INTERACTIVE ART

SERIELLE SCHNITTESTELLE ZWISCHEN ARDUINO UND PROCESSING

PROCESSING

programmierenvironment für interaktive animationen

www.processing.org

Warum selbst programmieren?

kreatives Ausdrucksmittel

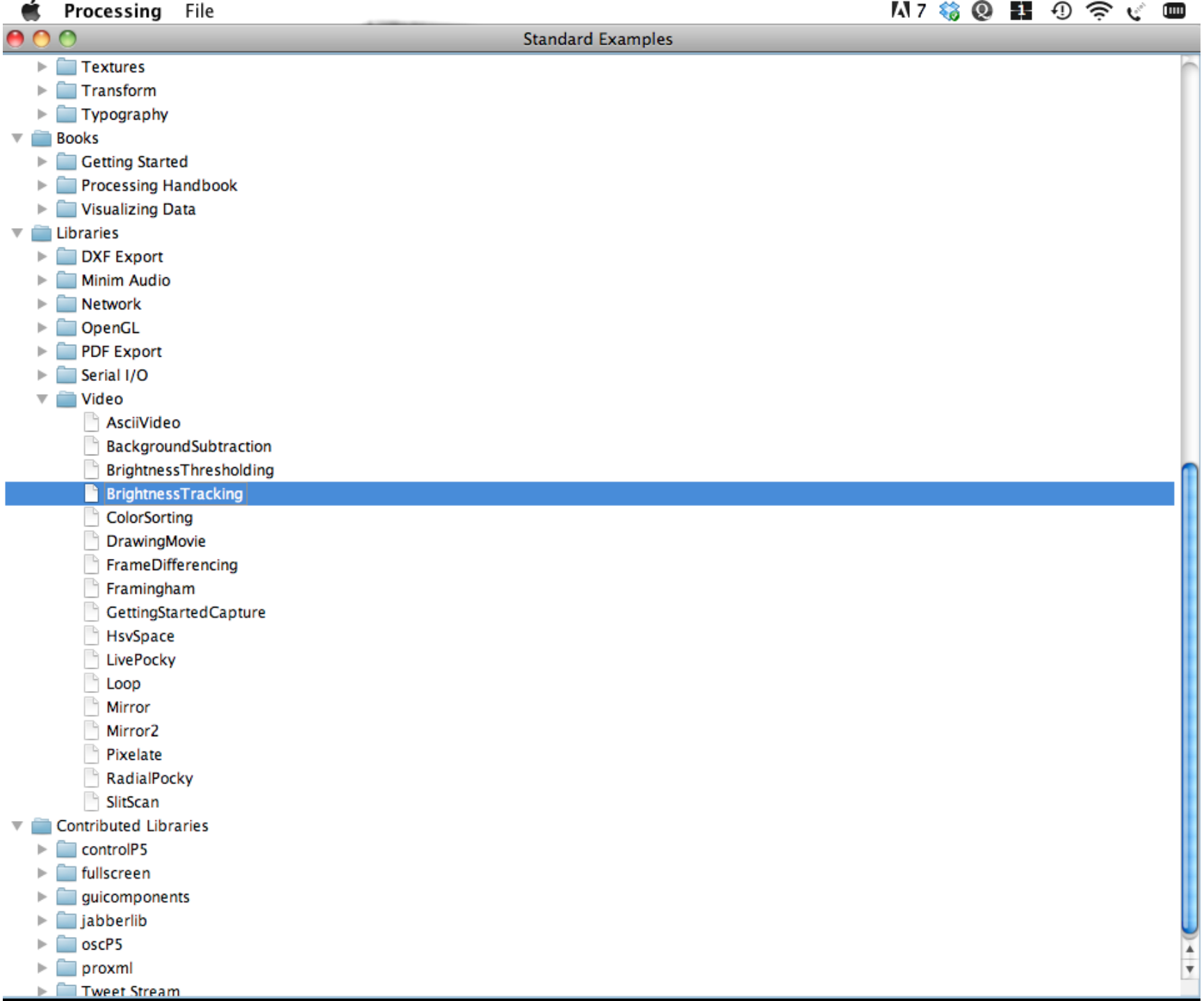
kostenlos

viel Hilfe online

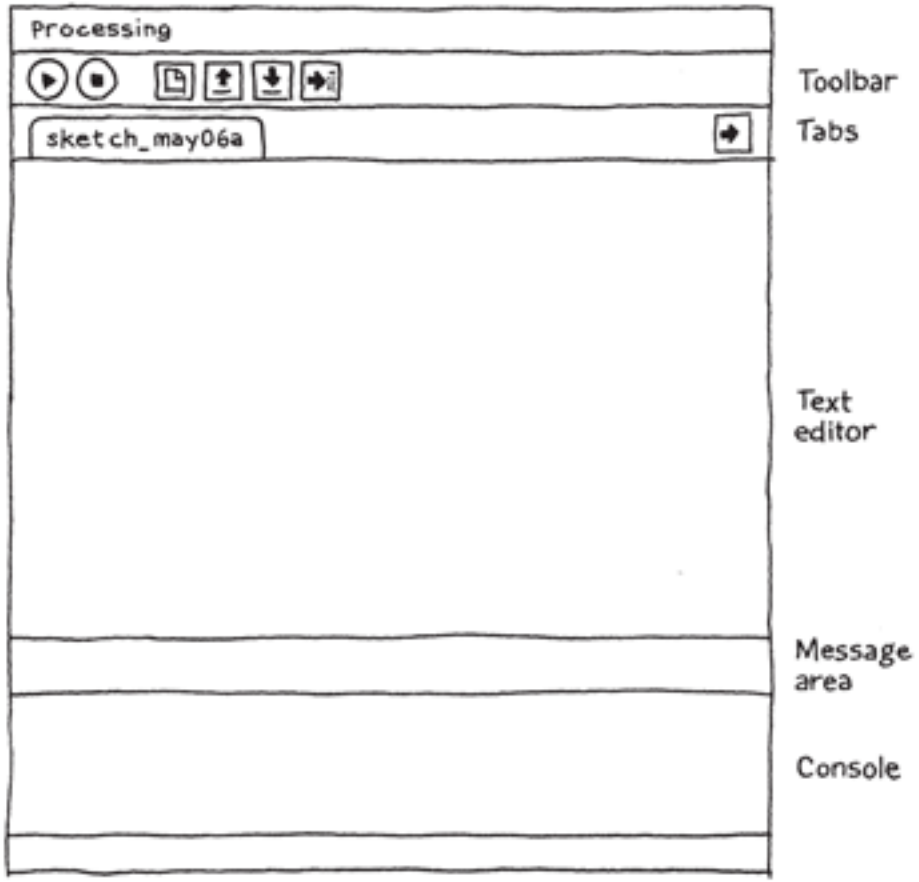
mehr Verständnis

Bitte öffne jetzt das Programm „Processing“

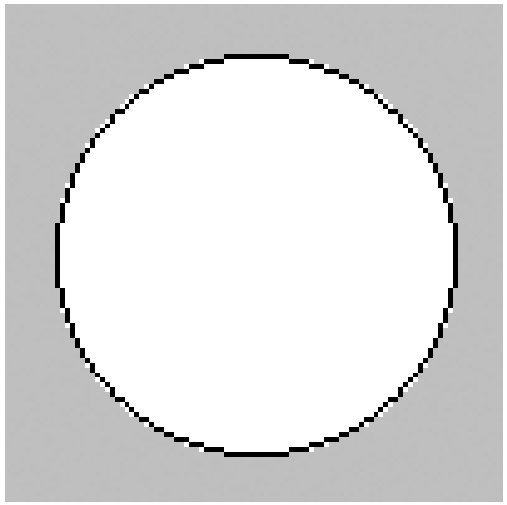
www.processing.org


Processing File
Standard Examples
Textures
Transform
Typography
Books
Getting Started
Processing Handbook
Visualizing Data
Libraries
DXF Export
Minim Audio
Network
OpenGL
PDF Export
Serial I/O
Video
AsciiVideo
BackgroundSubtraction
BrightnessThresholding
BrightnessTracking
ColorSorting
DrawingMovie
FrameDifferencing
Framingham
GettingStartedCapture
HsvSpace
LivePocky
Loop
Mirror
Mirror2
Pixelate
RadialPocky
SlitScan
Contributed Libraries
controlP5
fullscreen
guicomponents
jabberlib
oscP5
proxml
Tweet Stream

