Programmierung 1
Studiengang MI / WI

- Dipl.-Inf., Dipl.-Ing. (FH) Michael Wilhelm
- Hochschule Harz
- FB Automatisierung und Informatik
- mwilhelm@hs-harz.de
- http://mwilhelm.hs-harz.de
- Raum 2.202
- Tel. 03943 / 659 338

Inhalt der Vorlesung

Überblick:

- **Erste Beispiele, Interaktion**
- elementare Datentypen
- Kontrollstrukturen
- Arrays und Funktionen
- Objekte und Methoden
- Methoden
- Algorithmen und Pseudocode
- Laufzeitverhalten
- Simulation
- Bibliotheken

Grundlegende Algorithmen und Methoden:

- Suchen und Sortieren
- Hashing
- Rekursion
- Graphen
- Dynamische Programmierung

Von Processing zu Java

- Folien basierend auf Daniel Schiffman "*Learning Processing*" und Donald W. Smith
- Folien basierend auf Vorlesung „Programmierung1" von Prof. Singer

Literatur

- Java ist auch eine Insel
  geb., mit DVD
  1308 S., 49,90 Euro,
  Galileo Computing
  ISBN 978-3-8362-1802-3

Literatur

- Grundlagen und Konzepte der Informatik, Pearson Studium 2007

  Helmut Herold, Bruno Lurz, Jürgen Wohlrab

  ISBN-10: 3827373050 ISBN-13: 978-3827373052

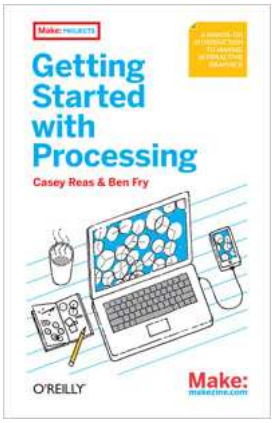
- Gumm, H.P.; Sommer, M.: Einführung in die Informatik,
  8. Auflage, Oldenbourg 2008,

  ISBN-10: 3486587242

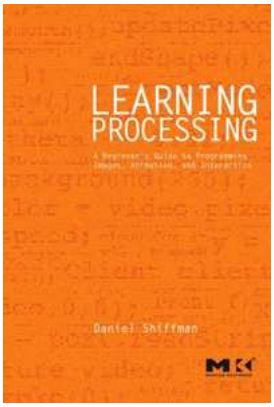
  ISBN-13: 978-3486587241

Literatur

Getting Started with Processing
Casey Reas and Ben Fry.
Published June 2010, O'Reilly Media. 208 pages. Paperback
ISBN-10: 144937980X
ISBN-13: 978-1449379803.



Learning Processing
A Beginner's Guide to Programming Images, Animation, and Interaction
Published August 2008, Daniel Shiffman.
450 pages. Paperback.
ISBN-10: 0123736021
ISBN-13: 978-0123736024



Vorlesung

- Vorlesung per Powerpoint
- Beispiele an der Tafel
- Beispiele auf der Homepage
- Führen eines Lehrtagebuches
- Abgabe der wöchentlichen Übungen
- Musterklausur am Ende der Vorlesung, Ende Dezember
- Separate Klausurvorbereitung
- Sondertermine

Termine

- **Vorlesung**
  - Montag, 08:00 bis 09:45, Raum 4.101
- **Übungen**
  - Montag, 09:45 bis 11:15
  - Räume: 9.328, 9.327, 9.326
- **Tutoren**
  - Alexander Johr
  - Philipp Müller
  - Waldemar Krause

Prüfungsleistungen

- **Klausur**
  - 120 min (Ende Januar)
- **Testat:**
  - 1. Übungsabgaben
    - Abgabe aller Aufgaben
    - 1x fehlerhafte Abgabe
    - 1x keine Abgabe
  - 2. Bestehen der **zwei** Tests im Tutorial
  - 3. Lerntagebuch (regelmäßige Abgabe)

Scripte / Aufgaben / Lösungs-Abgabe/Lerntagebuch

Vorlesung: Wilhelm_Programmierung1_WS_2012_13

Scripte →

Aufgaben →

Lösung abgeben →

Abgabe →

Scripte / Aufgaben / Lösungs-Abgabe / Lerntagebuch

Abgaben als Dateien in entsprechende Verzeichnisse in StudIP:

Vorlesung in StudIP:

Wilhelm_Programmierung1_WS_2012_13

Aufgaben bis Montag 08:00 Uhr vor der Vorlesung
(1 Woche nach der Vorlesung),
Ordner: ***Abgabe_Aufgaben***
dann Ordner der Ausgabe auswählen

Lerntagebuch:
Abgabe bis Sonnabend 12:00 Uhr in der Woche der Vorlesung

Hausaufgaben:

pde-Datei/java-Datei gezipped mit **Matrikelnummer**

- Dateiname: 12345_aufgabe01.zip

Die Zip-Datei enthält ein Verzeichnis mit dem Namen der Datei (z.B. XXXXX_01), in diesem Verzeichnis befindet sich der kommentierte Quellcode der Programme.

Lerntagebücher:

- pdf-Datei mit **Matrikelnummer** und Vorlesungsdatum als Dateiname:
- Dateiname: 12345_2012_10_08pdf

Skript und Aufgabenstellungen:

- jeweils nach der Vorlesung auf StudIP bzw. meiner Homepage

Lerntagebuch

vertieftes Verständnis durch regelmäßige Nachbearbeitung und Reflexion

Diese Reflexion kann sich auf alle Aspekte der Lerninhalte, die in Zusammenhang mit der betreffenden Sitzung stehen, beziehen, also auf die Lektüre des Hintergrundtextes, auf die Präsentation. Aus dieser Gesamtmenge potentieller Lernanstöße sollen diejenigen ausgewählt und bearbeitet werden, die Ihnen persönlich als besonders bedeutsam, interessant, fragwürdig oder neuartig empfunden werden.

Bewusstsein für den eigenen Lernprozess fördern

In diesem Zusammenhang ist auch die *Interaktion* zwischen dem eigenen Lernprozess und den im Seminar zum Einsatz kommenden Lehrmethoden von Interesse: Warum habe ich das Gefühl, dass mir eine bestimmte Form der Stoffvermittlung besonders "liegt" und warum nicht? Geht das den anderen Seminarteilnehmern und Seminarteilnehmerinnen auch so oder nicht?

Methode des Lernens

Die regelmäßige schriftliche Explikation der eigenen Gedanken in kompakter Form stellt auch außerhalb des Seminarkontexts eine sinnvolle Form der Förderung von Lernprozessen dar. Die "Verschriftlichung" der eigenen Gedanken kann insbesondere helfen, eigene Ideen zu generieren und zu entwickeln. Die Erstellung eines persönlichen Lerntagebuchs in diesem Seminar ist daher auch als das Einüben einer "Technik" des aktiven, selbstgesteuerten Lernens zu sehen.

Warum heißt das Lerntagebuch Lerntagebuch?

1. die Regelmäßigkeit der Aufzeichnungen, die es - in der Rückschau - ermöglichen soll, die eigene "Lerngeschichte" in Zusammenhang mit dem Seminarbesuch schnell zu

rekonstruieren. Das LT hat also, ähnlich wie ein normales Tagebuch, eine Art "Bilanzfunktion".

2. zur Führung ist ein persönlicher "Stil" der Aufzeichnung zu !nden. Es soll sich beim LT wie bei einem normalen Tagebuch um ein fortgesetztes Zwiegespräch des Autors/der Autorin mit sich selbst handeln. Es gibt daher keine allgemeinverbindliche Form, wie man es "richtig" macht.Programmierung 1

Die Analogie mit dem normalen Tagebuch hat aber natürlich auch ihre Grenzen. Besonders offensichtlich: Das LT wird geführt, um es zu veröffentlichen, denn es soll abgegeben werden. Das schränkt die Privatheit naturgemäß ein.

Rückmeldefunktion für den Dozenten

Neben den obigen Zielen, die unmittelbar auf Gewinne bei den Lernenden abstellen, hat das Lerntagebuch auch eine Rückmeldefunktion für den Lehrenden. Die Lerntagebücher können zum Einen explizit bewertende Stellungnahmen enthalten, zum Anderen liefern sie aber auch in den stärker deskriptiven Teilen Hinweise darauf, welche inhaltlichen Gesichtspunkte von den Lernenden auf welche Weise aufgenommen wurden und wo es eventuell zur Entwicklung von Fehlkonzepten gekommen ist.

