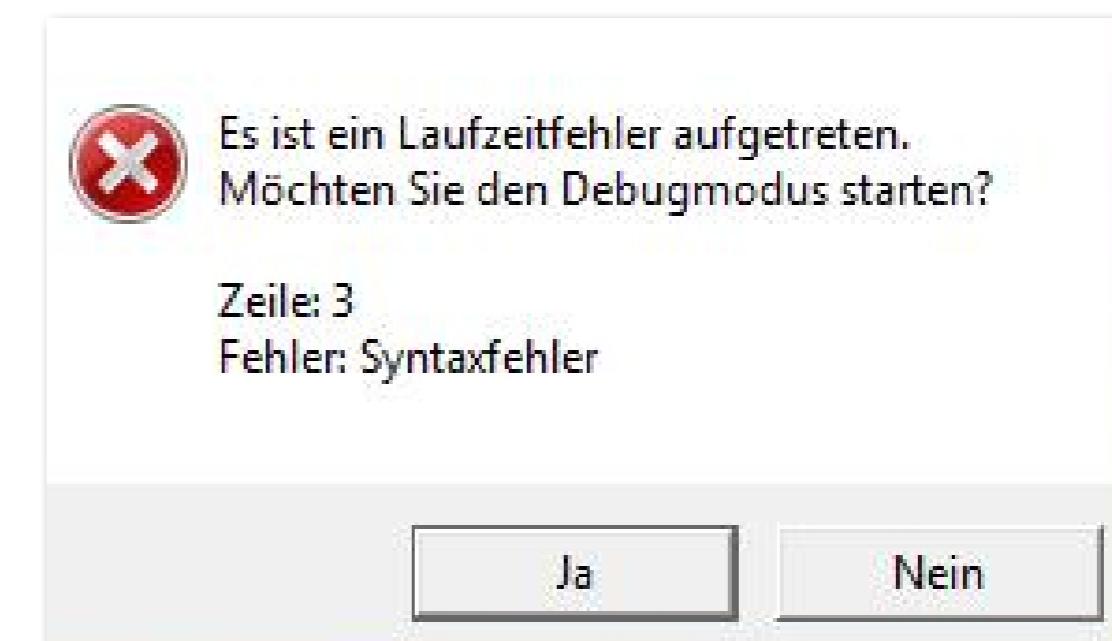


FEHLER

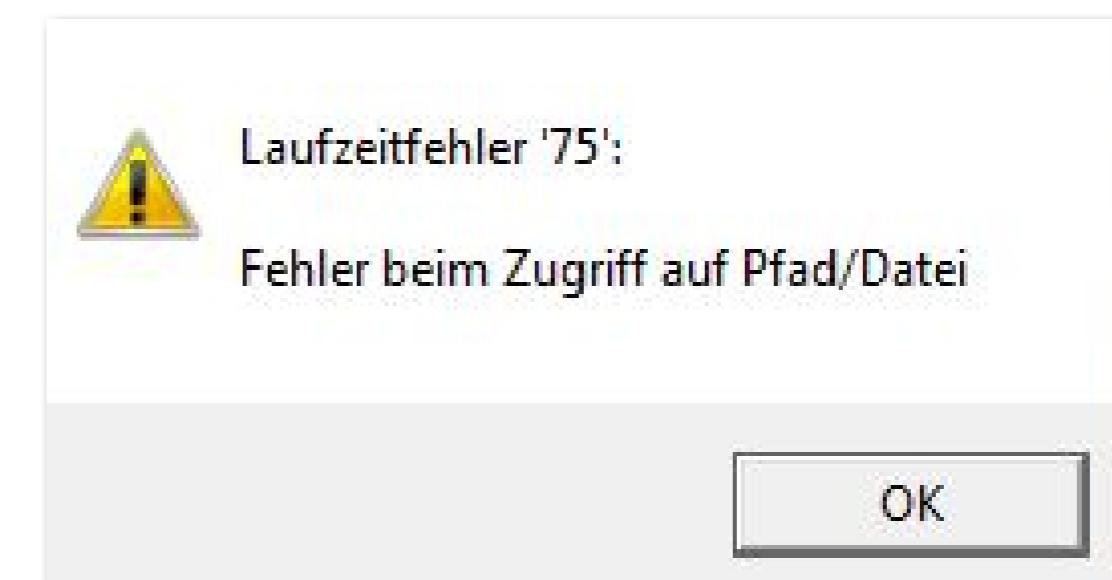
LOGISCHE FEHLER



SYNTAXFEHLER

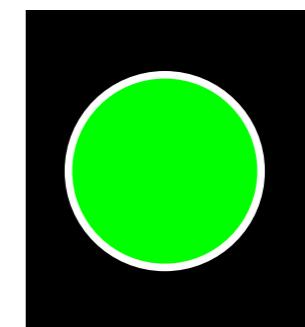


LAUFZEITFEHLER

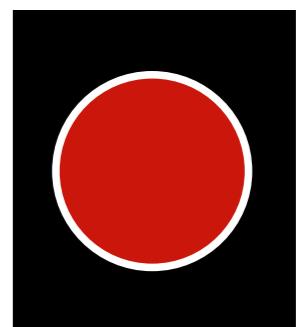


IF-BEDINGUNG

$$s(x) = \begin{cases} 1 & \text{für } x > 0, \\ 0 & \text{für } x = 0, \\ -1 & \text{für } x < 0 \end{cases}$$



FREI



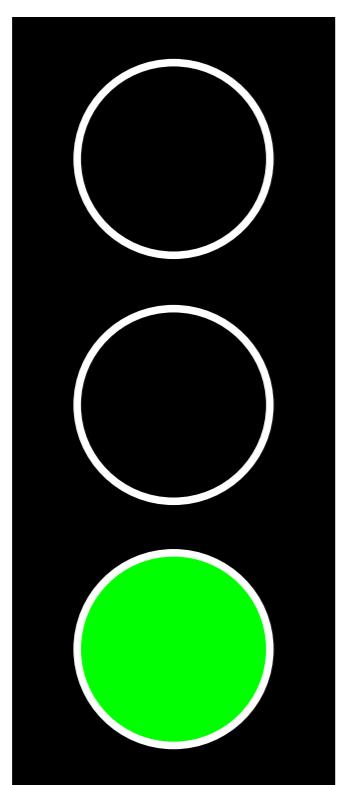
BESETZT



GTÜ / pixelio.de



PixelWookie / pixelio.de



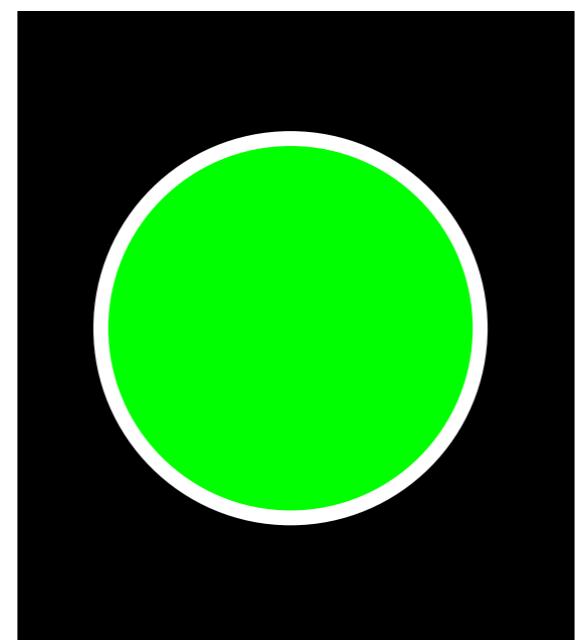
FotoHiero / pixelio.de



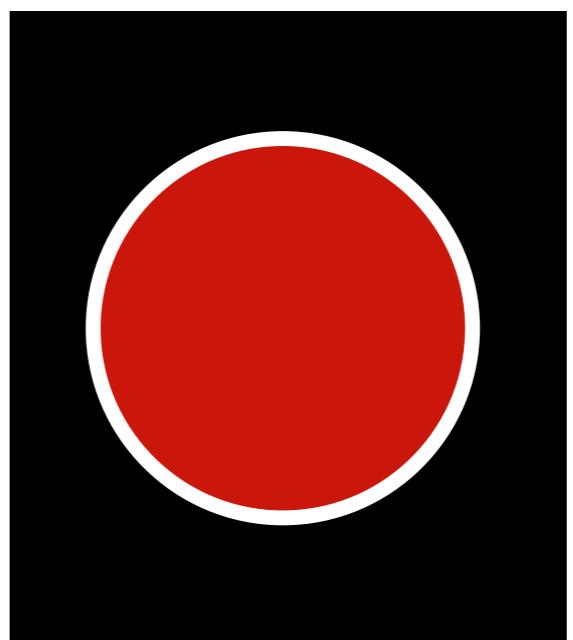
Quelle_Wikipedia

IF-BEDINGUNG

```
String toilette = "frei";  
  
if(toilette == "frei")  
{  
    println("Geh rein!");  
}  
else  
{  
    println("Warte!");  
}
```



FREI



BESETZT

IF-BEDINGUNG

```
boolean schildAufgestellt = true;  
  
if(schildAufgestellt == true)  
{  
    println("Durchgang verboten!");  
}  
else  
{  
    println("Gehe durch!");  
}
```



VERGLEICHSOPERATOREN

`==` Gleichheitsoperator

`!=` Ungleichheitsoperator

`<` Kleiner-als-Operator

`>` Größer-als-Operator

`<=` Kleiner-als-oder-gleich-Operator

`>=` Größer-als-oder-gleich-Operator

ÜBUNG: TRUE OR FALSE?