Display window

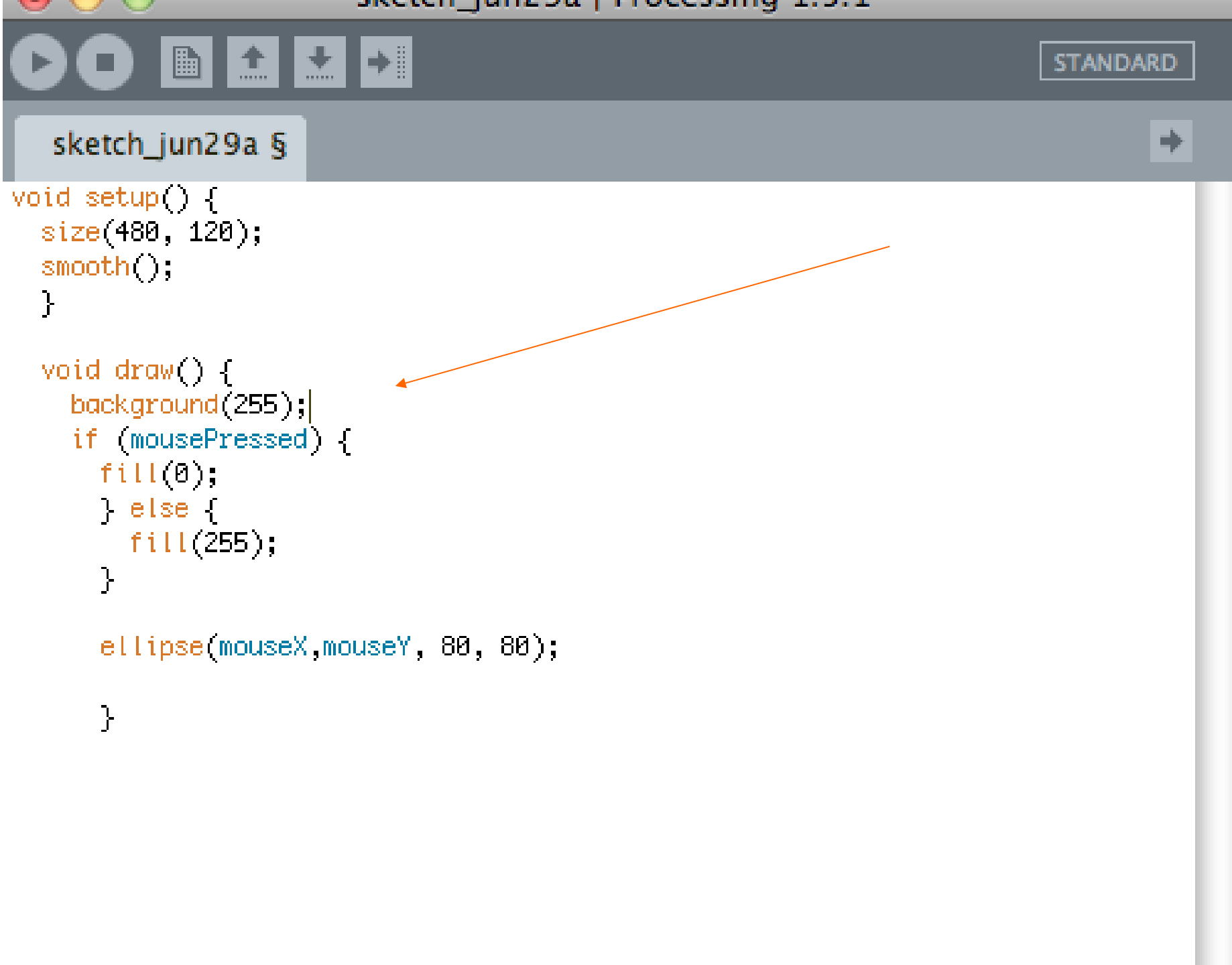


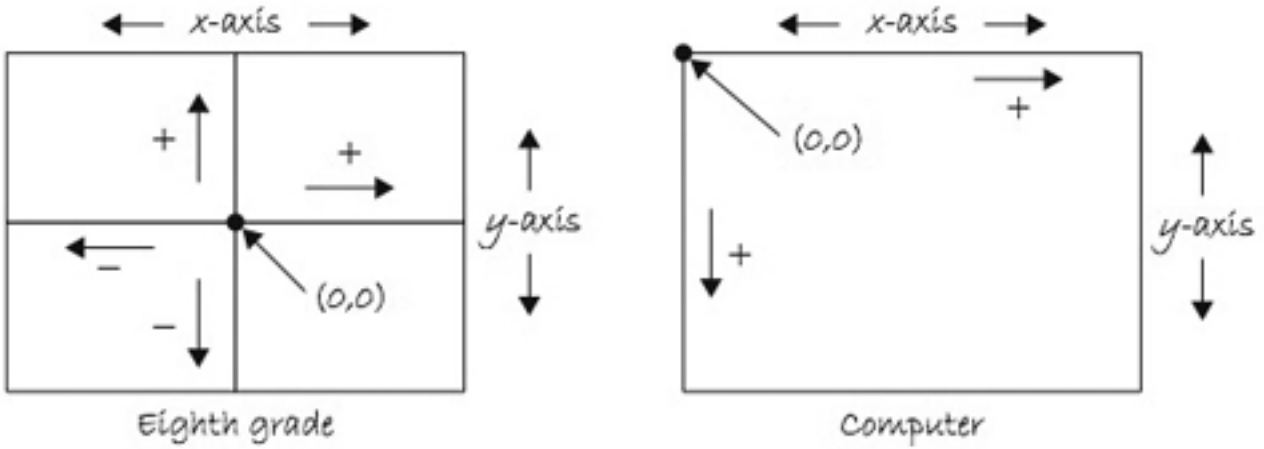
LIBRARIES

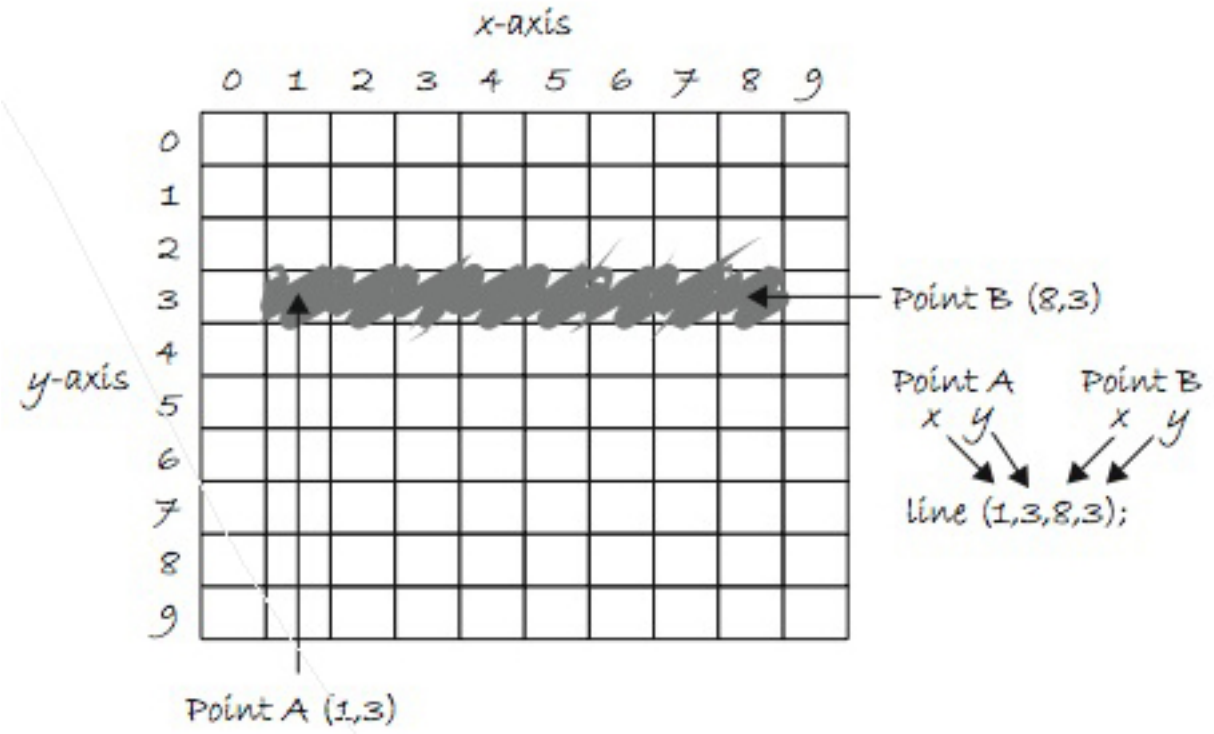


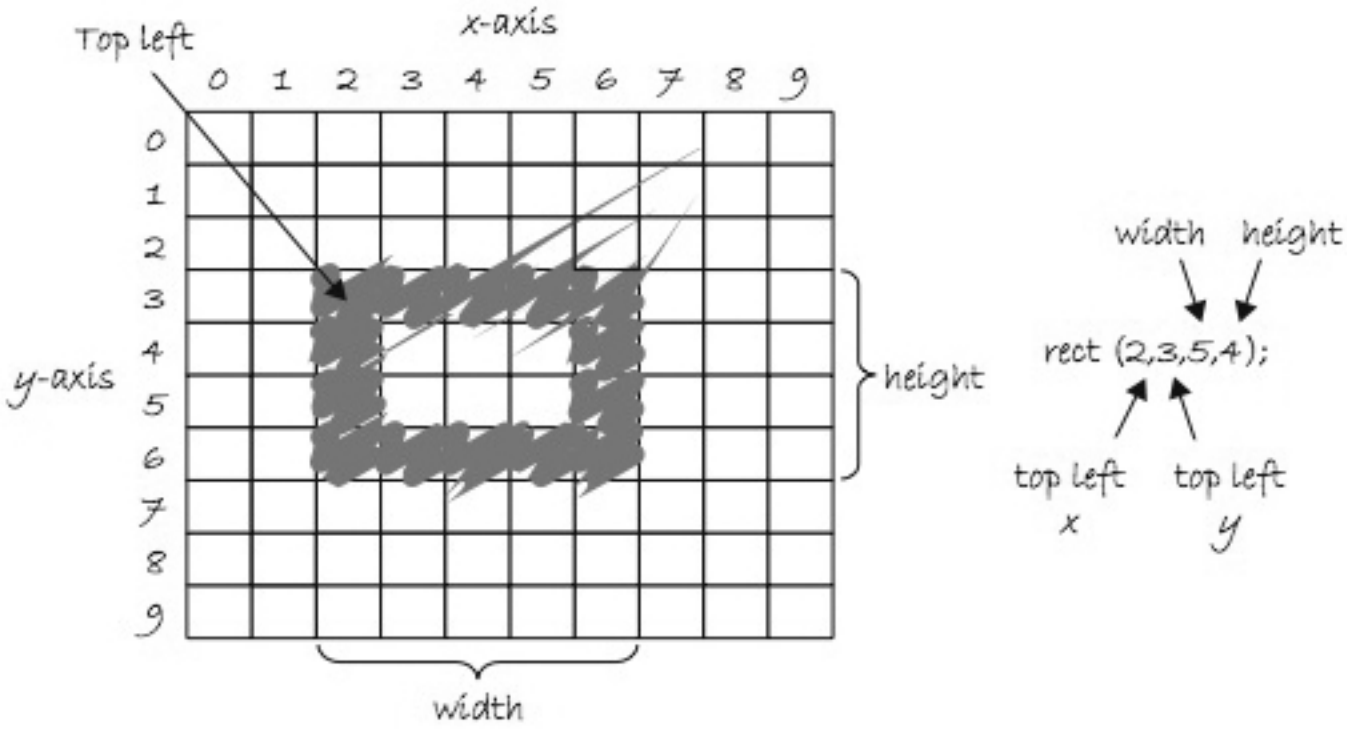
```
ellipse(50, 50, 80, 80);
```

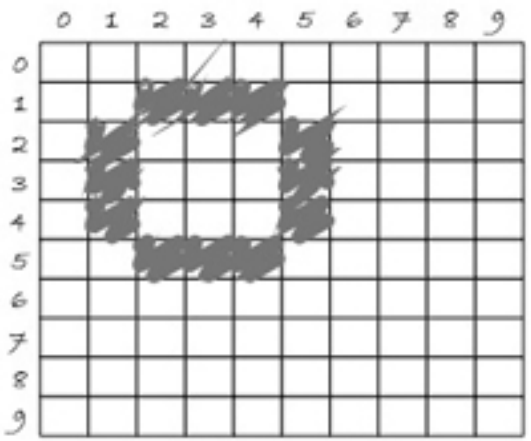
```
void setup() {
  size(480, 120);
  smooth();
}

void draw() {
  if (mousePressed) {
    fill(0);
  } else {
    fill(255);
  }
  ellipse(mouseX, mouseY, 80, 80);
}
```