Das Lerntagebuch kann dem Lehrenden Informationen in einer Breite liefern, wie sie auf anderem Wege kaum zu gewinnen sind.

- **Im Lerntagebuch werden regelmäßig folgende Fragen beantwortet:**
- Sind mir Bezüge und Anknüpfungspunkte zwischen dem Thema der Stunde und aus anderen Fächern/Seminaren bereits bekannten Theorien, Befunden oder Methoden aufgefallen?
- Erscheinen mir Hintergrundtext und Vertiefungstext stimmig und aufeinander bezogen? Wirkt die empirische Arbeit methodisch sauber und überzeugend?
- Welche Sachverhalte erscheinen mir so wichtig, dass ich sie noch einmal mit eigenen Worten auf den Punkt bringen möchte?
- Welche weiterführenden Fragen wirft das Gelernte auf? Regt es mich zu Gedanken an, die über den Stoff im engeren Sinne hinausführen?
- Welche zentralen Konzepte erscheinen mir so wichtig und nützlich, dass ich sie gerne behalten möchte? Kann ich diese kurz und prägnant definieren?
- Fallen mir Beispiele aus meiner eigenen (biografischen) Erfahrung ein, die das Gelernte illustrieren, bestätigen, oder ihm widersprechen?
- Welche Aspekte des Gelernten fand ich interessant, nützlich, überzeugend, und welche nicht? Warum?
- Welche Fragen blieben offen? Was erschien mir unklar oder auch falsch?
- Welche Aspekte des Gelernten kann ich bei gegenwärtigen oder zukünftigen Tätigkeiten selber nutzen? Wie könnte eine solche Nutzung aussehen?
- Habe ich Erfahrungen oder Beobachtungen gemacht, die mir bei zukünftigen Präsentationen helfen können?

Formale Anforderungen an das Lerntagebuch (LT)

Umfang:

Das LT soll für jede Seminarsitzung einen eigenen Abschnitt enthalten. Können Sie an einer Sitzung nicht teilnehmen, dann beziehen sich die Eintragungen im LT für diesen Termin auf die Lektüre eines Hintergrundtexts. Die Länge der Abschnitte sollte im Schnitt nicht unter einer Textseite betragen (bei üblicher Formatierung, also z.B. Schrift: 12pt; Zeilenabstand: 1 1/2; Seitenränder 2-3 cm).

Abgabe:

Das LT wird konsekutiv jeweils wenige Tage nach der Sitzung als PDF-Datei durch Hochladen auf StudIP in das aktuelle Abgabeverzeichnis eingereicht.

Beurteilungskriterien:

Es widerspräche dem Grundgedanken des LT, seinen *Inhalt* ("Was ist gelernt worden?") bewerten zu wollen. Für die Anerkennung zur Scheinvergabe gilt, dass das LT den Versuch einer ernsthaften Auseinandersetzung mit den Themen widerspiegelt. Nicht akzeptiert werden daher Texte,

- die ausschließlich stichpunktartig verfasst sind,
- die sich zu eng an ein in der Sitzung verteiltes Hand-out anlehnen,
- die zu kurz sind, d.h. im Schnitt deutlich weniger als eine halbe Seite pro Sitzung umfassen,
- die extreme formale Mängel aufweisen.

Programmiererfahrung ?

- Java
- C++
- C#
- Delphi
- Basic
- PHP
- HTML / CSS / Java-Script
- Haskell
- Lisp
- Prolog
- Assembler

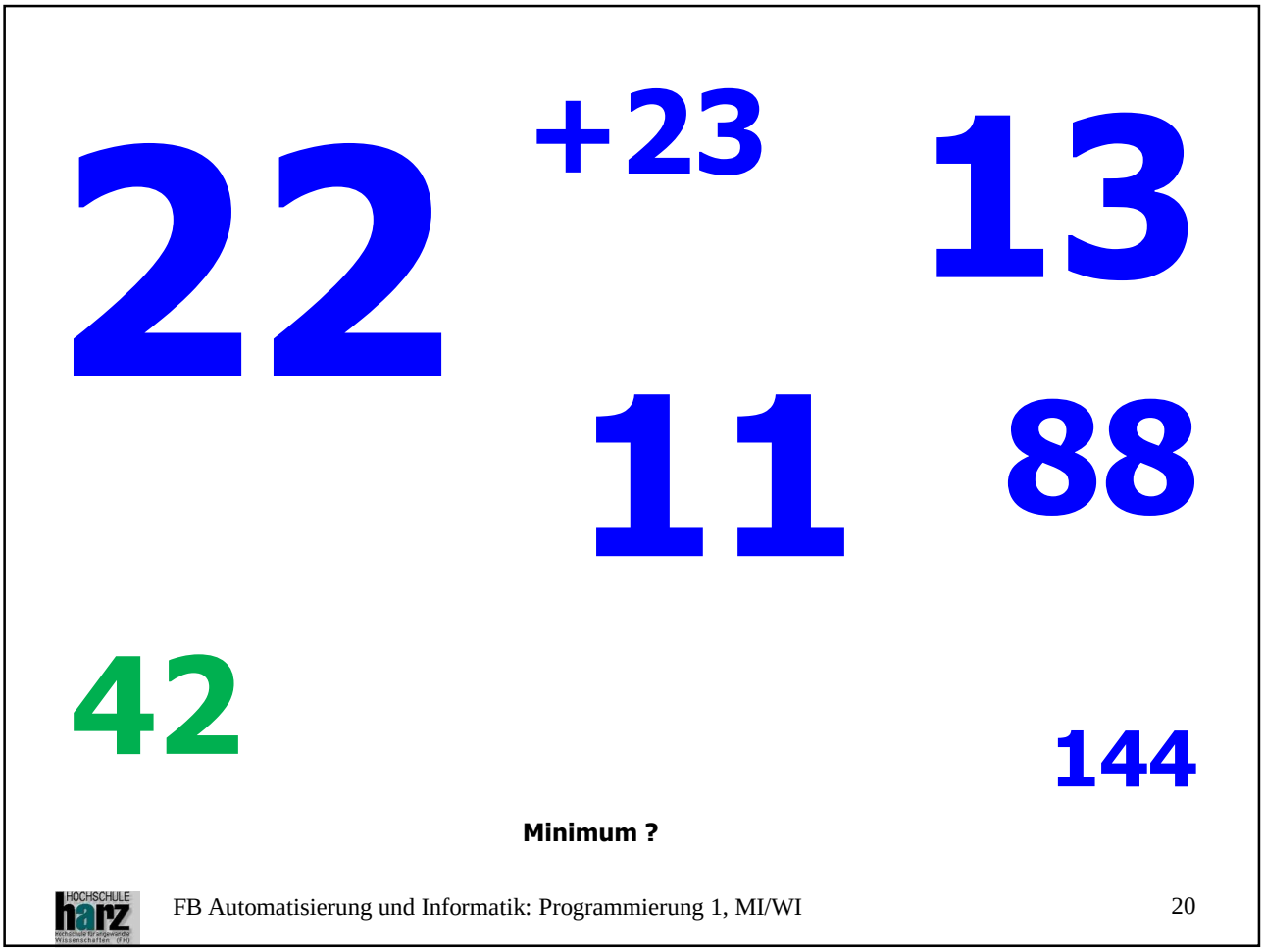
- **Informatik** **?**
- Problemanalyse ?
- PC [1] ?
- Algorithmus ?
- Codierung ?
- Debugging ?
- Programm ist fertig ?

11 -23 3

88 22

144 42

Maximum ?


22
+23
13
11
88
42
144
Minimum ?

