THM TECHNISCHE HOCHSCHULE MITTELHESSEN | CAMPUS GIESSEN | MNI Mathematik, Naturwissenschaften und Informatik

Brückenkurs Programmieren

Tag 3: Funktionen und Events

Jakob Czekansky, M.Sc.

Technische Hochschule Mittelhessen

15. März 2023

Rückblick - Tag 1: Variablen und Verzweigungen

Variablen

```
int i = 10;
float f = 3.5;
boolean b = true;
char c = 'x';
String blabla = "Blabla";
```

If-Abfrage

```
if (<Bedingung>) {
  <Anweisungsblock1>
} else {
  <Anweisungsblock2>
}
```

Logische Operatoren

> >= == !=

&& || !

Arithmetische Operatoren

\+ - * / %

Rückblick - Tag 2: Schleifen

Beispiel: For-Schleife

```
for (int i=0;i<end;i++) {
  ellipse(i*10,20,10,10);
}
```

Syntax: While-Schleife

```
while(<Bedingung>) {
  <Anweisungsblock>
}
```

Rückblick - Tag 2: Animationen und Arrays

setup und draw

```
void setup() {
  //einmal am Anfang
}
void draw() {
  //alle 15 ms
}
```

Mausposition

```
void draw() {
  ellipse(mouseX,20,10,10);
}
```

Arrays

```
int[] numbers = new int[10];
numbers[0] = 20;
numbers[3+2] = numbers[0];
int val = numbers[5];
```

Inhalt

Inhalt

Aufgabe

Zeichne ein Rechteck, einen Kreis und ein Dreieck nebeneinander. Bestimme die Füllfarben dabei zufällig, aber immer so, dass die Helligkeit von links nach rechts abnimmt.

Bunte Formen

Funktionen: Bunte Formen - Lösung 1

Bunte Formen

Lösung in Processing

```
size(600,200);
background(255);
```

Funktionen: Bunte Formen - Lösung 1

Bunte Formen

Lösung in Processing

```
size(600,200);
background(255);
fill(150+random(100),150+random(100),150+random(100));
rect(100,50,100,100);
```

Funktionen: Bunte Formen - Lösung 1

Bunte Formen

Lösung in Processing

```
size(600,200);
background(255);
fill(150+random(100),150+random(100),150+random(100));
rect(100,50,100,100);
fill(60+random(90),60+random(90),60+random(90));
ellipse(300,100,100,100);
```

Funktionen: Bunte Formen - Lösung 1

Bunte Formen

Lösung in Processing

```
size(600,200);
background(255);
fill(150+random(100),150+random(100),150+random(100));
rect(100,50,100,100);
fill(60+random(90),60+random(90),60+random(90));
ellipse(300,100,100,100);
fill(random(60),random(60),random(60));
triangle(400,150,450,50,500,150);
```

Funktionen: Definition

Problem: Man muss dreimal den (fast) gleichen Code schreiben.

Lösung: Neuer Befehl `randomColor(x,y)` zum Erzeugen einer zufälligen Farbe mit Helligkeitswert zwischen x und y.

Definition

Durch das Definieren einer Funktion gibt man einer oft benötigten Teillösung einen Namen, damit man sie für verschiedene Eingabewerte (Argumente) anwenden kann und den richtigen Rückgabewert bekommt.

Funktionen: Kaffeemaschine als Funktion

Funktionen: Syntax

Syntax

```
<Rückgabetyp> <Name> (<Parameter1>,<Parameter2>,..) {
  <Anweisungsblock>
  return <Rückgabewert>;
}
```

Beispiel

```
color randomColor(int minBr, int maxBr) {
  float span = maxBr-minBr;
  float r = minBr + random(span);
  float g = minBr + random(span);
  float b = minBr + random(span);
  return color(r,g,b);
}
```

Funktionen: Bunte Formen - Lösung 2

Bunte Formen

Lösung mit Funktion `randomColor`

```
void setup() {
  fill(randomColor(150,250));
  rect(100,50,100,100);
  fill(randomColor(60,150));
  ellipse(300,100,100,100);
  fill(randomColor(0,60));
  triangle(400,150,450,50,500,150);
}
```

Aufgabe: Meine erste Funktion

Schreibe deine erste Processing-Funktion:

- Name: `mouseBetween`
- Rückabetyp: `boolean`
- Parameter: Zwei x-Koordinaten `x1` und `x2` als Fließkommazahlen
- Aufgabe: Gibt an, ob sich die Maus zwischen `x1` und `x2` befindet.

Nutze diese Funktion, um den Hintergrund je nach Mausposition einzufärben (rot/grün/blau im linken/mittleren/rechten Fensterdrittel).