STANDARD
sketch_jun29a §
void setup() {
size(480, 120);
smooth();
}
void draw() {
background(255);
if (mousePressed) {
fill(0);
} else {
fill(255);
}
ellipse(mouseX,mouseY, 80, 80);
}

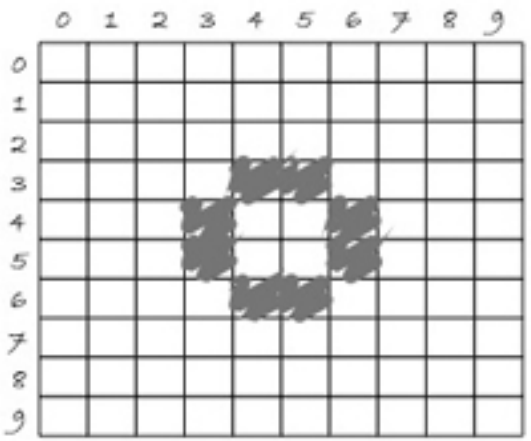

x-axis
+
+
y-axis
−
(0,0)
−
Eighth grade
x-axis
(0,0)
+
+
y-axis
Computer


x-axis
0 1 2 3 4 5 6 7 8 9
y-axis
0 1 2 3 4 5 6 7 8 9
Point B (8,3)
Point A (1,3)
Point A
x y
Point B
x y
line (1,3,8,3);

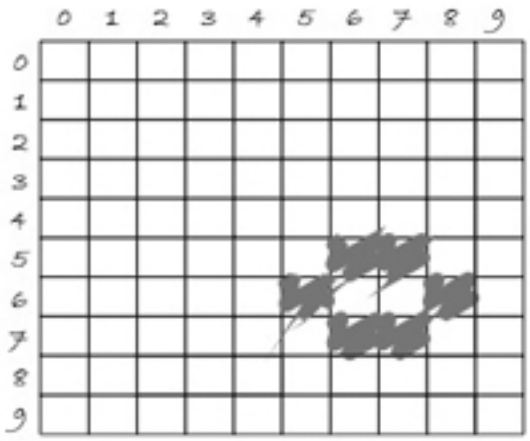

Top left
x-axis
0 1 2 3 4 5 6 7 8 9
0
1
2
3
4
5
6
7
8
9
y-axis
height
width
width height
rect (2,3,5,4);
top left
x
top left
y



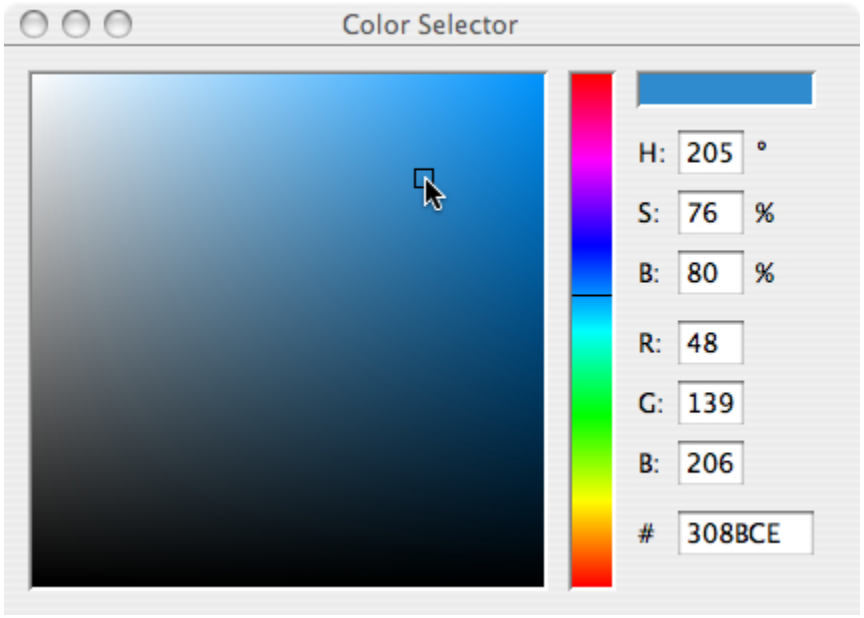
ellipseMode (CENTER);
ellipse (3,3,5,5);



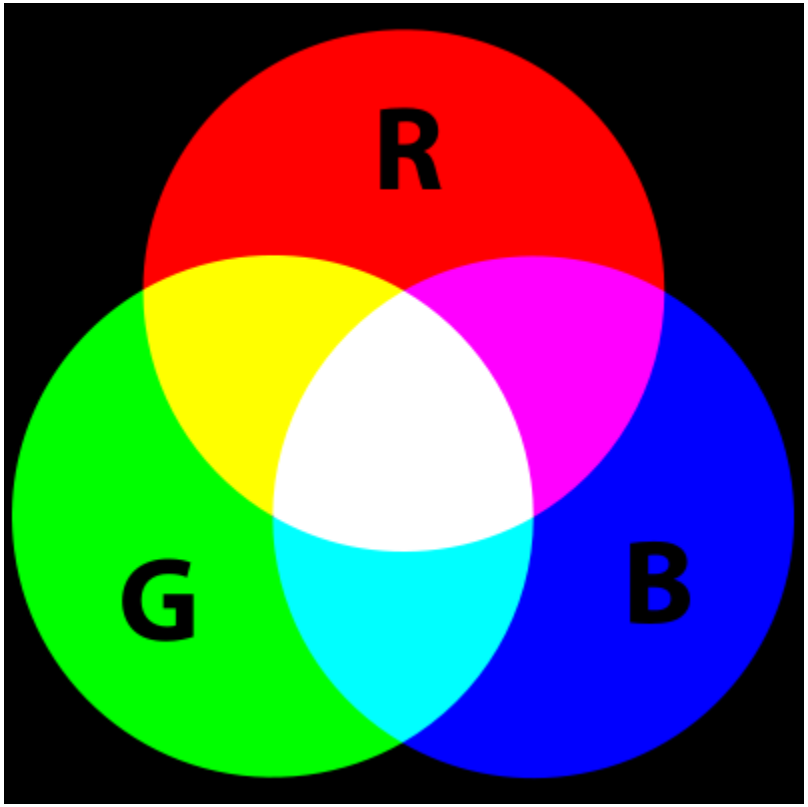
ellipseMode (CORNER);
ellipse (3,3,4,4);



ellipseMode (CORNERS);
ellipse (5,5,8,7);

