

GUIs mit Processing

CS1016 Programmierung interaktiver Systeme

von Prof. Dr. Weigel



GUIs mit Processing

1. **Empfohlene Programmierumgebungen**
2. Was ist Processing?
3. Die Methoden `settings()`, `setup()` und `draw()`
4. Erste Schritte mit Processing

Java 21 LTS

Für IDEs wie IntelliJ und zum Nutzen der JShell muss das **Java Development Kit** installiert werden
Für den Kurs nutzen wir nutzen **Java 21 LTS**

Empfohlen: in IntelliJ IDEA als SDK auswählen

Installationspakete:

<https://www.oracle.com/java/technologies/downloads/>

Alternative Installation für Windows:

`winget install -e Oracle.JDK.21`

FAQ: Warum nicht Java Version X?

1. Java 21 hat Long-Term-Support bis 2026
2. Wir brauchen Features aus neueren Javas

The screenshot shows the Oracle Java Downloads page for JDK 21. The page has a dark blue header with the Oracle logo and navigation links. Below the header, there's a section for 'Java Downloads' with a Java logo. A navigation bar includes 'Java downloads', 'Tools and resources', and 'Java archive'. A section titled 'Looking for other Java downloads?' contains buttons for 'OpenJDK Early Access Builds' and 'JRE for Consumers'. The main content area is titled 'Java 21 and Java 17 available now' and states that JDK 21 is the latest long-term support release. A button 'Learn about Java SE Subscription' is present. Below this, there's a tabbed interface with 'JDK 21', 'JDK 17', 'GraalVM for JDK 21', and 'GraalVM for JDK 17'. The 'JDK 21' tab is selected. Underneath, it says 'JDK Development Kit 21 downloads'. A paragraph explains that JDK 21 binaries are free to use in production and free to redistribute under the Oracle No-Fee Terms and Conditions (NFTC). It also mentions that subsequent updates will be licensed under the Java SE OTN License. Below this, there's another tabbed interface with 'Linux', 'macOS', and 'Windows'. The 'Windows' tab is selected. A table lists the download options for Windows:

Product/file description	File size	Download
x64 Compressed Archive	180.99 MB	https://download.oracle.com/java/21/latest/jdk-21_windows-x64_bin.zip (sha256)
x64 Installer	160.12 MB	https://download.oracle.com/java/21/latest/jdk-21_windows-x64_bin.exe (sha256)
x64 MSI Installer	158.90 MB	https://download.oracle.com/java/21/latest/jdk-21_windows-x64_bin.msi (sha256)

[Bildquelle: <https://www.oracle.com/java/technologies/downloads/#jdk21-windows>]

IntelliJ IDEA Community Edition

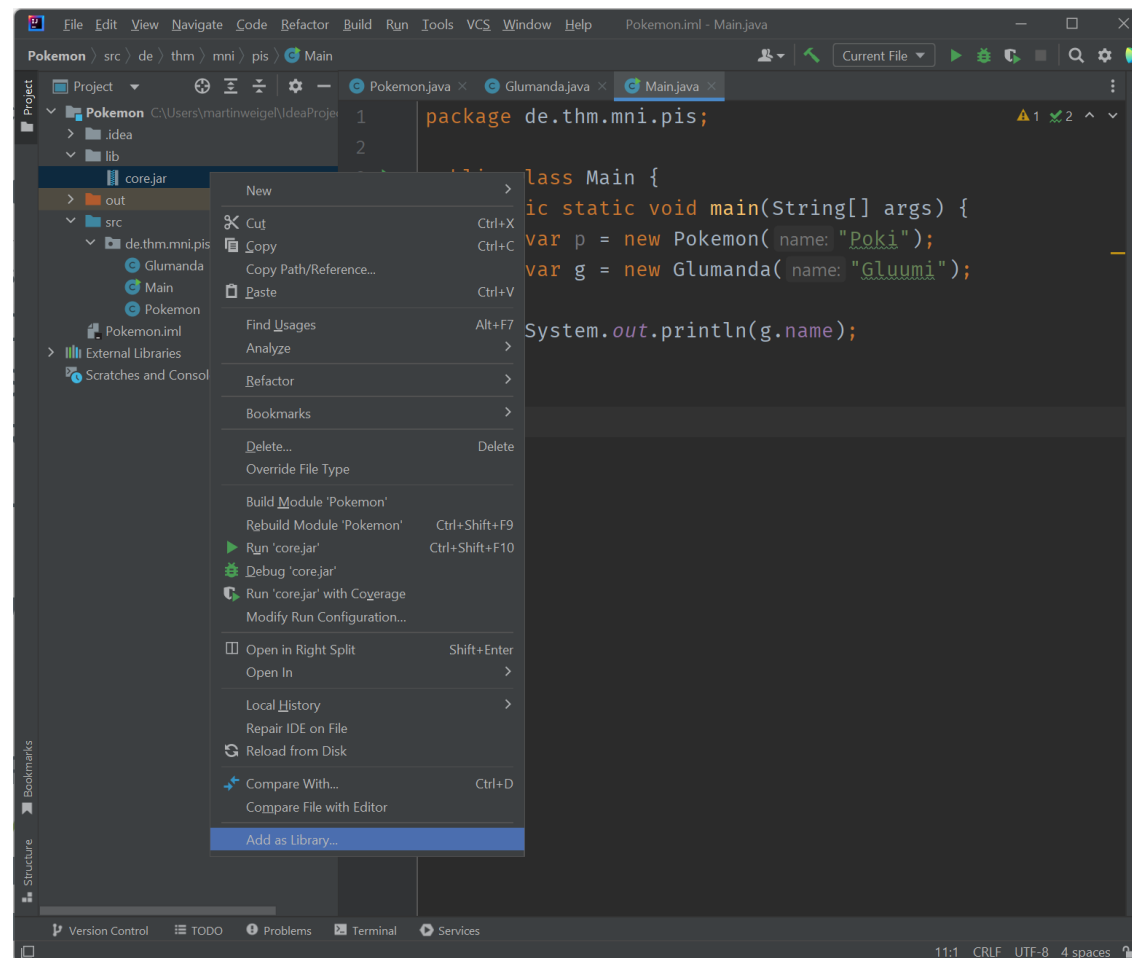
Kostenlose IDE für Java mit

- Code-Vervollständigung
- Schlaue Empfehlungen
- Refactoring
- ...