1 == 1 1 == a
1 == 2 'a' == 'a'
1 != 1 true != false
1 != 2
2 > 1
1 <= 1
1 >= 2

```
if(1 == 1)
{
    print("ja");
}
else
{
    print("nein");
}
```

== Gleichheitsoperator
!= Ungleichheitsoperator
< Kleiner-als-Operator
> Größer-als-Operator
<= Kleiner-als-oder-gleich-Operator
>= Größer-als-oder-gleich-Operator

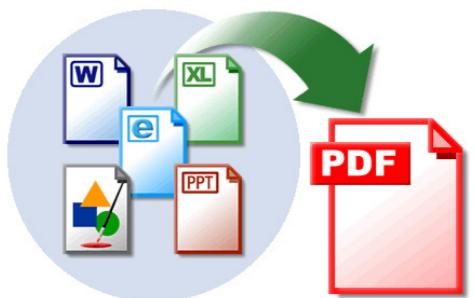
CLEAN CODE

- CamelCase
 - Variablennamen beginnen mit einem Kleinbuchstaben
 - Methodennamen beginnen mit einem Kleinbuchstaben
 - Wir verwenden aussagekräftige Namen: “anzahlStudenten” statt “n”
 - Wir rücken den Quelltext ein
 - Wir setzen ein Leerzeichen vor und nach jedem Operator und einfaches Istgleich-Zeichen
 - Kommentare
 - ...
-
- <http://code.jquery.com/jquery-1.10.2.min.js>
 - <http://code.jquery.com/jquery-1.10.2.js>

OPEN SOURCE



OPEN SOURCE



mozilla
Firefox®



You forgot a
semicolon.

PAIR PROGRAMMING

ÜBUNG: PAIR-PROGRAMMING

- // globale Variablen

```
int verbliebeneZeit; int gesamtZeit;
```

- // arc-endWert

```
float endWert = 2 * PI / gesamtZeit * (gesamtZeit - verbliebeneZeit);
```

- // Inputfenster:

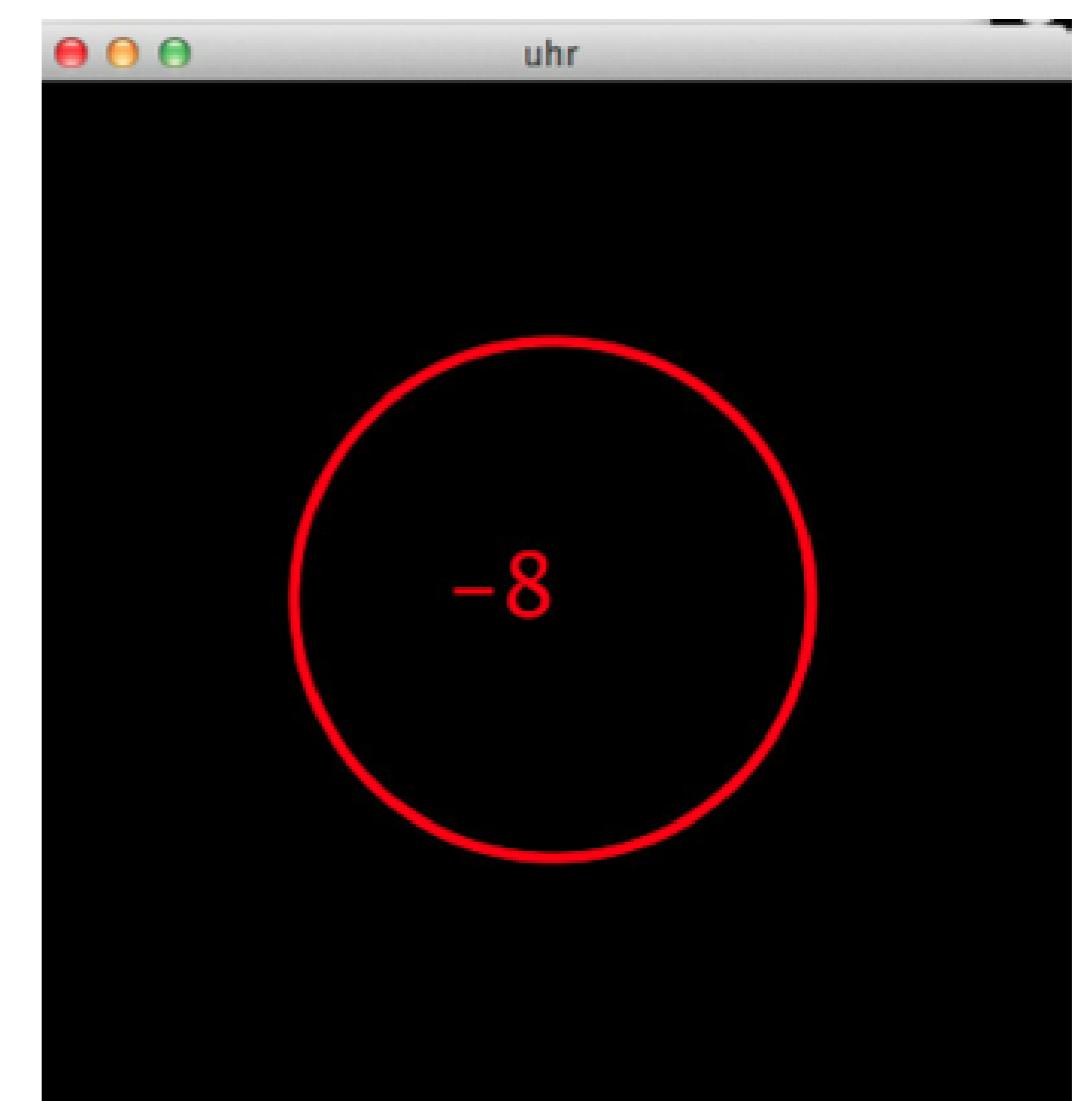
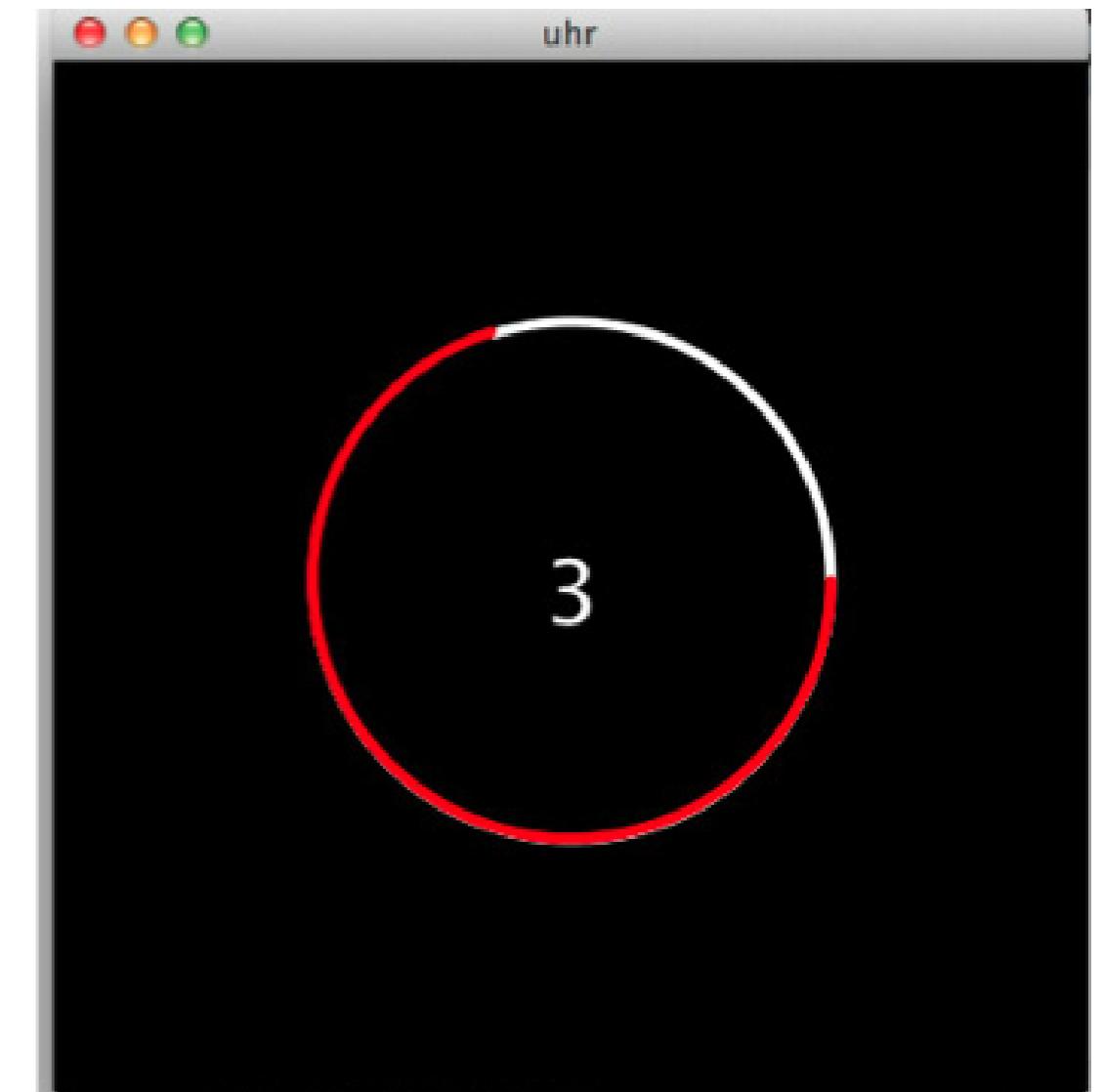
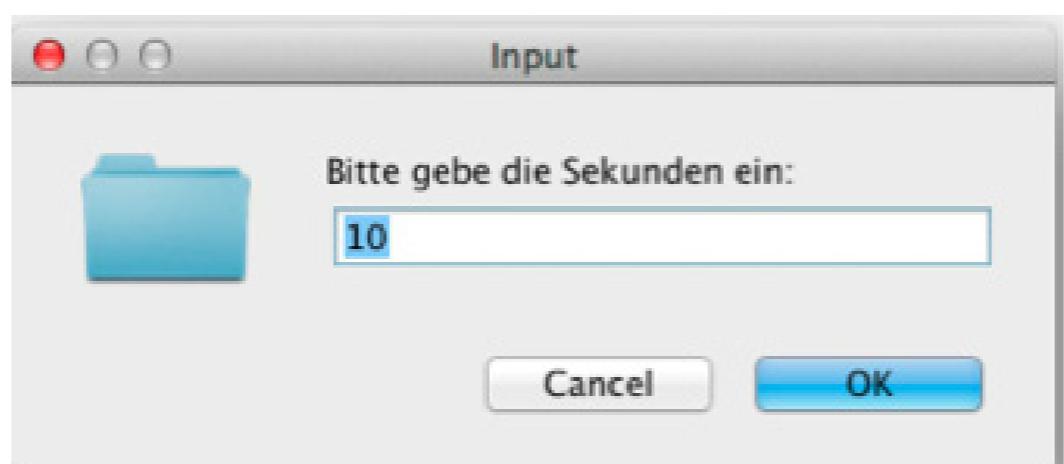
```
gesamtZeit = Integer.parseInt(javax.swing.JOptionPane.showInputDialog("Bitte gebe die Sekunden ein:", 10));
```