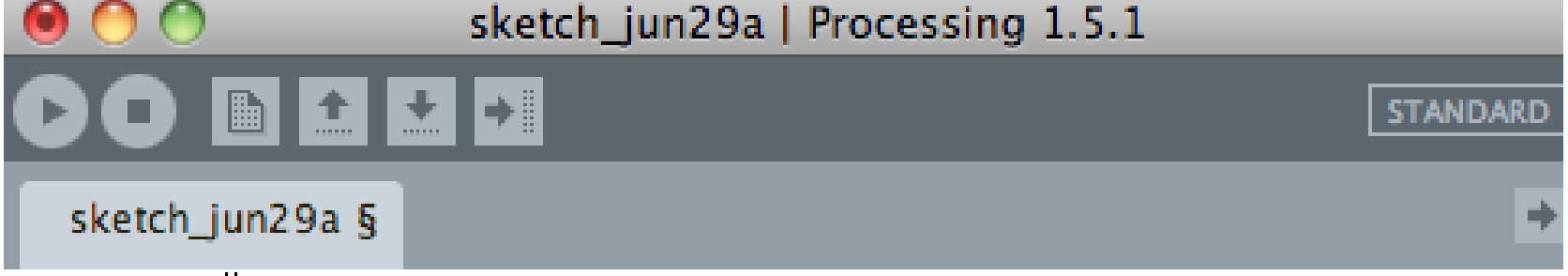

Color Selector
H: 205 °
S: 76 %
B: 80 %
R: 48
G: 139
B: 206
308BCE

2

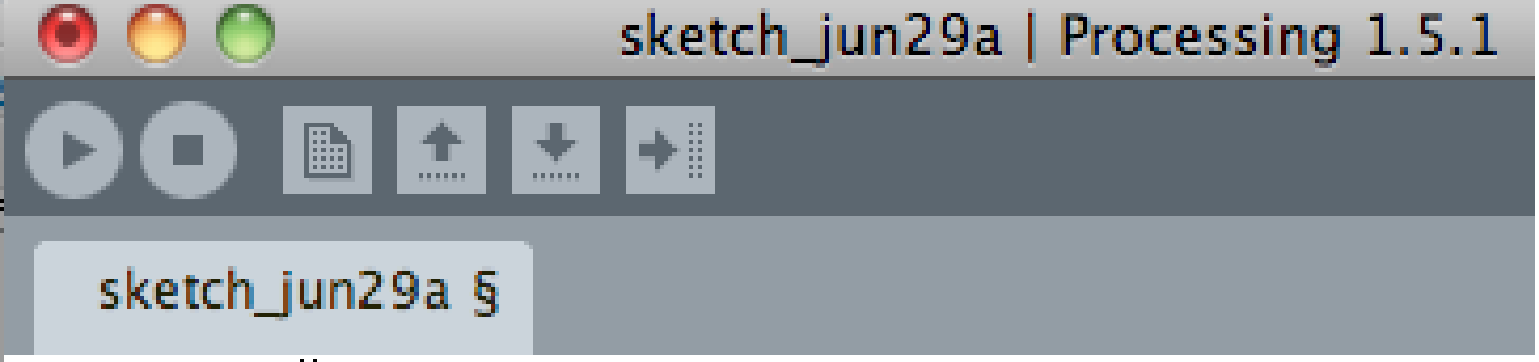


Java basiert

```
size(400, 400);
background(192, 64, 0);
stroke(255);
line(150, 25, 270, 350);
```



eine Linie zeichnen



einen Hintergrund zeichnen


```
void setup() {

    size(400, 400);
    stroke(255);
  }

  void draw() {
    line(150, 25, mouseX, mouseY);
  }

  void mousePressed() {
    background(255, 247, 3);
  }
```

FUNKTION : Wenn du die Maus klickst, verändert sich der Hintergrund

```
void setup() {
size(400, 400);
 stroke(255);
background(192, 64, 0);

}

void draw() {
 line(150, 25, mouseX, mouseY);
}
```

h i n t e r g r u n d n e u z e i c h n e n

```
void setup() {

  size(400, 400);
  stroke(255);
}

void draw() {
  line(150, 25, mouseX, mouseY);
}

void mousePressed() {
  background(255, 247, 3);
}
```

e x p o r t

File → Export Application

File → Export

Öffne **index.html** um den sketch im browser zu sehen

saveFrame("output.png")

```
size(400, 400);
// falsche art die position anzugeben

ellipse(200, 200, 50, 50);



// immer in der mitte, egal wie sich die size() line verändert

ellipse(width/2, height/2, 50, 50);
```

```
size(400, 400, JAVA2D);
size(400, 400, P2D);
size(400, 400, P3D);
size(400, 400, OPENGL);
size(400, 400, PDF, "output.pdf");
```

```
// JPEGs hineinladen
// daten von einer datei hineinladen


String[] lines = loadStrings(„irgendein.txt");

PImage image = loadImage(„meinBild.jpg");
```

libraries

Sketch → Import Library → PDF Export

Primitives

Float (32 bit information)

(floating-point numbers: 3.40282347E+38)

Int (32 bit)

Dezimal Zahlen: -2,147,483,647

Bol

(true or flase) richtig/falsch

Char

(Each **char** is two bytes (16 bits)
(typographic symbols such as A, d, and $)

VIDEO

object-oriented programming (OOP)

: : Objekte haben Eigenschaften und Funktionen

: : sie werden über „Klassen“ “classes“ definiert

www.processing.org

Beispiel online

http://www.learningprocessing.com/examples/chapter-8/example-8-2/

: : Die *class* ist die Keksform,
der Keks ist das *object*

Beispiel „cars“ pseudo code normal

Data (Global Variables):
Auto Farbe
Auto x Position
Auto y Position
Auto x Geschwindigkeit

Setup:
Initialisiere Auto Farbe.
Initialisiere Auto Standort am Anfang.
Initialisiere Auto Geschwindigkeit

Draw:
Füll den Hintergrund
Zeig das Auto am richtigen Ort mit der richtigen Farbe
Verändere die Autoposition mit der richitgen Geschwindigkeit

```
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);
      }
```

Pseudo code object orientiert

Data (Global Variables):
Auto Objekt.

Setup:
Initialisiere Auto Objekt.

