PROCESSING

WIR ZIEHEN UM!

Created by Michael Kirsch & Beat Rossmy

INHALT

RÜCKBLICK

JAVA?

THEORIE

SATZGENERATOR

- Wie können wir gewohnte Strukturen auf Java übertragen?
- Wo deklarieren, initialisieren und manipulieren wir?

```
// deklariere:
// Subjekt, Praedikat, Objekt

void setup () {
  // Initialisiere:
  // Subjekt, Präedikat, Objekt
}

void draw () {
  // wähle zufälliges Subjekt
  // wähle zufälliges Prädikat
  // wähle zufälliges Objekt

  // gebe Kombination aus
}
```

BOUNCING BALL

- Wie können wir darstellen?
- Wie können wir Programmhandlungen stetig wiedeholen?

```
// deklariere:
// xPosition, yPosition
// xRichtung, yRichtung
// width, height

void setup () {
  // Initialisiere:
  // Variablen
}

void draw () {
  // verändere Position
  // behandle Grenzen
  // zeichne Darstellung
}
```

PIXELTAUSCH

- Wo und wie definieren wir Funktionen?
- Wie greifen wir richtig auf Werte zu?

```
// deklariere:
// Pixel Array

void setup () {
  // Initialisiere:
  // Pixel Array
}

void draw () {
  // bestimme zufällige Positionen
  // tausche Pixel ->
  // zeichne Darstellung
}

void tauschePixel (int a, int b) {
  // sichere Wert an Stelle a
  // schreibe in Stelle a Wert b
  // schreibe gesicherten Wert in b
}
```

WO SIND SETUP UND DRAW?

- Unsere übliche Vorgehensweise muss nun der Struktur von Java angepasst werden.
- Anstelle der immerwieder ausgeführten **`draw`** Methode können wir uns Schleifen bedienen um Handlungsabfolgen auszuführen.

```
public class Test {

  public static void main (String []
    int a;

    // void setup () {
    a = 1;
    // }

    // void draw () {
    for (int n=0; n<10000; n++) {
      a++;
      System.out.println(a);
    }
    //
  }

}
```

WIE ZEICHNEN WIR ETWAS?

- Aus der letzen Sitzung ist klar, zeichnen wie in Processing ist ohne weiteres nicht ganz möglich.
- Aber wir können fürs erste auch mit Workarounds arbeiten.
- Konsolenausgaben können wie ein Display funktionieren.

```
public class Test {

  public static void main (String []
    // setup ...

    // draw
    for (int n=0; n<10000; n++) {

      for (int y=0; y<10; y++) {
        // einen Pixel pro Spalte
        for (int x=0; x<10; x++) {
          System.out.print(". ");
        }
        // nächste Zeile
        System.out.println();
      }
      // Abstand zum nächsten Bild
      System.out.println("\n");
    }
  }
}
```

ALLES IN EINER DATEI?

- Alle KLassen und Interfaces, etc in einer Datei zu definieren, ist in Java nicht möglich.
- Jede Klasse, etc erhält ihr eigenes File.
- Dieser Umstand zwingt uns zu mehr Ordnung, vergleichbar zu der Strukturierung durch Tabs in Processing.

```
void setup () {...}

void draw () {...}

public class A {...}

public class B {...}

public class C {...}
```

FUNKTIONEN DEFINIEREN?

- In Processing können Funktionen definiert werden, die einfach durch ihren Namen aufgerufen werden können.
- In Java beschwert sich der Compiler, dass dies nur mit **`static`** ausgezeichneten Funktionen funktioniert.
- Warum ist das so?

```
public class Test {

  public static void main (String []
    testFunktion();
  }

  public static void testFunktion ()
    System.out.println(
      "Warum static?"
    );
  }
}
```

VERKNÜPFUNG

BIBLIOTHEKEN

- Um nun Sound einbinden zu können, müssen wir eine Bibliothek mit dem Namen Sound importieren.
- Dazu laden wir diese herunter, und importieren sie anschließend über das Menü Sketch.
- Nun können wir die Bibliothek verwenden.

```
import processing.sound.*;

Animation[] animations;

void setup () {
  ...
}

void draw () {
  ...
}
```

SOUNDFILE

- Nun wird der Klasse Animation im Konstruktor ein **`SoundFile`** übergeben, das in einem Feld abgelegt und somit abrufbar ist.
- Dieser Sound wird nun in **`triggerOn`** gestartet.

```
class Animation implements Plotable,
  int animationCounter;
  char triggerKey;
  SoundFile sound;

  public Animation (char c,
  SoundFile s) {
    triggerKey = c;
    animationCounter = 0;
    sound = s;
  }

  void triggerOn (char c) {...}
  void triggerOff (char c) {...}
  void plot () {}
}
```

SOUNDFILE

- Der Methodenaufruf **`play()`** auf dem **`SoundFile`** startet die Wiedergabe des Sounds.
- Wo erzeugen wir das Objekt, das im Konstruktor übergeben werden soll?

```
class Animation implements Plotable,
  int animationCounter;
  char triggerKey;
  SoundFile sound;

  public Animation (char c,
  SoundFile s) {...}

  void triggerOn (char c) {
    if (triggerKey == c) {
      animationCounter = 1;
      sound.play();
    }
  }

  void triggerOff (char c) {...}

  void plot () {}
}
```

SOUNDFILE

- Wir erzeugen das **`SoundFile`** Objekt direkt im Konstruktoraufruf.
- Als zweiten Paramter übergeben wir den Dateipfad.
- Als ersten Parameter **`this`**. Dies bezieht sich in diesem Fall auf unseren Sketch.

```
Animation[] animations;

void setup () {
  animations = new Animation[]{
    new Animation('1',new SoundFile(
  }

}

void draw () {...}

void keyPressed () {...}

void keyReleased () {...}
```

DATA

- Im Sketchordner können wir einen Ordner mit dem Namen **`data`** erezugen.
- Dateien in diesem Ordner können direkt mit dem Namen referenziert werden.

```
Animation[] animations;

void setup () {
  animations = new Animation[]{
    new Animation('1',new SoundFile(
  }

}

void draw () {...}

void keyPressed () {...}

void keyReleased () {...}
```

QUELLEN