Download: <https://www.jetbrains.com/idea/>

Für Processing:

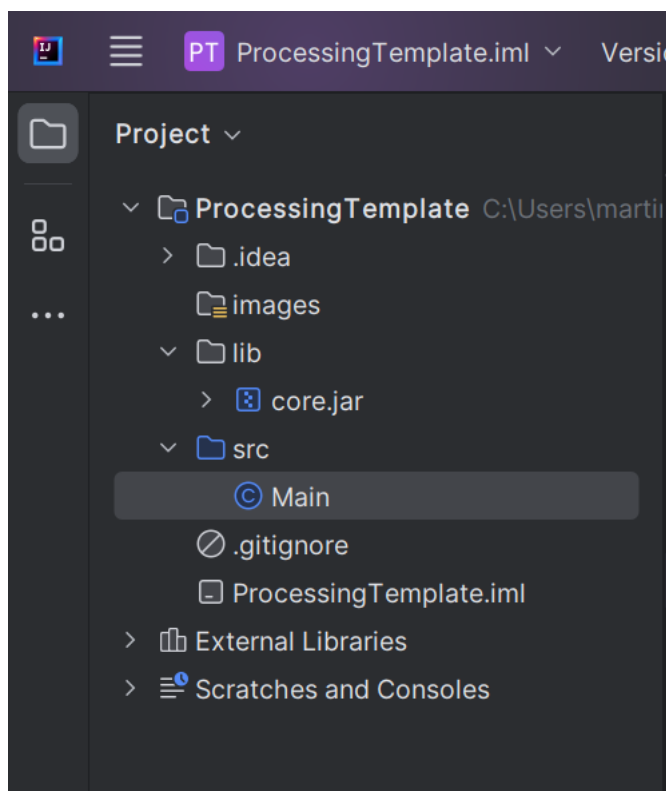
1. Processing herunterladen und `core/library/core.jar` extrahieren
2. Im Projekt im Ordner `lib/` ablegen
3. In IntelliJ IDEA mit Rechtsklick auf `lib/core.jar` und "Add as Library" auswählen



[Bildquelle: <https://www.jetbrains.com/idea/>]

IntelliJ Template für Processing

Wichtige Verzeichnisse

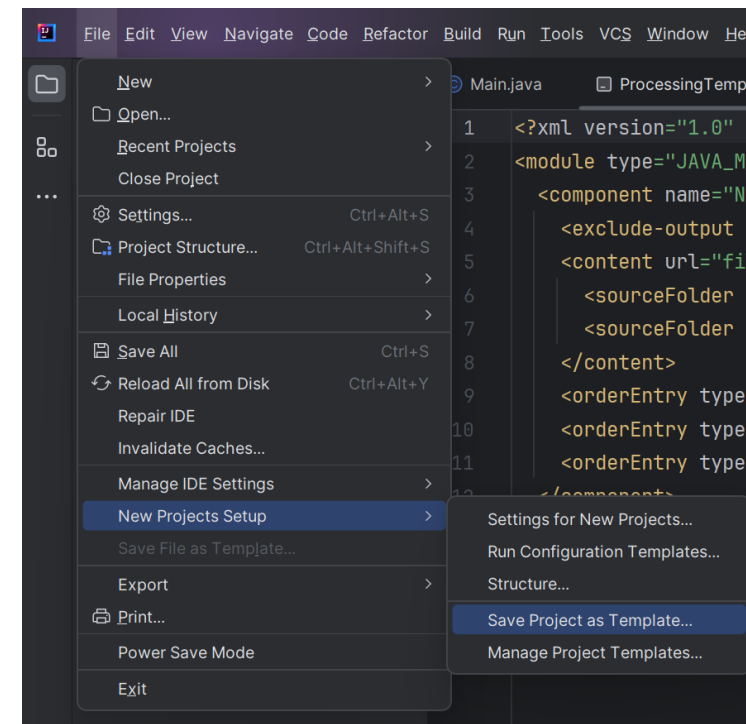


/images (Ressourcen Ordner)
Beinhaltet alle Bilder des Projektes. Diese werden automatisch beim Kompilieren in der /out Verzeichnis kopiert.

/lib (Bibliotheken)
Enthält eine aktuelle *core.jar* Bibliothek von Processing.

/src
Der Quellcode des Projektes

Tipp: Als Template speichern





GUIs mit Processing

1. Empfohlene Programmierumgebungen
2. Was ist Processing?
3. Die Methoden settings(), setup() und draw()
4. Einfache Formen zeichnen

Was ist Processing?

Processing ist ein Java-Framework zum zeichnen von **Animationen** und **interaktiven Anwendungen**

Genutzt von:

- Schüler*innen
- Student*innen
- Designer*innen
- Wissenschaftler*innen
- Bastler*innen

zum Programmieren-Lernen und Rapid Prototyping

Existiert seit 2001

Open Source und kostenlos

Vielfältige Zusatzbibliotheken:

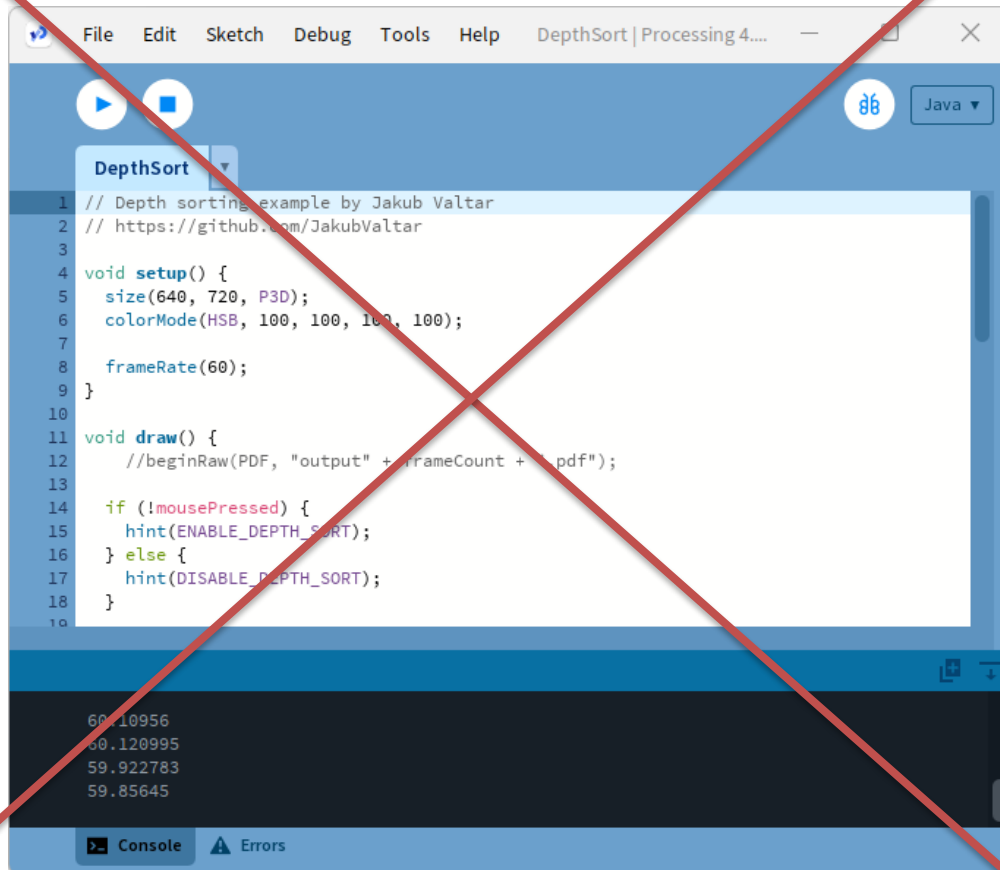
- Sound (minim)
- Netzwerkkommunikation (MQTT)
- Serielle Schnittstellen (Serial)
- Gestenerkennung (\$1)
- Angepasst an Sketch-Architektur
- Werden wir aber nicht verwenden

Inspirierte Frameworks für andere Programmiersprachen:



Processing

PDE (Editor / IDE)



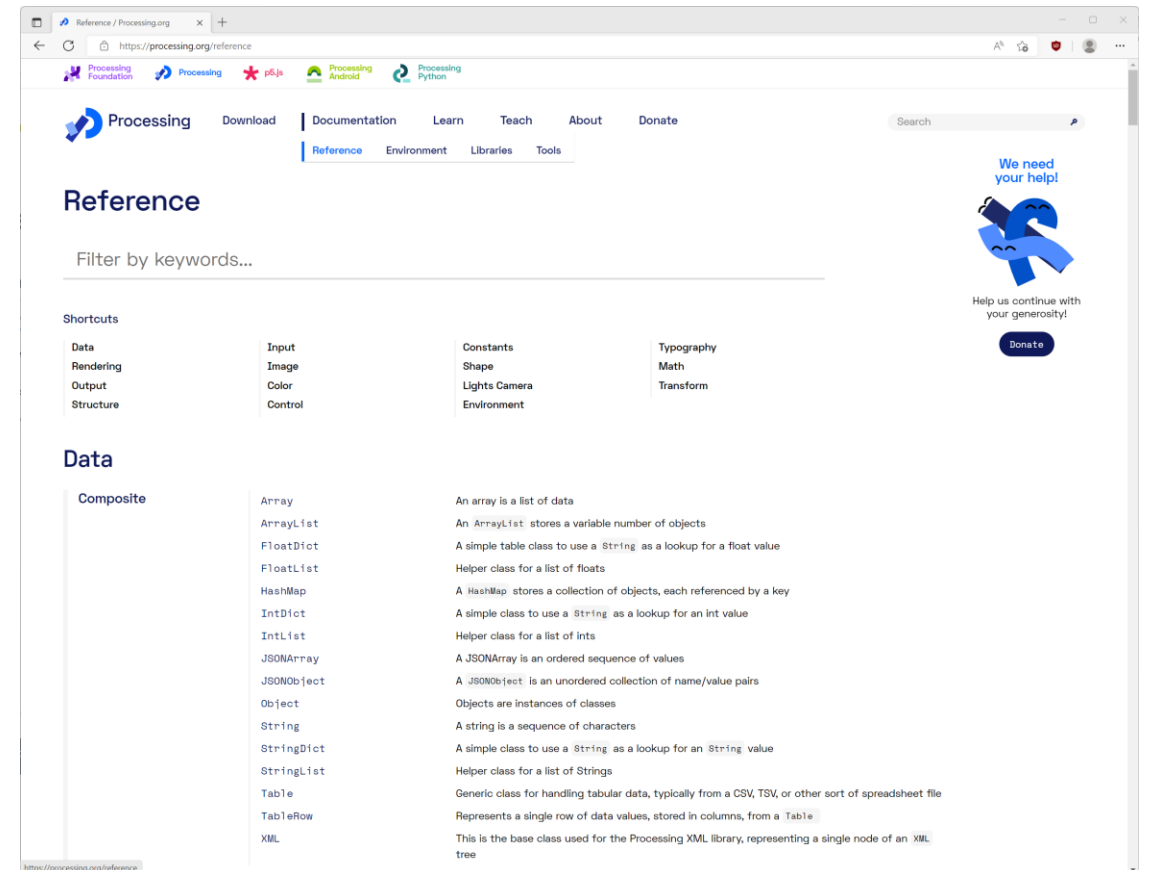
```

1 // Depth sorting example by Jakub Valtar
2 // https://github.com/JakubValtar
3
4 void setup() {
5   size(640, 720, P3D);
6   colorMode(HSB, 100, 100, 100, 100);
7
8   frameRate(60);
9 }
10
11 void draw() {
12   //beginRaw(PDF, "output" + frameCount + ".pdf");
13
14   if (!mousePressed) {
15     hint(ENABLE_DEPTH_SORT);
16   } else {
17     hint(DISABLE_DEPTH_SORT);
18   }
19 }

```

60.10956
60.120995
59.922783
59.85645

Framework (core.jar, uvm.)



Processing.org Reference

Filter by keywords...