Draw:
Füll den Hintergrund
Zeig das Auto Objekt.
Bewege das Auto Objekt.

http://processing.org/learning/objects/

```
Car myCar;

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

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

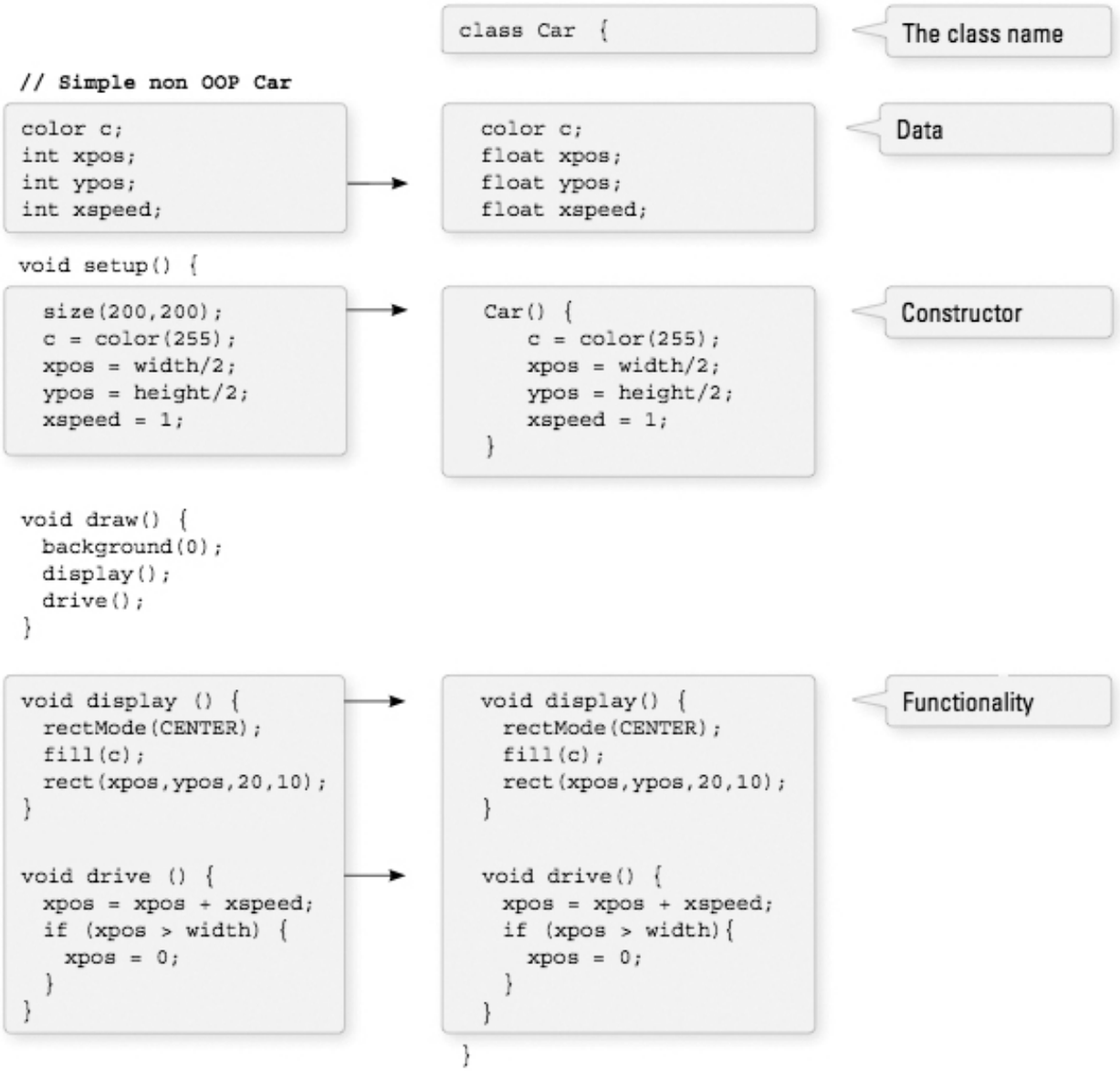
C L A S S - E L E M E N T E

1. name - mach keksform class car {

1. Data - nenne variablen

1. Constructor - mach den keks
   Car myCar = new Car();"

4. methods definier die funktionen durch methods


class Car {
The class name
// Simple non OOP Car
color c;
int xpos;
int ypos;
int xspeed;
color c;
float xpos;
float ypos;
float xspeed;
Data
void setup() {
size(200,200);
c = color(255);
xpos = width/2;
ypos = height/2;
xspeed = 1;
Car() {
c = color(255);
xpos = width/2;
ypos = height/2;
xspeed = 1;
}
Constructor
void draw() {
background(0);
display();
drive();
}
void display () {
rectMode(CENTER);
fill(c);
rect(xpos,ypos,20,10);
}
void drive () {
xpos = xpos + xspeed;
if (xpos > width) {
xpos = 0;
}
}
void display() {
rectMode(CENTER);
fill(c);
rect(xpos,ypos,20,10);
}
void drive() {
xpos = xpos + xspeed;
if (xpos > width){
xpos = 0;
}
}
}
Functionality

Class ist ein block, den du überall hinstellen kannst, solange er außerhalb von setup () und draw () ist

```
void setup() {
}

void draw() {
 }

class Car {

}
```

```
// Step 1. Deklariere das Objekt
Car myCar;
void setup() {

// Step 2. Initialisiere das Objekt.
myCar = new Car();
}

void draw()
{
background(255);

// Step 3. Rufe die Funktionen des Objekts auf
myCar.move();
myCar.display(); }
```

1. Objekt Variable deklarieren

```
Car myCar;
```

Primitives
v.s.
compex data types

komplexen Datentypen, weil sie mehrere Informationen beinhalten (die Daten und Funktionalitäten. Primitive speichern nur Daten)

Objekt initialisieren

Wenn man eine normale primitive Variable (zum Beispiel eine integer) initiert
gibst du ihr einfach einen Wert:

```
// Variable Initialization
var = 10; // var equals 10
```

Um ein Objekt zu initialisieren, muss du etwas mehr tun, nicht nur Wert zu weisen, sondern als ganzes konstruieren „construct“. Du konstruierst ein Objekt mit dem neuen Operator

```
// Object Initialization
myCar = new Car();    // The new operator is used to make a new object.
```

"myCar" ist der Objektname
"=„ zeigt an, dass wir etwas neues kreieren, ein neues Keksobjekt,
der *constructor ruft eine spezielle Funktion namens* **Car()** auf.

Diese Funktion initialisiert automatisch alle Variablen des Objekts.
Sie bekommen den Wert null, der aber überhaupt nicht „0“ heißt,
sondern einfach gar nichts. Leere.

Die Fehlermeldung "NullPointerException" kommt meist daher, dass du vergessen hast das Objekt zu initialisieren.

Nützung des Objekts

Autos fahren, Blumen wachsen, Sonnen strahlen: spezielle Objekte haben spezielle Funktionen

syntax:

variableName.objectFunction(Function Arguments);

// Functions are called with the "dot syntax".

myCar.draw();

myCar.display();

constructor für mehere objekte nützen