+23

13

22

11

88

1126

42

154

Welche Zahl ist durch drei teilbar ?

http://processing.org Info zu Processing, Tutorien

Pixels (picture elements)

Angabe der Pixelkoordinaten

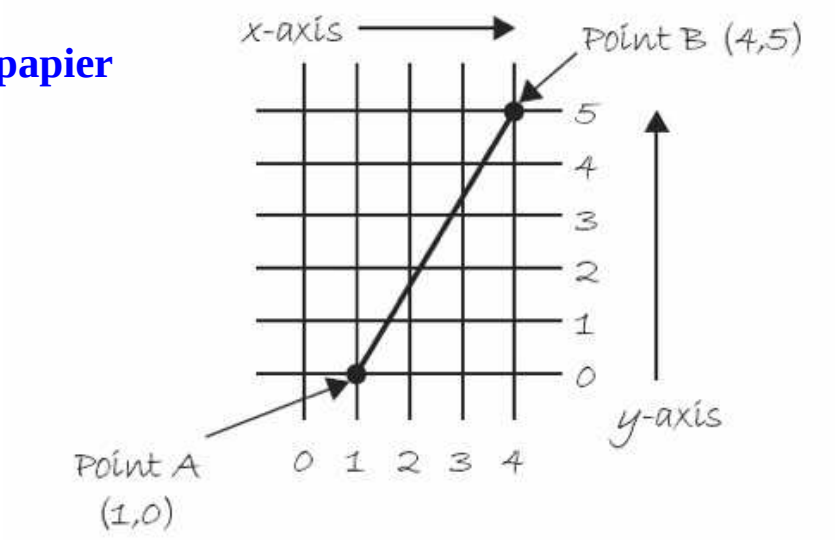
Grundformen:

- Punkt
- Linie
- Rechteck
- Ellipse
- Dreieck (Triangle)
- Viereck, Trapez, Quad

Farbe:

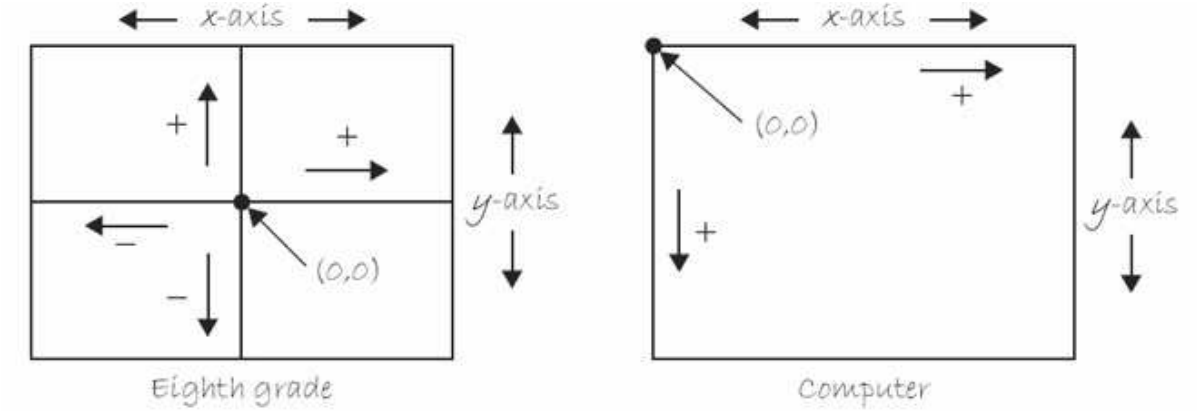
- Graustufen
- RGB
- Transparenz, Alpha-Wert

kariertes Zeichenpapier



- jeder Punkt auf dem Bildschirm ist ein Pixel.
- er hat eine Position: (x,y)

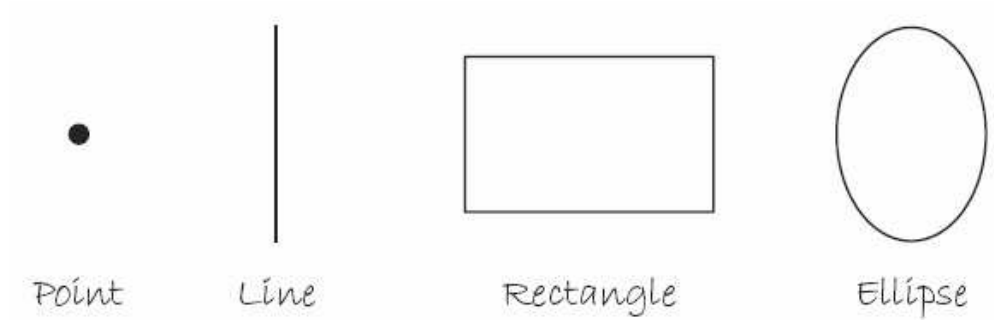
Koordinatensysteme auf Computer und in Mathematik unterscheiden sich



linke obere Ecke ist (0, 0).

- X läuft von links nach rechts
- Y läuft von oben nach unten

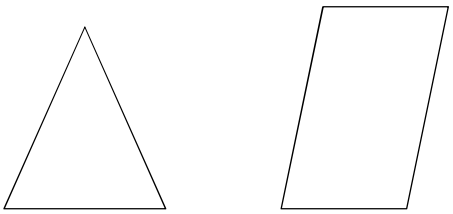
Einfache Formen zum Zeichnen einer Grafik



Welche Information wird benötigt um Formen anzugeben?

- Punkt: x und y
- Linie: Zwei Endpunkte?
- Rechteck: Zwei Ecken? **Oder**?
- Ellipse: ????

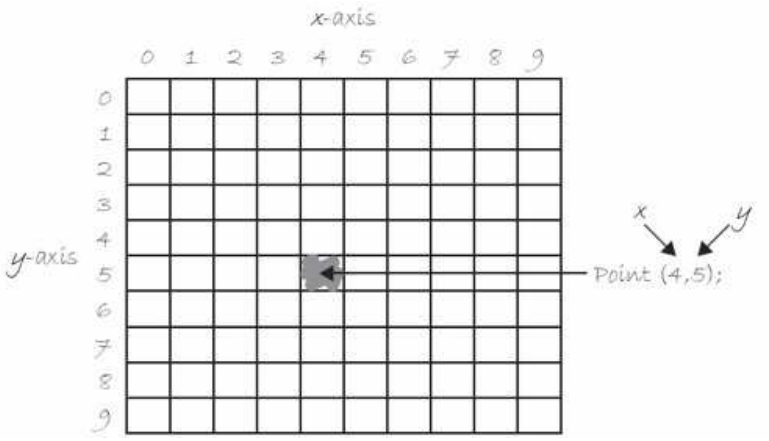
Einfache Formen zum Zeichnen einer Grafik



Welche Information wird benötigt um Formen anzugeben?

- Triangle: x1, y1 x2,y2 und x3,y3
- Viereck: x1, y1 x2,y2 x3,y3 und x4,y4

Punkt

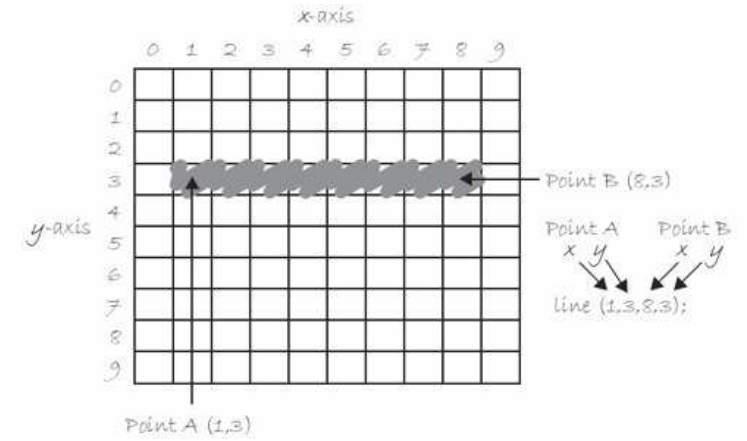


Beachte: x kommt vor y

In Processing: point(x, y);

- klein geschrieben, Englisch
- zwei “Parameter” in Klammern, durch Komma getrennt
- Strichpunkt am Ende

Linie

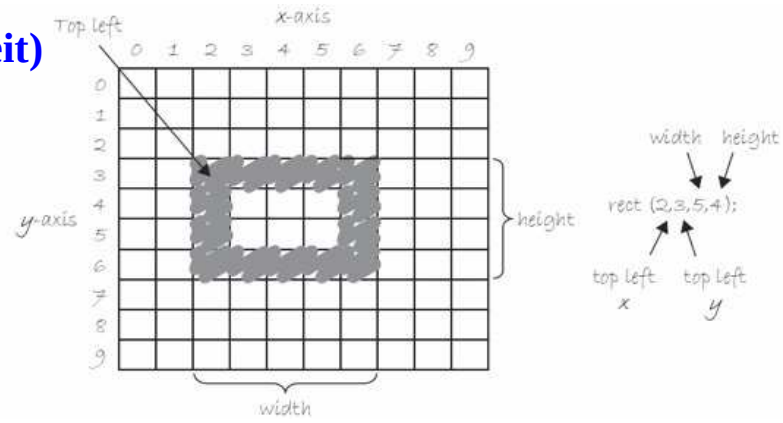


Zwei Punkte: A und B

in Processing: line(x1, y1, x2, y2);