Erinnerung: Syntax einer Funktionsdefinition

```
<Rückgabetyp> <Name> (<Parameter1>,<Parameter2>,..) {
  <Anweisungsblock>
  return <Rückgabewert>;
}
```

Funktionen: Funktionen als Strukturierungsmittel

Auch wenn ein Stück Code nur einmal aufgerufen wird, kann es Sinn machen dieses in eine eigene Funktion auszulagern, um den Code lesbarer zu machen und die Fehlersuche zu erleichtern.

```
void setup() {
  int[] ar = {1,5,2,4,10,2,5,12,5};

  for(int i = 0; i < ar.length; i++) {
    int mi = findMinIdx(ar,i);
    swap(ar,mi,i);
  }
}
int findMinIdx(int[] haystack, int start) { ... }
void swap(int[] ar, int idx1, int idx2) { ... }
```

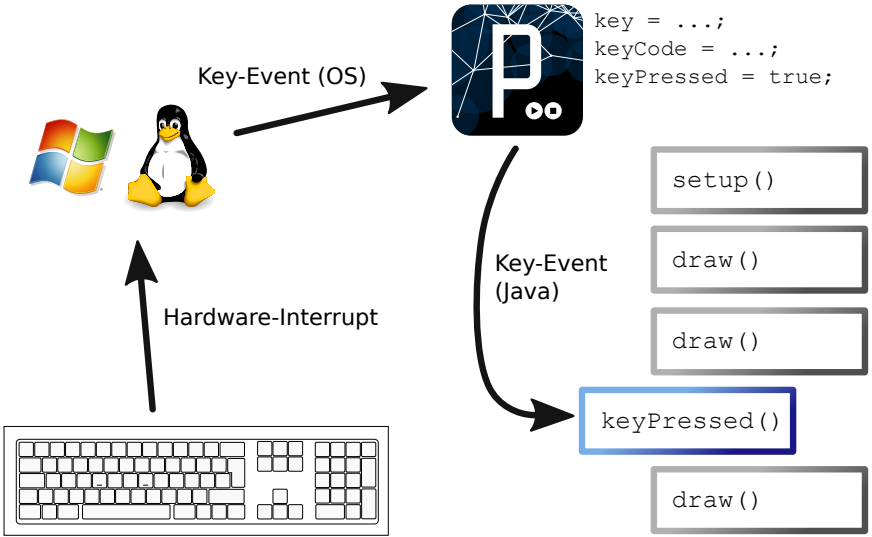
Inhalt

Events: Besondere Funktionen in Processing

Processing-code kann in Funktionen stehen, die zu unterschiedlichen Zeitpunkten aufgerufen werden:

- `setup()` ✓
- `draw()` ✓
- `mousePressed()` ?
- `keyPressed()` ?
- `keyReleased()`
- ...

Events: Wie entsteht ein Event?



Wikimedia-Bilder: Tux von Ewing, Budig und Gerwinsik, Processing von Woodmath (CC3), Tastatur von Cy21 (CC3)

Events: keyPressed

Beispiel

```
void keyPressed() {
  if(key == 'w') y--;
  else if (keyCode == CONTROL) y = 0;
}
```

Bausteine

Besondere Variablen:

- `key` (`char`) für druckbare Zeichen (`'a'`,`'b'`,`'7'`,`' '`, ...)
- `keyCode` (`int`) für Steuerzeichen (`CONTROL`, `ALT`, `LEFT`, ...)

```
void keyPressed() {
  //wird bei jedem Tastendruck ausgeführt
}
```

Events: mousePressed

Beispiel

```
void mousePressed() {
  if(mouseButton == LEFT && mouseX > 100) running = true;
}
```

Bausteine

Besondere Variablen:

- `mouseButton` (`int`) entweder `LEFT`, `RIGHT` oder `CENTER`
- `mouseX`, `mouseY` (`int`) als Koordinaten im Fenster

```
void mousePressed() {
  //wird bei jedem Mausdruck ausgeführt
}
```

Aufgabe: Disco Klicker

Verändere jedes mal, wenn die Maus geklickt wird die Hintergrundfarbe zu einer neuen Zufallsfarbe.

Vorgabe

```
void setup() {
  ... //Initialisierung
}
void draw() {} //kann leer bleiben, muss aber da sein
void mousePressed() {
  ... //Verändern der Hintergrundfarbe
}
```

Zusammenfassung - I

Funktionsdefinition

```
int add(int a, int b) {
  return a + b;
}
```

Funktionsaufruf

```
int x = add(10,3);
```

Zusammenfassung - II

keyPressed

```
int counter = 0;
void keyPressed() {
  if(key == 'a') {
    counter++;
  } else if (key == 'x') {
    exit(0);
  }
}
```

mousePressed

```
void mousePressed() {
  if(mouseButton == LEFT){
    background(255);
  }
  else{
    background(0);
  }
}
```