Shortcuts

- Data
- Rendering
- Output
- Structure
- Input
- Image
- Color
- Control
- Constants
- Shape
- Lights Camera
- Environment
- Typography
- Math
- Transform

Data

Composite

- Array
- ArrayList
- FloatDict
- FloatList
- HashMap
- IntDict
- IntList
- JSONArray
- JSONObject
- Object
- String
- StringDict
- StringList
- Table
- TableRow
- XML

An array is a list of data

An `ArrayList` stores a variable number of objects

A simple table class to use a `String` as a lookup for a float value

Helper class for a list of floats

A `HashMap` stores a collection of objects, each referenced by a key

A simple class to use a `String` as a lookup for an int value

Helper class for a list of ints

A `JSONArray` is an ordered sequence of values

A `JSONObject` is an unordered collection of name/value pairs

Objects are instances of classes

A string is a sequence of characters

A simple class to use a `String` as a lookup for an `String` value

Helper class for a list of Strings

Generic class for handling tabular data, typically from a CSV, TSV, or other sort of spreadsheet file

Represents a single row of data values, stored in columns, from a `Table`

This is the base class used for the Processing XML library, representing a single node of an XML tree



GUIs mit Processing

1. Empfohlene Programmierumgebungen
2. Was ist Processing?
3. Die Methoden `settings()`, `setup()` und `draw()`
4. Erste Schritte mit Processing

Grundgerüst eines Processing Programmes

```
import processing.core.PApplet;

public class Main extends PApplet {
    public static void main(String[] args) {
        PApplet.main(Main.class);
    }

    public void settings() {
        System.out.println("1. Settings");
    }

    public void setup() {
        System.out.println("2. Setup");
    }

    public void draw() {
        System.out.println("3. Draw");
    }
}
```

Import

Die *core.jar* von Processing muss dem Java Projekt als Bibliothek hinzugefügt und importiert werden.

PApplet

Basis-Klasse für Processing Sketches
Stellt alle Processing-Funktionen und "Hooks" zur Verfügung

Muss in jeder Anwendung von *einer* Klasse überschrieben werden

Main

Startet den Processing Sketch

Grundgerüst eines Processing Programmes

```
import processing.core.PApplet;

public class Main extends PApplet {
    public static void main(String[] args) {
        PApplet.main(Main.class);
    }

    public void settings() {
        System.out.println("1. Settings");
    }

    public void setup() {
        System.out.println("2. Setup");
    }

    public void draw() {
        System.out.println("3. Draw");
    }
}
```

① Settings

Grundlegende Zeicheneigenschaften
wie Größe und Anti-Aliasing

② Setup

Anwendungscode der *einmalig* beim
Start ausgeführt werden soll

③ Draw

Anwendungscode der für *jedes*
gezeichnete *Frame* ausgeführt werden soll

Default: 60 Frames / s

Änderbar mit `frameRate(<fps>)`

Reale Framerate kann aber darunter liegen

Behind the Scenes*

```
public class PApplet {
    public static main(Class<?> mainClass) {
        var sketch = new mainClass();
        sketch.settings();
        sketch.setup();
        while(true) {
            if(<time to draw next frame>)
                sketch.draw();
        }
    }
    public void PApplet()      { [...] }
    public void settings()    { [...] }
    public void setup()       { [...] }
    public void draw()        { [...] }
    [...]
}
```

} PApplet ist die Basisklasse eines Sketches
Implementiert neuen Methoden wie println() und circle()

} Main Loop

} Leere Methoden die von uns überschrieben werden

*) Stark vereinfachtes Modell



GUIs mit Processing

1. Empfohlene Programmierumgebungen
2. Was ist Processing?
3. Die Methoden `settings()`, `setup()` und `draw()`
4. **Erste Schritte mit Processing**

Background

background(...)

Überschreibt den kompletten Fensterinhalt mit der gegebenen Farbe

→ Meist am Anfang von draw(), um den alten Zustand zu überschreiben

Häufige Aufrufe:

```
background(0); // Schwarz
```

```
background(255); // Weiß
```

```
background(<r>, <g>, <b>);
```

```
var thmColor = color(128, 186, 39);  
background(thmColor);
```

Beispielcode:

```
void settings() {  
    size(1000, 1000);  
}
```

```
void setup() {  
    frameRate(1);  
}
```

```
void draw() {  
    int r = (int)(Math.random() * 255);  
    int g = (int)(Math.random() * 255);  
    int b = (int)(Math.random() * 255);  
    background(r, g, b);  
}
```

Basic Shapes

Ein Kreis

`circle(<x>, <y>, <extent>);`

- `x` (float) x-coordinate of the ellipse
- `y` (float) y-coordinate of the ellipse
- `extent` (float) width and height of the ellipse by default

Beispielcode:

```
void settings() { size(1000, 1000); }
```

```
void setup() { frameRate(1); }
```

```
void draw() {  
    int size = (int)(Math.random() * width);  
    circle(width/2, height/2, size);  
}
```

Andere Shapes

<code>arc()</code>	Draws an arc in the display window
<code>circle()</code>	Draws a circle to the screen
<code>ellipse()</code>	Draws an ellipse (oval) in the display window
<code>line()</code>	Draws a line (a direct path between two points) to the screen
<code>point()</code>	Draws a point, a coordinate in space at the dimension of one pixel
<code>quad()</code>	A quad is a quadrilateral, a four sided polygon
<code>rect()</code>	Draws a rectangle to the screen
<code>square()</code>	Draws a square to the screen
<code>triangle()</code>	A triangle is a plane created by connecting three points