- klein geschrieben, Englisch
- vier “Parameter” in Klammern, durch Kommata getrennt Strichpunkt

Rechteck (1. Möglichkeit)



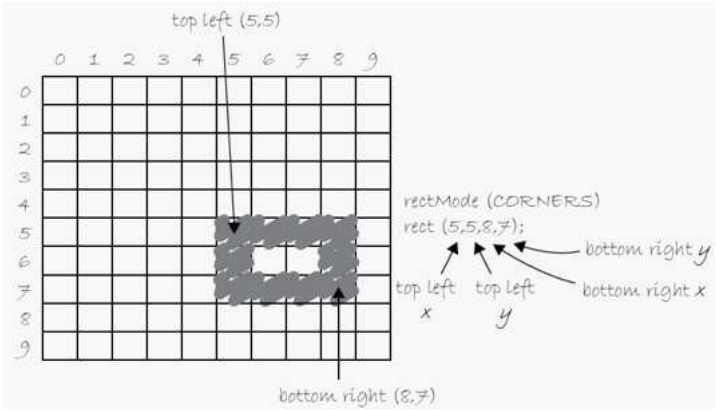
Ausgehend von Ecke: Ein Punkt für linke obere Ecke

In Processing: rect(x, y, breite, hoehe);

- Vier Parameter
- Strichpunkt

Beachte: Vermeide Umlaute und Sonderzeichen, nicht unbedingt portabel (Mac, Windows)

Rechteck (2. Möglichkeit)



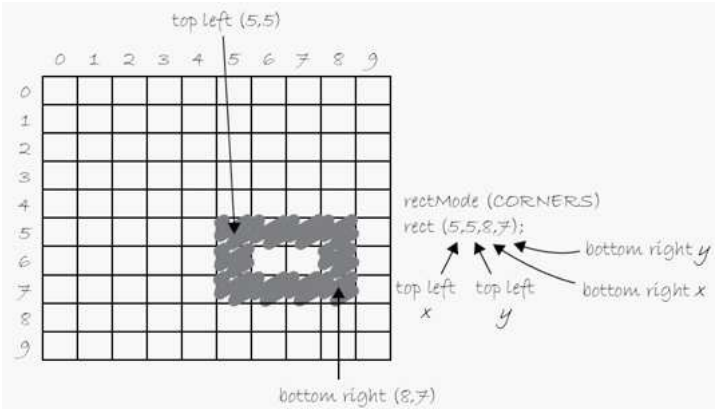
Ausgehend von Ecke: Durch Punkte (Top/Left/Bottom/Right)

In Processing:

- **rectMode(CORNERS);**
- rect(x1, y1,x2, y2);

Beachte: Processing unterscheidet Gross- und Kleinschreibung

Rechteck (3. Möglichkeit)

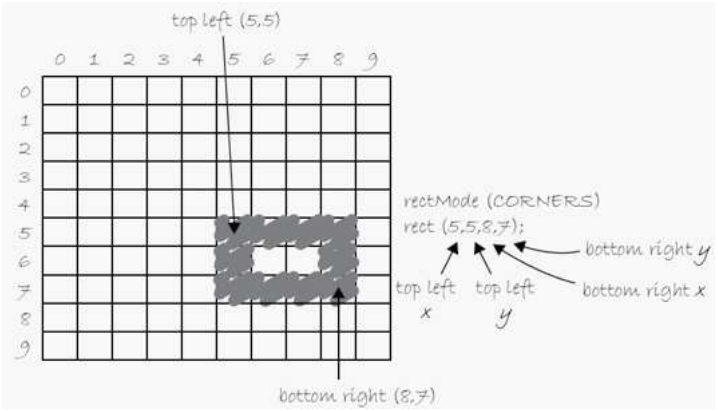


Ausgehend von Ecke: Durch Punkte (Top/Left/Bottom/Right)

In Processing:

- **rectMode(CENTER);**
- rect(x1, y1,width, height);

Rechteck (4. Möglichkeit)

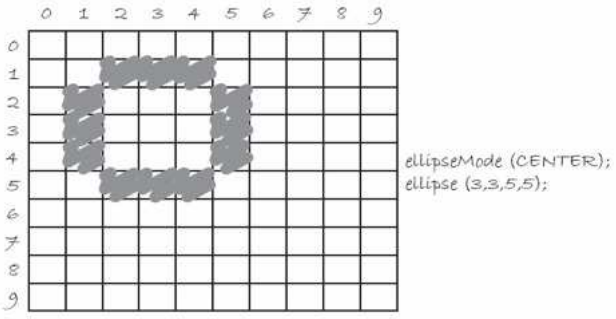


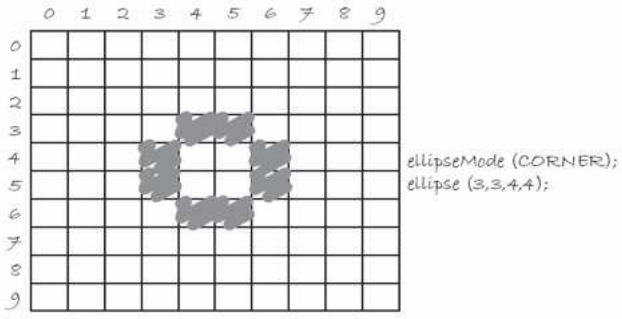
Ausgehend von Ecke: Durch Punkte (Top/Left/Bottom/Right)

In Processing:

- **rectMode(RADIUS);**
- rect(x, y,width/2, height/2);

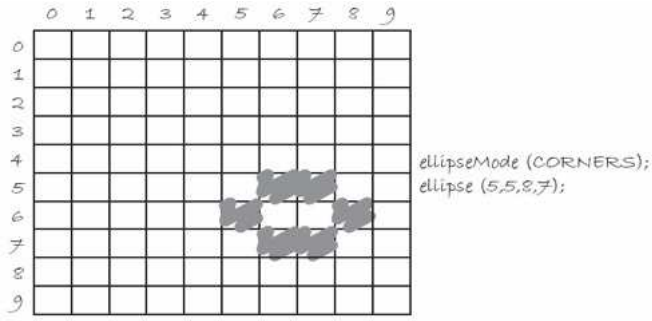
Ellipsen





analog zum Rechteck:

- CENTER: x, y, breite, hoehe
- CORNER: x, y, breite, hoehe
- CORNERS: x1, y1, x2, y2
- RADIUS: x,y, breite/2, höhe/2



- Beachte: Zeichnet Ellipse in umgebendes Rechteck (bounding box)
- Kreis ist Sonderfall der Ellipse: breite = hoehe

Rechtecke und Ellipsen

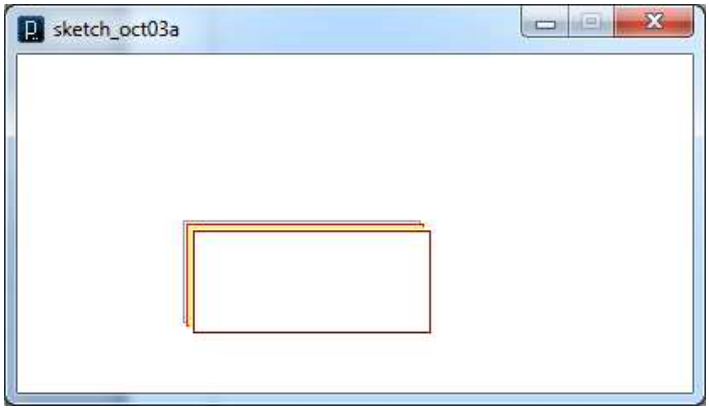
Rechteck:

- rectMode(CENTER): x, y, breite, hoehe
- rectMode(CORNER): x, y, breite, hoehe
- rectMode(CORNERS): x1, y1, x2, y2
- rectMode(RADIUS): x,y, breite/2, höhe/2

Elippse:

- ellipseMode(CENTER): x, y, breite, hoehe
- ellipseMode(CORNER): x, y, breite, hoehe
- ellipseMode(CORNERS): x1, y1, x2, y2
- ellipseMode(RADIUS): x,y, breite/2, höhe/2

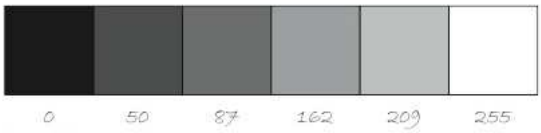
Größe



Die Größe der Zeichenfläche (“canvas”) kann zu Beginn vorgegeben werden:

- size(breite, hoehe);
- Kann nicht verändert werden
- Nutzen Sie etwa 400 x 400.