```
Car myCar = new Car(color(255,0,0),0,100,2);
```

ARGUMENTS

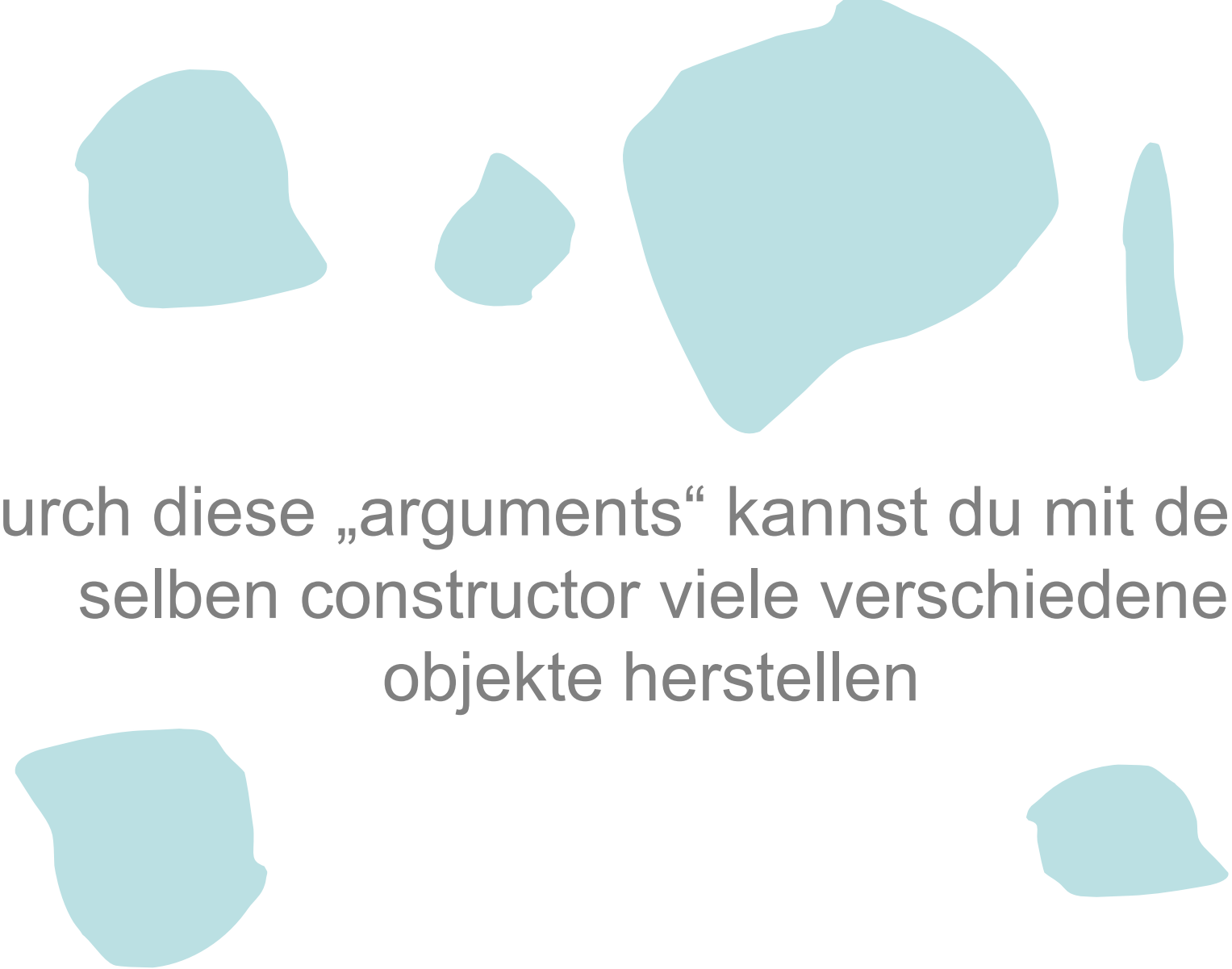
Arguments sind lokale Variablen, die nur im Inneren einer Funktion verwendet werden und mit Werten gefüllt werden, wenn die Funktion aufgerufen wird.

Sie haben nur einen Zweck:

Die Variablen in einem Objekt zu initialisieren. Das sind die wichtigen Variablen, die die eigentliche Farbe oder die temporäre Position des Autos beinhalten. Die „Arguments“ des „constructors“ sind nur *temporär und existieren nur um den Wert vom Entstehungsort des Objekts zum Objekt selbst weiterzuleiten.*

ARGUMENTS

```
Car(color tempC, float tempXpos, float tempYpos, float tempXspeed) {

      c = tempC;
      xpos = tempXpos;
      ypos = tempYpos;
      xspeed = tempXspeed;