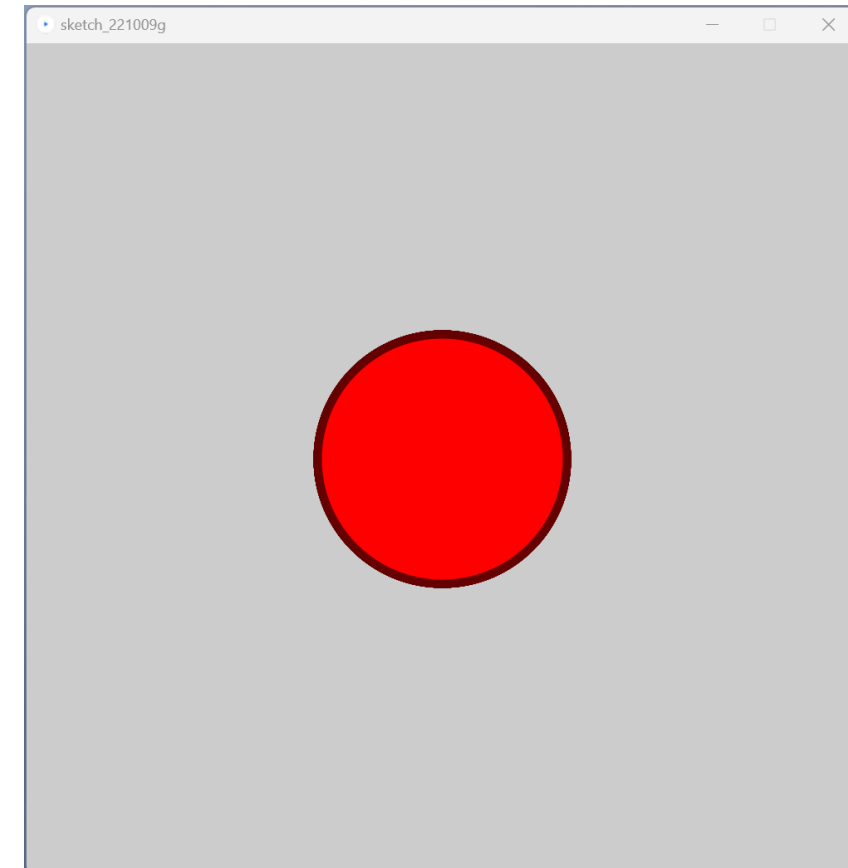
Füllung und Kontur

Beispielcode

```
void settings() { size(1000, 1000); }
```

```
void draw() {  
  strokeWeight(10);           // Konturdicke  
  stroke(color(100, 0, 0));    // Konturfarbe: Dunkelrot  
  fill(color(255, 0, 0));      // Füllfarbe: Helles Rot  
  circle(width/2, height/2, 300);  
  
  // Füllung und Kontur bleiben dauerhaft erhalten  
}
```

Sketch:



Bilder einbetten

`PImage eevee;`

`int xPos = 0;`

`void settings() {`

`size(1000, 200);`

// loadImage(<url / pfad>) darf erst in oder nach settings() benutzt werden.

`eevee = loadImage("https://assets.pokemon.com/assets/cms2/img/pokedex/full/133.png");`

`}`

`void draw() {`

`background(255);`

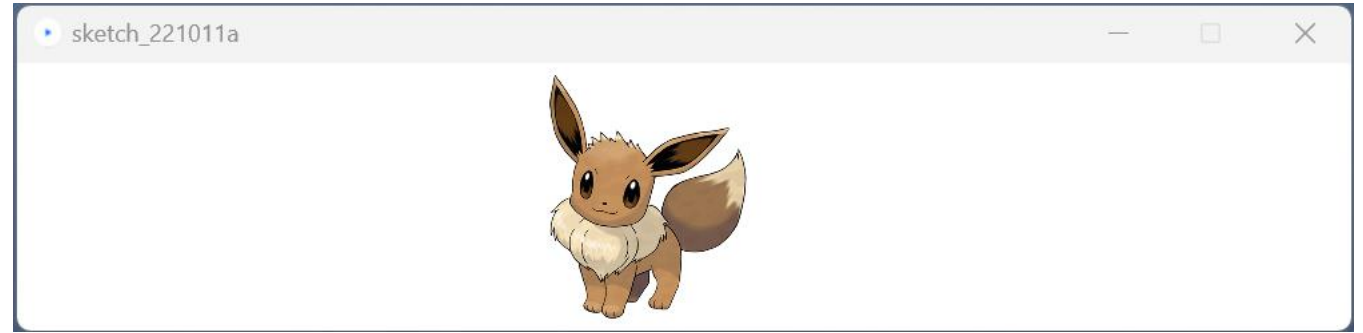
// Parameter: image(<PImage>, xPos, yPos, width, height)

`image(eevee, xPos, 0, 200, 200);`

// Falls am linken Rand zurücksetzen, sonst 3px nach links

`xPos = (xPos <= 0) ? width : xPos - 3;`

`}`



Entspricht:

`if(xPos <= 0)`

`xPos = width;`

`else`

`xPos = xPos - 3;`

More Pokemon?!

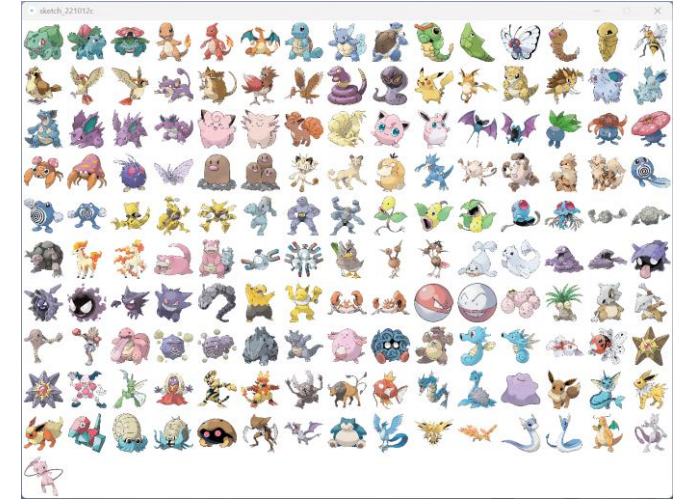
```
PImage[] gen1;
```

```
int size = 100;
```

```
void settings() {
    size(1500, 1100);
    gen1 = new PImage[151];
    for(int i=0; i<gen1.length; i++) {
        String p =
        String.format("https://assets.pokemon.com/assets/cms2/im
g/pokedex/full/%03d.png", i+1);
        gen1[i] = loadImage(p);
    }
}
```

```
void draw() {
    background(255);

    for(int i=0; i<gen1.length; i++) {
        int xPos = (int)(i % 15) * size;
        int yPos = (int)(i / 15) * size;
        image(gen1[i], xPos, yPos, size, size);
    }
}
```



Typografie

```
text("Hello World", xPos, yPos);
```

stellt einen Text im Fenster da

```
textSize(36);
```

setzt Schriftgröße für nachfolgende Texte
Wert in Pixeln

Viele weitere Funktionen, z.B.