Graustufen: ein Parameter



Die Farbe (hier in Graustufen) von Linien und Flächen kann individuell gesetzt werden:

- 0 entspricht schwarz (keine Helligkeit)
- 255 entspricht weiß (maximale Helligkeit)

Parameter:

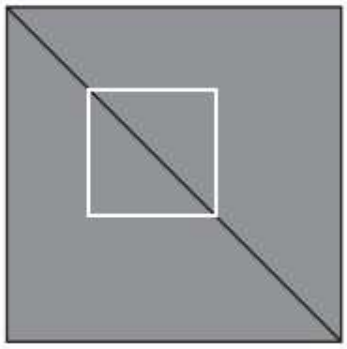
```
background(255);          // Hintergrund weiss
stoke(0);                 // Rand wird schwarz
fill(150);                // Flaeche wird grau
rect(50, 50, 75, 100);    // nutzt die letzte Einstellungen
```

Kommentare: von // bis Zeilenende

/* Kommentar */

Beispiel Graustufen

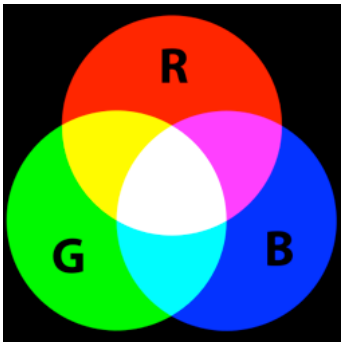
```
background(150);
stroke(0);
line(0,0,100,100);
stroke(255);
noFill();
rect(25,25,50,50);
```



- falls noFill() aktiv ist, werden Flächen nicht ausgefüllt.
- fill setzt die aktuelle Farbe des „Musters“
- Immer **VORHER** setzen

RGB Farbe

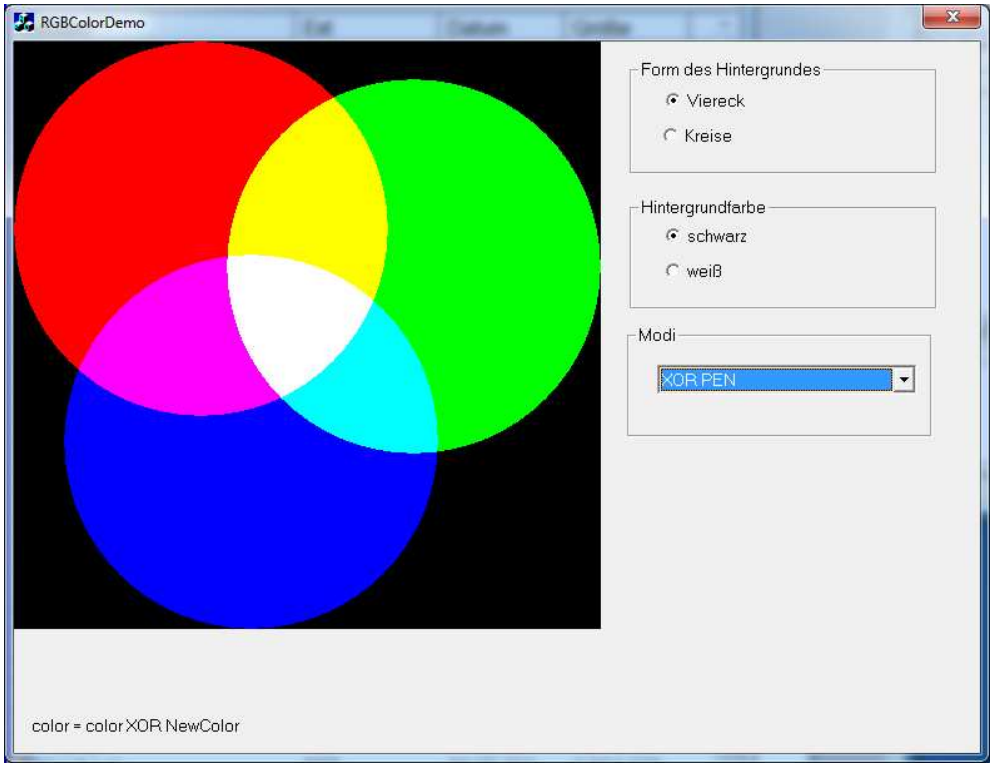
Additive Farbmischung



- rot + grün = gelb
- rot + blau = purpur
- grün + blau = cyan
- rot + grün + blau = weiß
- keine Farben = schwarz
- 0.5·rot + 0.5·grün + 0.5·blau = ?

- jeder Farbkanal (R,G,B) hat einen Wert zwischen 0 und 255
- 0: kein Beitrag dieses Farbkanals
- 255: maximaler Beitrag dieses Farbkanals

RGB Farbe: RGBColorDemo.exe

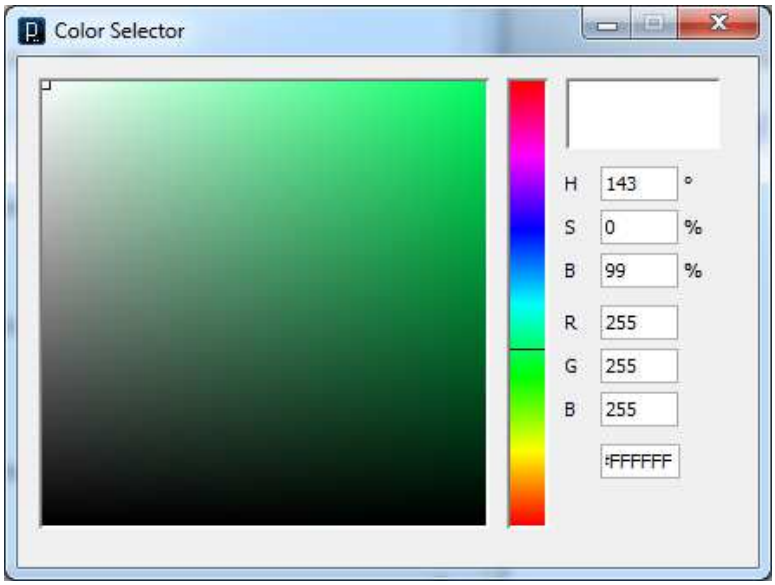


RGB Farbe durch direkte Wahl

```
background(255);
background(255, 255, 255);
noStroke();                 // kein Rand
fill(255,0,0);              // (Hell)Rot
ellipse(20,20,16,16);
fill(127,0,0);              // Dunkelrot
ellipse(40,20,16,16);
fill(255,200,200);          // Rosa
ellipse(60,20,16,16);
```

- filll(), background(), stroke() können mit jeweils 3 Parametern
- (Farbe statt Grauwerte) genutzt werden.