- // Center

```
textAlign(CENTER, CENTER);
```

- // ungerade?

```
if(verbliebeneZeit % 2 == 0) { }
```



boolean sindDieStudentenStartklar(Studenten Startklar2014)

{

- Sandwich;
- Telefonbuch;
- processing;
- size();
- Semikolon;
- IDE;
- background();
- RGB;
- point();
- stroke();
- CamelCase;
- Kommentare;
- line();
- triangle();
- rect();
- ellipse();
- text();
- fill();
- Logo malen
- Roboter spielen;
- setup();
- void;
- draw();
- Methoden;
- println();
- i++;
- satzDesPythagoras;
- Variablen;
- Datentypen;
- int;
- float;
- boolean;
- char;
- String;
- Variablennamen;
- Sichtbarkeit;
- Logische Fehler;
- Syntaxfehler;
- Laufzeitfehler;
- Tastatur-Input;
- Aufruf Methoden;
- if;
- ==;
- !=;
- <;
- >;
- <=;
- >=;
- Sauberer Code;
- Open Source;
- Pair Programming;

return true++;

}

EINFÜHRUNG

SOFTWAREENTWICKLUNG

Hochschule Augsburg

Fakultät Informatik

Wintersemester 2014/2015

STEPHAN BATTEIGER

+49 (0) 8131 779979-1

stephan.batteiger@peerigon.com

peerigon.com

THE END

THANK YOU & GOODBYE