- **textFont()** setzt eine Schriftart
- **textAlign(CENTER)** zentriert den Text
- ...

```
void settings() { size(1000, 1000); }
```

```
void draw() {
```

```
    background(0);
```

```
    fill(255);
```

```
    rect(25, 25, 500, 500);
```

```
    fill(color(255, 0, 0));
```

```
    textSize(36);
```

```
    text("Lorem ipsum dolor sit amet, consectetur  
adipiscing elit, sed do eiusmod tempor  
incididunt ut labore et dolore magna aliqua. Ut  
enim ad minim veniam, quis nostrud exercitation  
ullamco laboris nisi ut aliquip ex ea commodo  
consequat. [...]",
```

```
        50, 50, 450, 450);
```

```
}
```

Interaktion: Tastatureingabe

Keyboard

<code>key</code>	The system variable that always contains the value of the most recent key on the keyboard that was used (either pressed or released)
<code>keyCode</code>	Used to detect special keys such as the UP, DOWN, LEFT, RIGHT arrow keys and ALT, CONTROL, SHIFT
<code>keyPressed</code>	The boolean system variable that is <code>true</code> if any key is pressed and <code>false</code> if no keys are pressed
<code>keyPressed()</code>	Called once every time a key is pressed
<code>keyReleased()</code>	Called once every time a key is released
<code>keyTyped()</code>	Called once every time a key is pressed, but action keys such as Ctrl, Shift, and Alt are ignored

Instanzvariablen von PApplet
Zugreifbar in `draw()`

Event-Listener Methoden
Können überschrieben werden

Auf Tastaturdruck die Hintergrundfarbe ändern...

```
int r, g, b;
```

```
void settings() {  
    size(1000, 1000);  
}
```

```
void draw() {  
    background(r, g, b);  
}
```

```
void keyPressed() {  
    r = (int)(Math.random() * 255.0);  
    g = (int)(Math.random() * 255.0);  
    b = (int)(Math.random() * 255.0);  
}
```

Interaktion: Mauseingabe

Mouse

<code>mouseButton</code>	Shows which mouse button is pressed
<code>mouseClicked()</code>	Called once after a mouse button has been pressed and then released
<code>mouseDragged()</code>	Called once every time the mouse moves and a mouse button is pressed
<code>mouseMoved()</code>	Called every time the mouse moves and a mouse button is not pressed
<code>mousePressed</code>	Variable storing if a mouse button is pressed
<code>mousePressed()</code>	Called once after every time a mouse button is pressed
<code>mouseReleased()</code>	Called every time a mouse button is released
<code>mouseWheel()</code>	The code within the <code>mouseWheel()</code> event function is run when the mouse wheel is moved
<code>mouseX</code>	The system variable that always contains the current horizontal coordinate of the mouse
<code>mouseY</code>	The system variable that always contains the current vertical coordinate of the mouse
<code>pmouseX</code>	The system variable that always contains the horizontal position of the mouse in the frame previous to the current frame
<code>pmouseY</code>	The system variable that always contains the vertical position of the mouse in the frame previous to the current frame

Event-Listener Methoden
 Können überschrieben werden

Instanzvariablen in PApplet
 Zugreifbar in `draw()`

Positionsabhängige Hintergrundfarbe

```
int r, g, b;
```

```
void settings() {    size(1000, 1000); }
```

```
void draw() {  
    background(r, g, b);  
    textSize(30);  
    text(String.format("Farbe: (%d, %d, %d)", r, g, b), 50, 50);  
}
```

```
void mouseMoved() {  
    r = (int)(255.0 * mouseX / width);  
    g = (int)(255.0 * mouseY / height);  
    b = 0;  
}
```

Hinweis:

String.format() erstellt einen neuen String mit den gegebenen Werten:

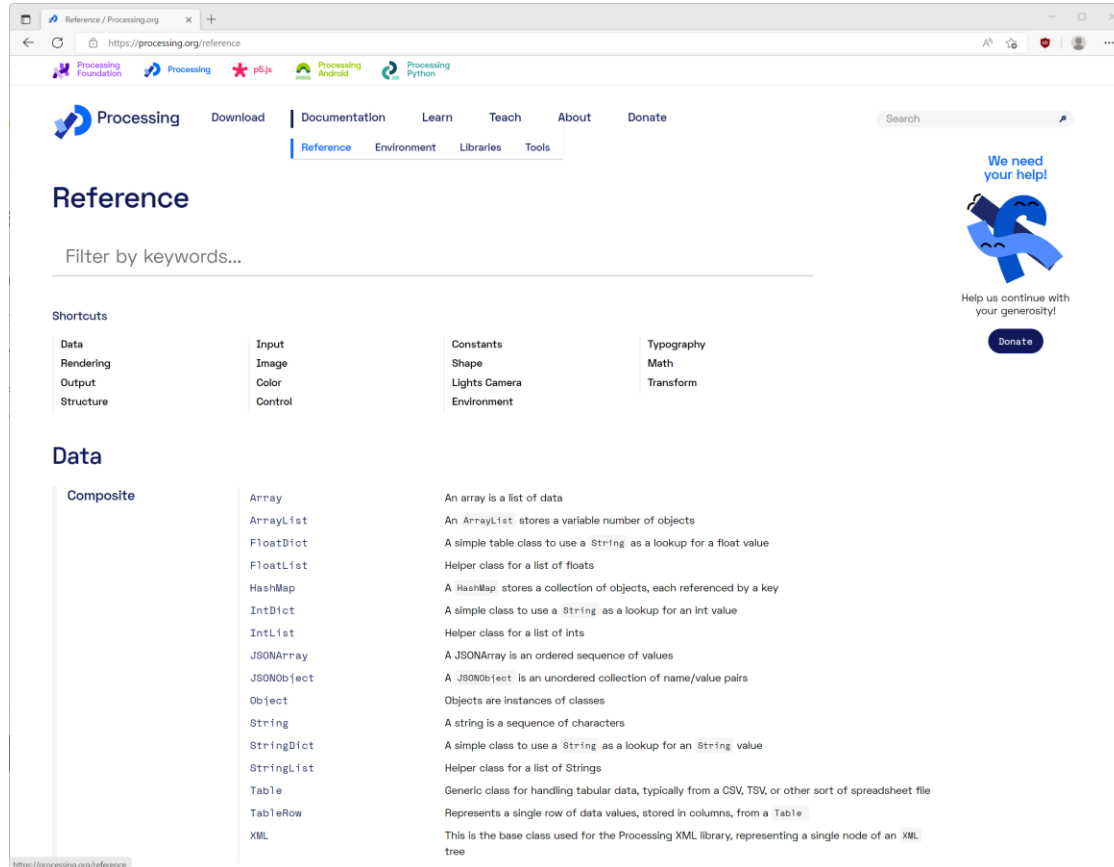
- Das erste %d wird ersetzt mit dem **Dezimal**-Wert von r
- Das zweite %d wird ersetzt mit dem **Dezimal**-Wert von g
- Das dritte %d wird ersetzt mit dem **Dezimal**-Wert von b

Ergebnis ist identisch zu:

"Farbe: (" + r + ", " + g + ", " + b + ")"

Processing

API Referenz



Reference

Filter by keywords...