Processing bietet Hilfe bei der Farbauswahl:



eingebauter Farbwähler:

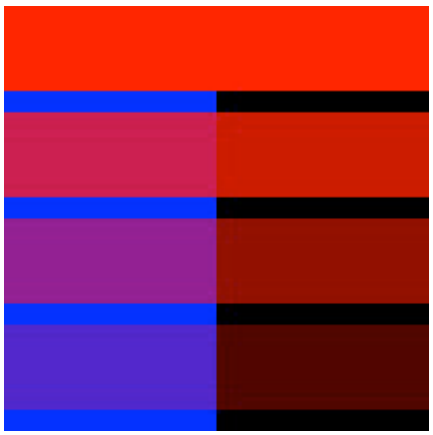
Menü Tools -> Color Selector

Transparenz

```
// 50% opacity.
fill(255,0,0,127);

// 25% opacity.
fill(255,0,0,63);

rect(0,150,200,40);
```



Vierter Parameter (genannt Alpha):

- 0 bedeutet Transparent
- 255 bedeutet undurchsichtig (opaque)
- Kein vierter Parameter entspricht undurchsichtig

Zusammenfassung:

Pixel sind Punkte auf dem Bildschirm

- X und Y Koordinaten beginnen bei 0 oben links
- Die Größe des “canvas” kann zu Beginn gesetzt werden

Es stehen einfache Formen zur Verfügung:

- Punkt, Linie, Rechteck, Ellipse

Formen können in unterschiedlichen Modi gezeichnet werden:

- CENTER, CORNER, CORNERS, RADIUS

Stroke, Fill, Background können unterschiedlich gesetzt werden:

- Graustufen: ein Parameter
- RGB: drei Parameter für Farben
- RGBA: Transparenz mit viertem Parameter (alpha)

Der Beginn des Codierens

Processing:

- Installation
- Menüs
- Code schreiben
- Fehler
- Referenz
- der Play-Knopf
- der erste Sketch

Beispiel: „grafische“ Addition zweier Zahlen

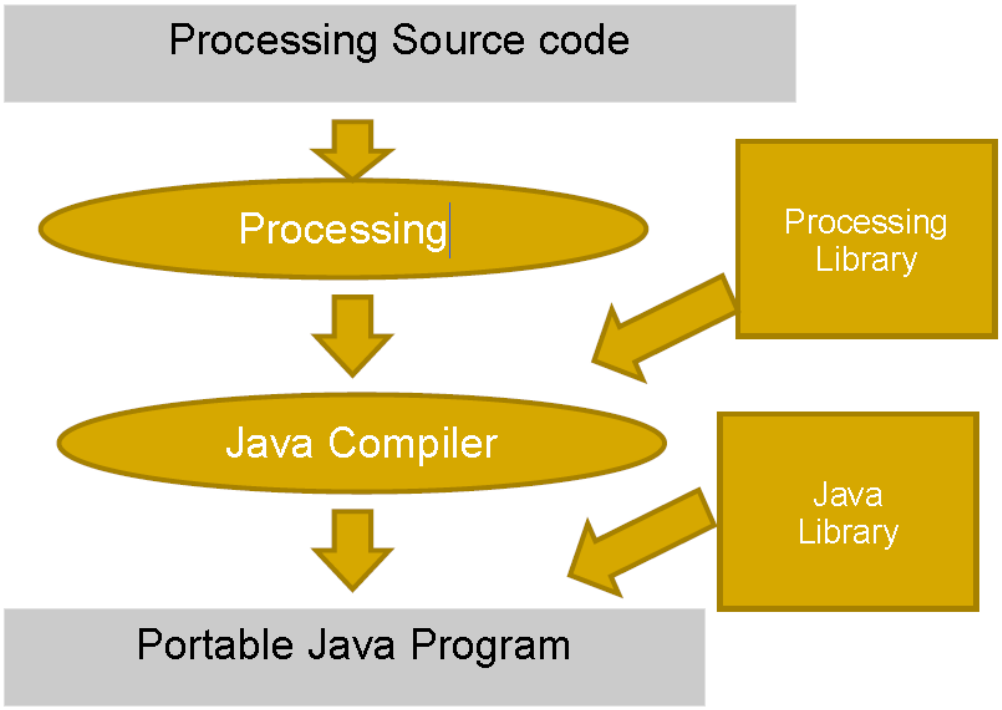
Processing:

- a=20
- b=30
- Zeichne 1. Linie
- Zeichne 2. Linie
- Berechne die Summe von a und b
- Speichern des Ergebnisses in c
- Zeichne 3. Linie

Wo kommt Java ins Spiel?

- Processing ist eine auf Java basierende Programmierumebung
- Processing benötigt das Java SDK (Software Development Kit)
- Processing besitzt eine eigene Grafikbibliothek
- Alle Java-Bibliotheken können auch in Processing genutzt werden

Wo kommt Java ins Spiel?

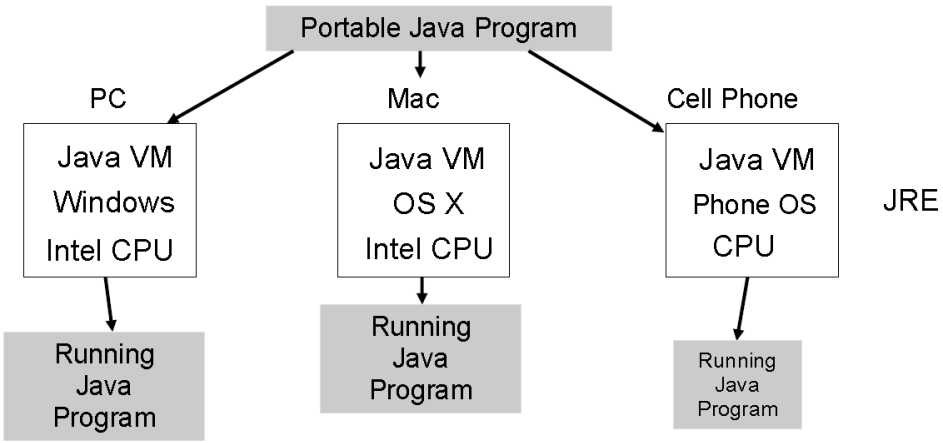


Virtuelle Java-Maschine (JVM)

Java wurde für entworfen “eingebettete” Systeme um eine virtuelle Maschine herum entworfen

einmal schreiben - läuft überall

genannt: JRE (Java Runtime Environment)



Sketch Menü Optionen

Processing hat eine *Sketch* genannte
Programmierumgebung ('PDE'):

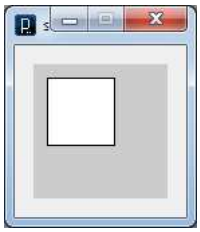
- **File**: New, Open, Quit, Examples!
- **Edit**: Copy, Paste, Select, Indent, Comment…
- **Sketch**: Run, Stop, Show Sketch folder…
- **Tools**: Auto format, Color chooser…
- **Help**: Getting Started, Reference, Find in Reference

PDE

Toolbar

abspielen

Anzeigefenster mit der Default-Size

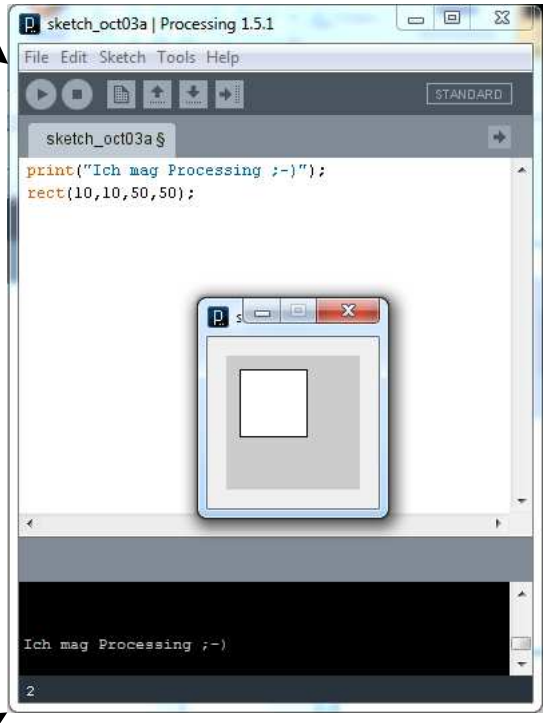


Text Editor

Statuszeile

Textausgabe

Zeilennr.