}
```



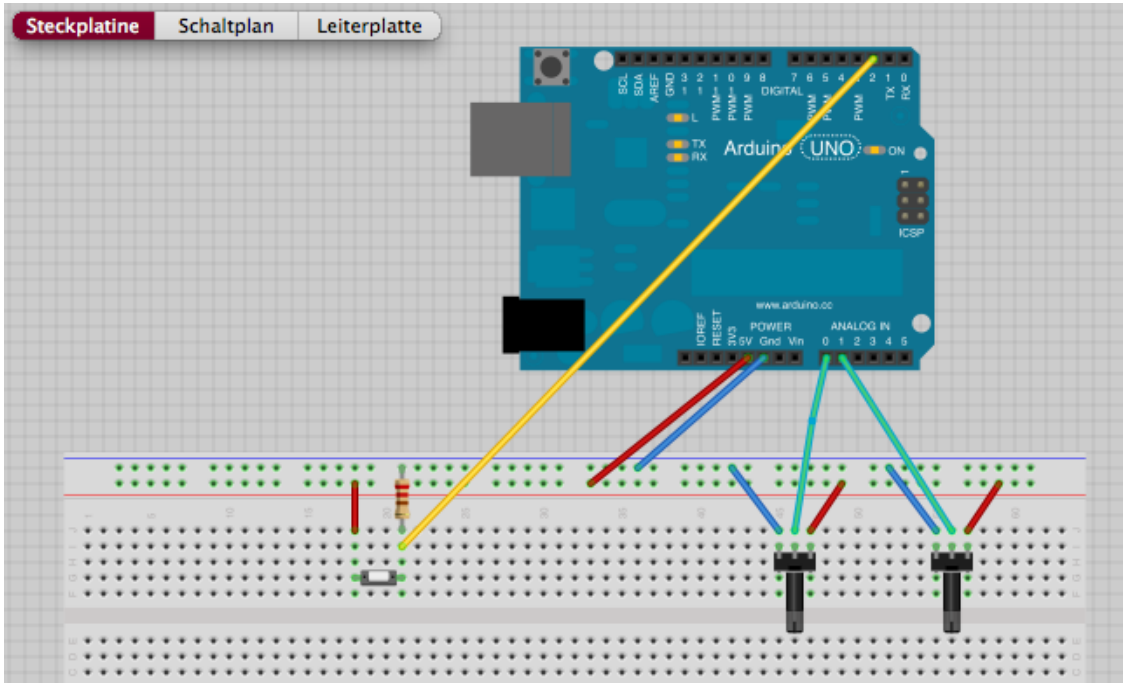
durch diese „arguments“ kannst du mit dem selben constructor viele verschiedene objekte herstellen

ganzer code:

http://www.learningprocessing.com/examples/chapter-8/example-8-2/

BOUNCING BALLS

http://www.learningprocessing.com/examples/chapter-10/example-10-2/


Steckplatine
Schaltplan
Leiterplatte
Arduino
UNO
DIGITAL
POWER
ANALOG IN
ICSP
www.arduino.cc

elektronik workshop

PD verbunden mit Arduino

PDuino

http://at.or.at/hans/pd/objects.html

http://firmata.org/wiki/Main_Page

http://arduino.cc/en/Reference/Firmata

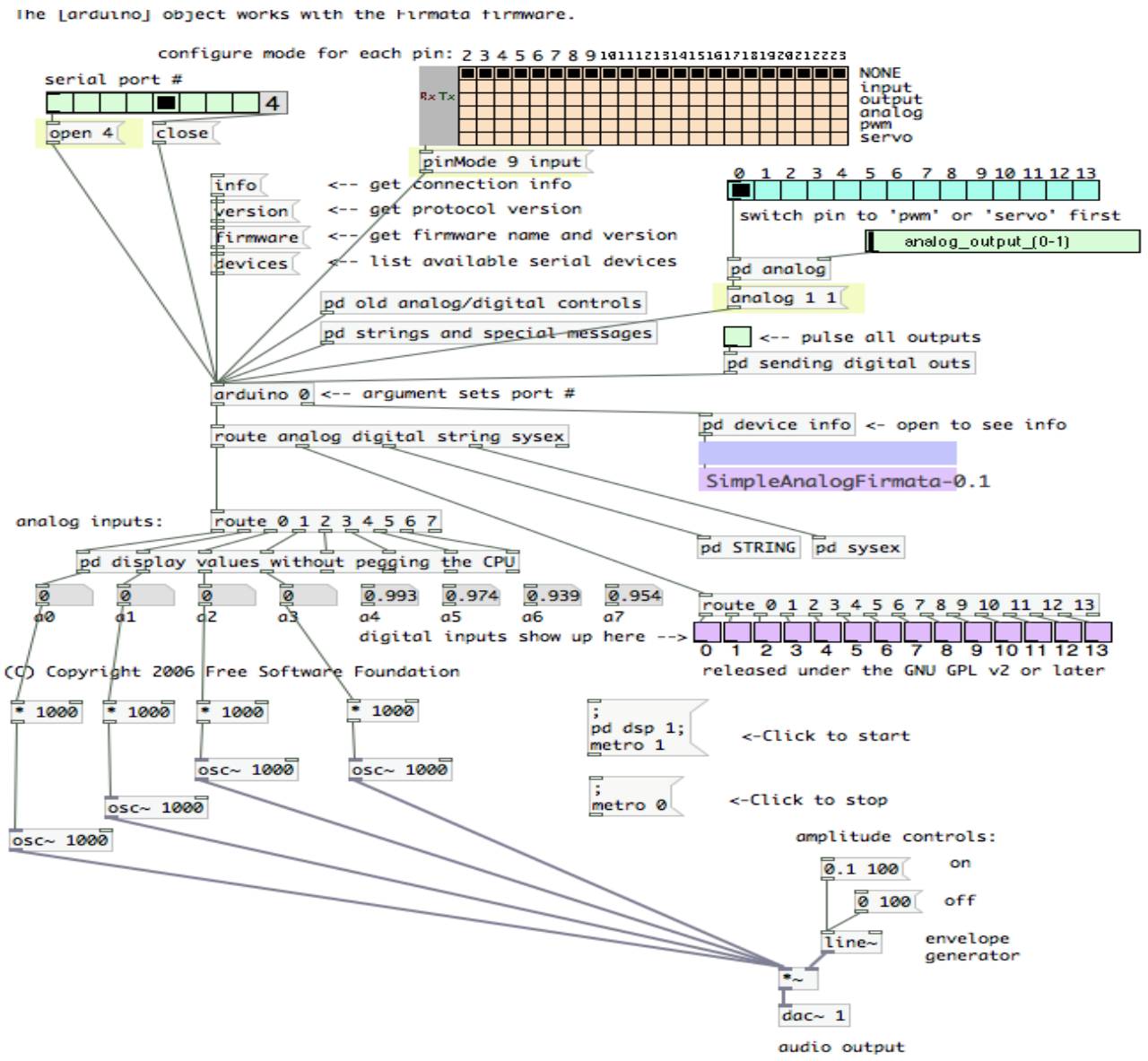
http://firmata.org

To install Firmata into Arduino, you'll need to delete the included one and replace it with this one. First find the existing 'Firmata' library and either delete or rename it. Here are some typical locations based on OS:

GNU/Linux: ~/Desktop/arduino-0011/hardware/libraries
Windows: C:\Program Files\arduino-0011\hardware\libraries
Mac OS X: /Applications/Arduino.app/Contents/Resources/Java/hardware/libraries
(to see this, right-click on Arduino.app and select "Show Package Contents")

0. delete the existing 'Firmata' library in the above location
1. unzip the Firmata zip file
2. move the included "Firmata" folder into your Arduino installation. (Do not move the whole "Firmata-2.1beta7" folder, just the included "Firmata" folder.)

Now you can launch Arduino and install a Firmata example firmware. 'StandardFirmata' is probably the best place to start. You can tell if the above update succeeded because you'll see the new 'I2CFirmata' example firmata in with the Firmata examples.


the [arduino] object works with the firmata firmware.
configure mode for each pin: 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23
NONE
input
output
analog
pwm
servo
serial port #
4
open 4
close
pinMode 9 input
info
<-- get connection info
version
<-- get protocol version
firmware
<-- get firmware name and version
devices
<-- list available serial devices
0 1 2 3 4 5 6 7 8 9 10 11 12 13
switch pin to 'pwm' or 'servo' first
analog_output_(0-1)
pd analog
analog 1 1
pd old analog/digital controls
pd strings and special messages
<-- pulse all outputs
pd sending digital outs
arduino 0 <-- argument sets port #
pd device info <- open to see info
route analog digital string sysex
SimpleAnalogFirmata-0.1
analog inputs:
route 0 1 2 3 4 5 6 7
pd STRING
pd sysex
pd display values without pegging the CPU
0 0 0 0 0.993 0.974 0.939 0.954
a0 a1 a2 a3 a4 a5 a6 a7
digital inputs show up here -->
route 0 1 2 3 4 5 6 7 8 9 10 11 12 13
0 1 2 3 4 5 6 7 8 9 10 11 12 13
(C) Copyright 2006 Free Software Foundation
released under the GNU GPL v2 or later
* 1000
* 1000
* 1000
* 1000
osc~ 1000
osc~ 1000
osc~ 1000
osc~ 1000
; pd dsp 1; metro 1
<-Click to start
; metro 0
<-Click to stop
amplitude controls:
0.1 100
on
0 100
off
line~
envelope generator
*~
dac~ 1
audio output

oscillator object