Shortcuts

- Data
- Rendering
- Output
- Structure

Input

- Image
- Color
- Control

Constants

- Shape
- Lights Camera
- Environment

Typography

- Math
- Transform

Data

Composite

- Array
- ArrayList
- FloatDict
- FloatList
- HashMap
- IntDict
- IntList
- JSONArray
- JSONObject
- Object
- String
- StringDict
- StringList
- Table
- TableRow
- XML

Array

An array is a list of data

An `ArrayList` stores a variable number of objects

A simple table class to use a `String` as a lookup for a float value

Helper class for a list of floats

A `HashMap` stores a collection of objects, each referenced by a key

A simple class to use a `String` as a lookup for an int value

Helper class for a list of ints

A `JSONArray` is an ordered sequence of values

A `JSONObject` is an unordered collection of name/value pairs

Objects are instances of classes

A string is a sequence of characters

A simple class to use a `String` as a lookup for an `String` value

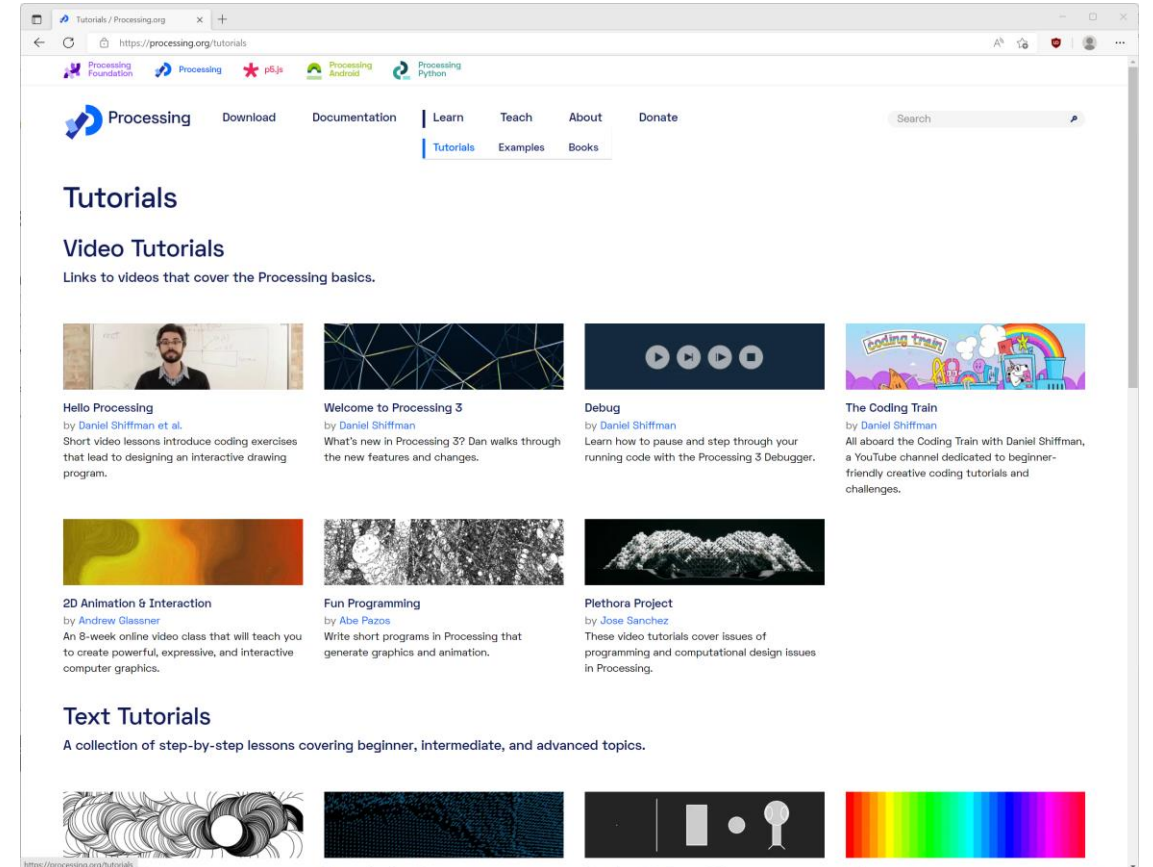
Helper class for a list of Strings

Generic class for handling tabular data, typically from a CSV, TSV, or other sort of spreadsheet file

Represents a single row of data values, stored in columns, from a `Table`

This is the base class used for the Processing XML library, representing a single node of an XML tree

Tutorials



Tutorials

Video Tutorials

Links to videos that cover the Processing basics.

- Hello Processing**
by Daniel Shiffman et al.
Short video lessons introduce coding exercises that lead to designing an interactive drawing program.
- Welcome to Processing 3**
by Daniel Shiffman
What's new in Processing 3? Dan walks through the new features and changes.
- Debug**
by Daniel Shiffman
Learn how to pause and step through your running code with the Processing 3 Debugger.
- The Coding Train**
by Daniel Shiffman
All aboard the Coding Train with Daniel Shiffman, a YouTube channel dedicated to beginner-friendly creative coding tutorials and challenges.

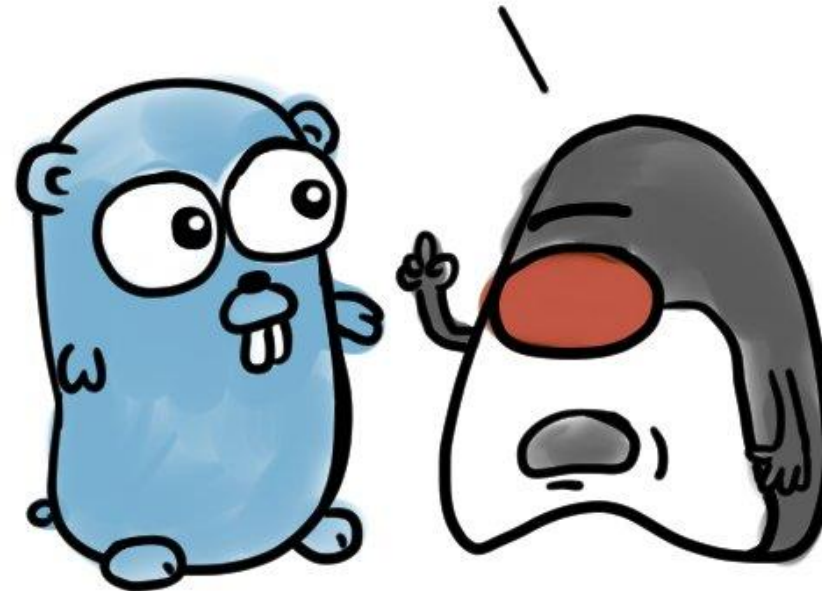
- 2D Animation & Interaction**
by Andrew Glassner
An 8-week online video class that will teach you to create powerful, expressive, and interactive computer graphics.
- Fun Programming**
by Abe Pazos
Write short programs in Processing that generate graphics and animation.
- Plethora Project**
by Jose Sanchez
These video tutorials cover issues of programming and computational design issues in Processing.