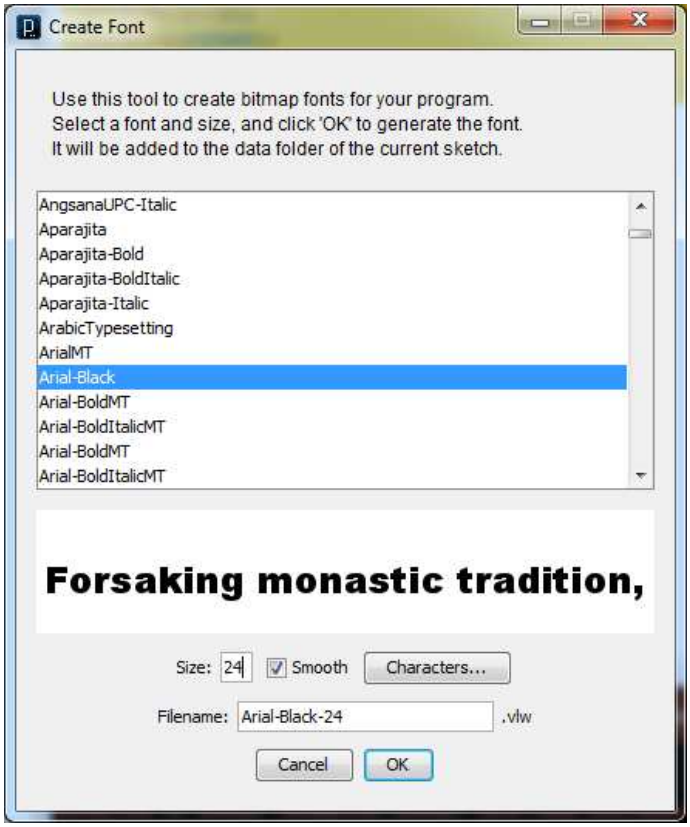
PDE: Ändern der Einstellung: File/Preferences

Schrift

- Menü Tools
- Eintrag „createFont“

- Auswahl des Fonts
- Schriftgröße
- Name des Fonts
- Speichern mit „OK“

- Datei „Arial-Black-24.vlw“ wird im Ordner „data“ erzeugt



Schrift

- PFont font; // Variable für den Font

- Der Font muss aber vorher erzeugt werden
- Danach wird er im Programm geladen

- font = loadFont("Arial-Black-24.vlw");

- textFont(font, 32);
- fill(color);
- text("hier ist ein Text",10,10);

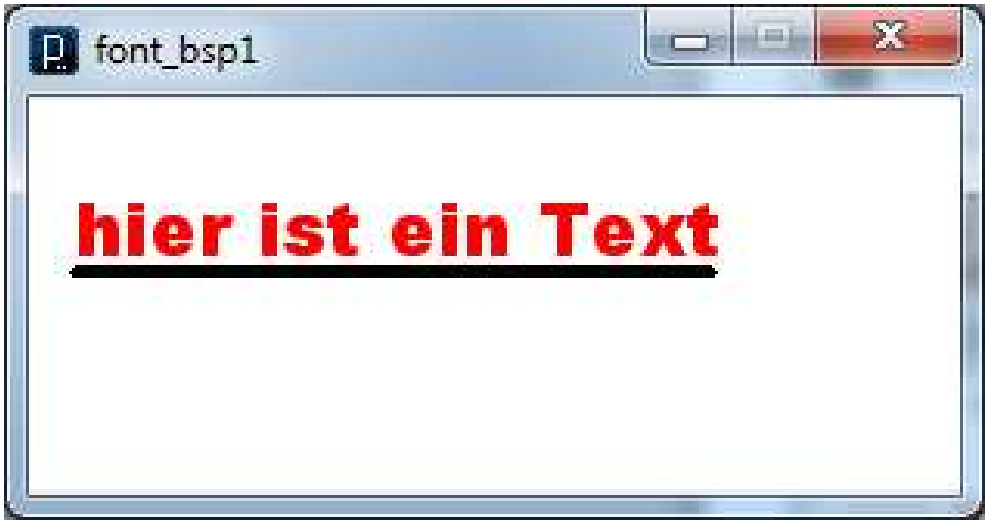
1. Schrift-Beispiel

```
PFont font;
void setup() {
  size(225, 425); // hier
  font = loadFont("Arial-Black-24.vlw");
  textFont(font);
  strokeWeight(4);
}
void draw() {
  textFont(font, 24);   // Fontsize kann geändert werden
  background( 255);
  fill(255,0,0);
  text("hier ist ein Text", 15, 50);
  line(15,55,  220, 55);
}
```

2. Schrift-Beispiel

```
PFont font;
void setup() {
  size(225, 425); // hier
  font = createFont("Arial",24);
  textFont(font);
  strokeWeight(4);
}
void draw() {
  textFont(font, 14);   // Fontsize kann geändert werden
  background( 255);
  fill(255,0,0);
  text("hier ist ein Text", 15, 50);
  line(15,55,  220, 55);
}
```

Schrift-Beispiel



Zusammenfassung:

- Processing benötigt Java
- Processing hat ein einfaches IDE, genannt PDE
- Sketches können gespeichert und kopiert werden
- Hilfe:
  - Referenz (alle Schlüsselwörter)
  - Find in Reference
- Findet den ersten Fehler und markiert die Zeile darüber

Interaktion

Programmaufbau

```
// Initialisierung
void setup() {
  size(300, 300);
  background(255);
}
// ständig oder noLoop()
void draw() {
  fill(255, 0, 0);
  line(10, 20, 300, 300);
}
```

Interaktion

- Interaktionen mit der Maus
  - mousePressed // Status, einmalig
  - mouveMoved // Status, ständig
  - mouseReleased // einmalig bei MouseReleased
  - mouseClicked // einmalig bei MouseReleased
    nicht bei MouseDrag

- Interaktion mit der Tastatur
  - keyPressed // Methode: Status, einmalig
  - keyReleased // Methode: Status, einmalig
  - keyPressed // als bool‘sche Variable

```
void mousePressed() {
  println("mousePressed, X:"+mouseX+"  Y:"+mouseY);
}
void mouseMoved() {
 println("mouseMoved, X:"+mouseX+"  Y:"+mouseY);
}
void mouseReleased() {
  println("mouseReleased, X:"+mouseX+"  Y:"+mouseY);
}
void mouseClicked() {
  println("mouseClicked, X:"+mouseX+"  Y:"+mouseY);
}
void keyPressed() {
  println("keyPress, key:"+key);
}
void keyReleased() {
  println("keyReleased, key:"+key);
}
```

Programmfluss

Spiele werden mit Schleifen geschrieben:

- analog einem Wettrennen:
  - Vorbereitung (Schuhe anziehen, ...)
  - Wiederhole: setze einen Fuß vor den Anderen.
  - Laufe bis das Rennen vorbei ist.

Programme bestehen aus Teilen:

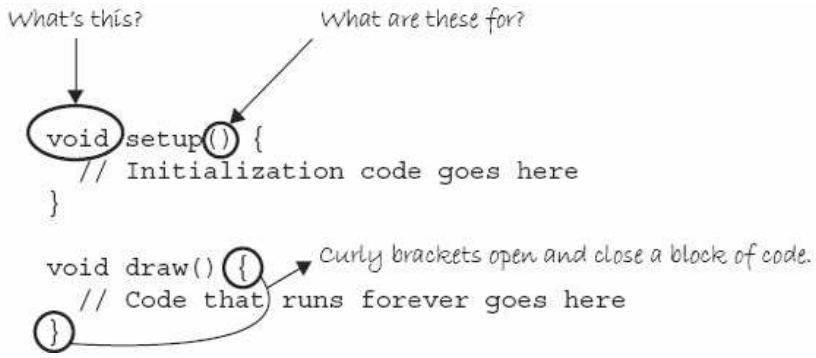
- sequentiell
- Bedingungen (Konditionale)
- Wiederholung (Iteration)

Erste Methoden

setup() und draw() sind Methoden

- Benannte Codeblöcke
- Sie schreiben den Code in den Blöcken
- Neue Symbole und neue Schlüsselwörter

```
void M-Name()
{
  Code
 }
```



Codeblöcke

- Programmteile werden in Blöcke separiert
- In Java, C, C++, PHP und vielen anderen Sprachen:
  - Blöcke werden von geschweifen Klammern umgeben.
- Denken Sie sich Blöcke als einen Grundriss der Unterschritte

- 1
  - 1a
  - 1b
- 2

```
{
    Ein Codeblock
    {
        Ein Block in einem Block
    }
}
```

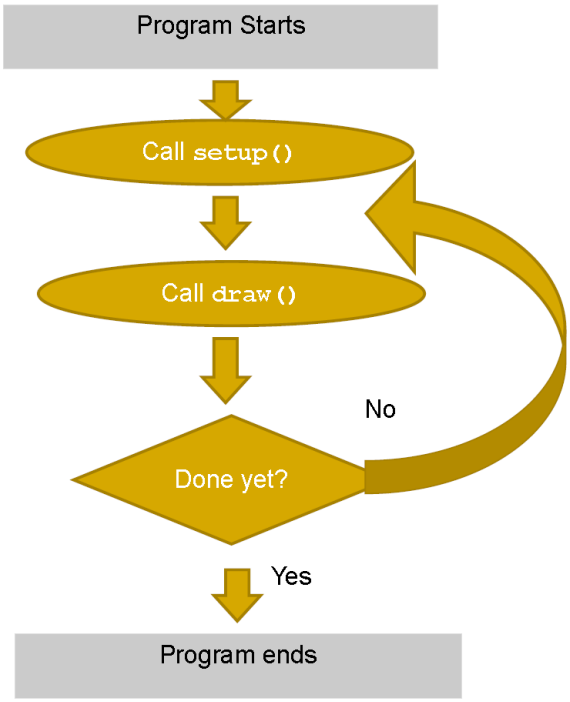
- Programmierer (und Editoren) ordnen Code in Blöcken an
- Die Anweisungen im Block werden eingerückt.

Die Methoden setup() und draw()

Beim Programmstart ruft Processing

- einmal setup()
- wiederholt draw()
- bis das Programm endet

```
void setup() {
    // Schritt 1a
    // Schritt 1b Einmal
    // Schritt 1c
}
void draw() {
    // Schritt 2a
    // Schritt 2b Wiederhole
}
```



Programmcode teilt sich in mindestens zwei Methoden auf

```
void setup() {
    // Groesse des Fensters
    size(200, 200);
}

void draw() {
    // Hintergrund setzen
    background(255);
    ellipseMode(CENTER);
    rectMode(CENTER);
    stroke(0);
    fill(150);
    rect(100, 100, 20, 100);
    ....
}
```

setup() läuft nur einmal.
size() sollte immer die erste Zeile in setup() sein.

draw() wird kontinuierlich wiederholt bis das Sketch-Fenster geschlossen wird.

Mausposition

Processing merkt sich die aktuelle Mausposition:

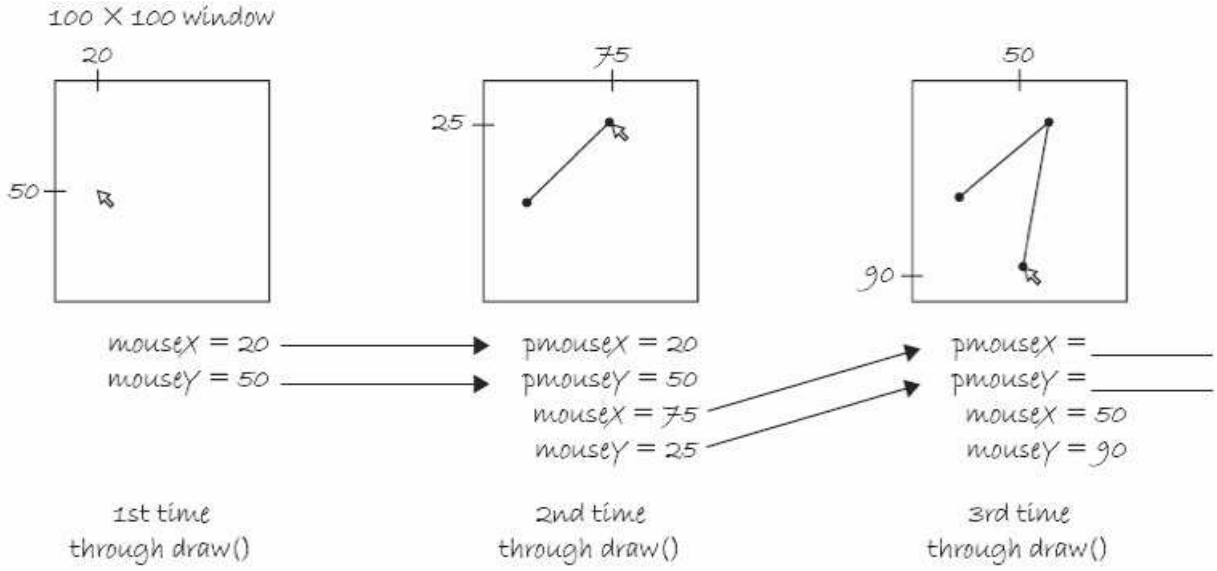
- mouseX: die aktuelle x-Koordinate der Maus
- mouseY: die aktuelle y-Koordinate der Maus
- beide sind nutzbare Schlüsselwörter.
- Die Werte werden immer der Mausposition angepasst.

```
void setup() {
    size(200, 200);
}
void draw() {
    background(255);
    stroke(0);
    fill(175);
    rectMode(CENTER);
    rect(mouseX, mouseY, 50, 50);
}
```

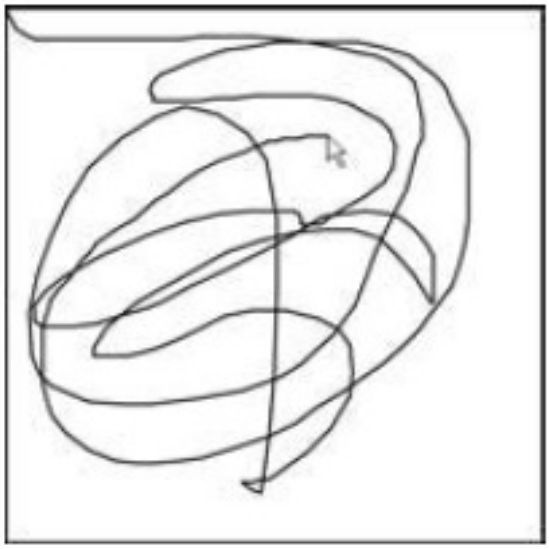
Processing merkt sich die Mausposition des vorhergehenden draw():

pmouseX: die vorherige x-Position der Maus

pmouseY: die vorherige y-Position der Maus



Scribble-Anwendung



```
void setup() {
    size(400, 400);
    background(255);
    smooth();
}
void draw() {
    stroke(0);
    line(pmouseX, pmouseY, mouseX, mouseY);
}
```

Hinweis: Keine Speicherung der Positionen

Mausklicks und Tastatureingaben

Zwei Methoden zur Behandlung von Maus- und Tastaturereignissen. Processung ruft diese Methoden auf, wann immer entsprechende Ereignisse eintreten:

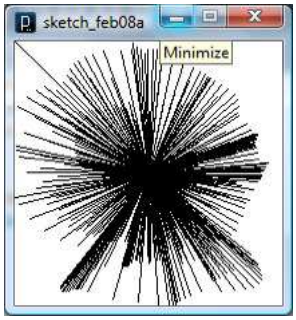
- mousePressed()
  - Eintragen eines Quadrats
  - Zeichnen **NICHT** in „draw“
- keyPressed()
  - Löschen des Bildschirms

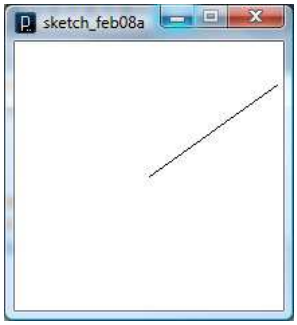
```
void setup() {
    size(200, 200);
    background(255);
}
void draw() {
}
void mousePressed() {
    stroke(0);
    fill(175);
    rectMode(CENTER);
    rect(mouseX, mouseY, 16, 16);
}
void keyPressed() {
    background(255);
}.
```

innerhalb von setup() können benutzt werden:

- size(200, 200);
- smooth();
- frameRate(30); // Voreinstellung: 60 Bilder pro Sekunde
- background(255); // Fensterinhalte löschen

Beispiel von Linien: line1 und line2






```
void setup() {
    size(200,200);
    background(255);
}
void draw() {
    line(mouseX, mouseY, 100,100);
}
// kein Neuzeichen
```

```
void setup() {
    size(200,200);
}
void draw() {
    background(255);
    line(mouseX, mouseY, 100,100);
}
// Neuzeichnen
```

Zusammenfassung:

Codeblöcken kann ein Name zugewiesen werden (Methoden)

```
methodenName {
      // Eine Folge von Anweisungen
}
```

- Processing läuft in einer Schleife.

  setup() und draw() Methoden werden implementiert.
- Processing merkt sich die aktuelle und vorhergehende Mausposition
- Processing kann mit Methoden gesteuert werden:
  - background(),
  - frameRate(),
  - size(),
  - smooth() // anti-aliased