Text Tutorials

A collection of step-by-step lessons covering beginner, intermediate, and advanced topics.

Fragen?

I had my time, believe me.
Sooner or later they will
call you slow, verbose,
old fashioned...



Daniel Stori {turnoff.us}

[Bildquelle: https://twitter.com/turnoff_us/status/856155022375170050/photo/1]

Für die Übung bitte IntelliJ IDEA installieren + Processing Template laden

Download:

<https://www.jetbrains.com/idea/>

Processing Template aus dem Moodle laden und entpacken

Anm.: Manche Anti-Viren-Scanner melden die *core.jar* aus neue Processing-Versionen als potentiellen Virus

→ Falscher Alarm
(s. Online-Diskussionen)

```

1 import processing.core.PApplet;
2
3 /**
4  * Just a basic Processing template to check if your IDE is configured correctly.
5  */
6 public class Main extends PApplet {
7     public static void main(String[] args) { PApplet.main(Main.class); }
8
9     no usages
10    public Main() {}
11
12    @Override
13    public void settings() {
14        setSize( width: 600, height: 400);
15        pixelDensity(2);
16    }
17
18    @Override
19    public void setup() {}
20
21
22
23

```

Let's program

Evolis

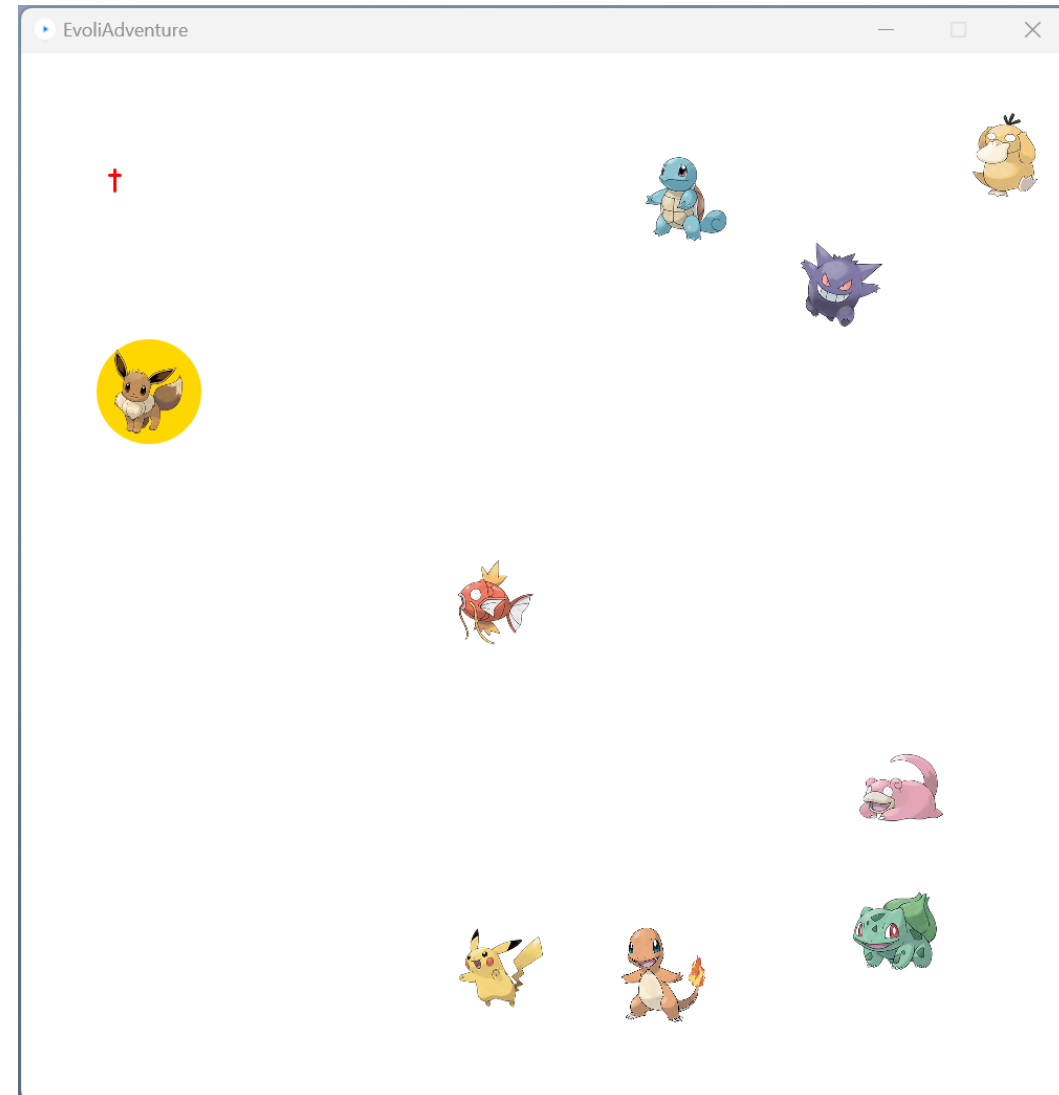
Adventure

Das wahrscheinlich
einfachste "Pokemon"
Spiel der Welt



[Bildquelle: <https://pokemonletsgo.pokemon.com/de-de/story/>]

Live Demo



Gotta Catch 'Em All!

Schritte:

1. Überlegen was wir brauchen

2. Programmieren

- “Pair”-Programming: Ich programmiere mit Ihnen
- Versuchen Sie gerne parallel mitzuprogrammieren
- Bilder finden Sie unter <https://www.pokemon.com/us/pokedex/>
- Allein oder mit Ihren Sitznachbarn

→ Aber bitte konzentriert bleiben und andere nicht stören 😊

3. Spielen