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

Brückenkurs Programmieren

Tag 2: Animationen, Schleifen und Arrays

Jakob Czekansky, M.Sc.
Gastvortrag von Hendrik Wagner

Technische Hochschule Mittelhessen

14. 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

\+ - * / %

Inhalt

Inhalt

Animationen

Bisher: Erzeugen eines einzelnen Bildes ⇒ laaangweilig

Beispiel: Animation

```
float x = 0;
float vx = 8;
void setup() {
  size(800,100);
  noStroke();
  background(0);
}
```

```
void draw() {
  fill(0,0,0,20);
  rect(0,0,width,height);
  fill(128,186,36);
  float y = height/2.0;
  y += 20*sin(x*0.02);
  ellipse(x,y,50,50);
  x += vx;
  if(x >= width || x <= 0) {
    vx *= -1;
  }
}
```

Animationen: Was ist bekannt, was ist neu?

- Variablen ✓
- Arithmetische Operatoren +, −, ∗, / ✓
- Zuweisungsoperatoren =, +=, ∗=. ... ✓
- `rect`, `fill`, `ellipse`, ... ✓
- If-Abfrage ✓

Animationen: Was ist bekannt, was ist neu?

- Variablen ✓
- Arithmetische Operatoren +, −, ∗, / ✓
- Zuweisungsoperatoren =, +=, ∗=. ... ✓
- `rect`, `fill`, `ellipse`, ... ✓
- If-Abfrage ✓
- `void setup() {...}` ?
- `void draw() {...}` ?

Animationen: `setup()` und `draw()`

Definition

`setup()` und `draw()` sind vordefinierte Funktionen von Processing. Sie dienen dazu, den Code zu strukturieren und eine game loop aufzubauen.

- `setup()` wird einmal zum Start ausgeführt.
- `draw()` wird ca. alle 15 ms ausgeführt in einer Schleife (engl. *loop*).

Syntax

```
<Variablendefinitionen>
void setup() {
  <Anweisungsblock>
}
void draw() {
  <Anweisungsblock>
}
```

Aufgabe: Der springende Punkt

Tippe das folgende Beispiel ab und erweitere es, so dass der Ball auch eine vertikale Geschwindigkeit bekommt.

Beispiel

```
//horizontale Position
float x = 1;
//horizontale Geschwindigkeit
float vx = 4;

void setup() {
  size(400,300);
}
```

```
void draw() {
  background(0);
  x = x + vx;
  //Ball prallt ab (l/r)
  if (x <= 0 || x >= width){
    vx = vx * -1;
  }
  ellipse(x, height/2.0, 20, 20);
}
```

Animationen: Das Spiel mit der Maus

Immer noch keine Interaktion $\Rightarrow$ immer noch alles langweilig

Idee

Fang den springenden Punkt mit der Maus!

Umsetzung

- Wo befindet sich die Maus?
- Wie lautet die formale Bedingung für „getroffen“?

Animationen: Wo befindet sich die Maus?

Definition

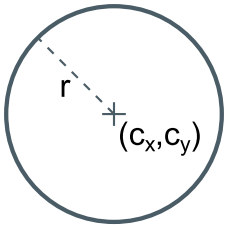
Die Variablen `mouseX` und `mouseY` werden von Processing automatisch gesetzt. Sie geben die aktuelle Mausposition an.

Beispiel

```
void setup() {
  size(400,300);
}
void draw() {
  ellipse(mouseX,mouseY,10,10);
}
```

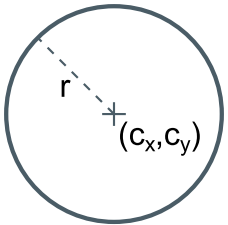
Animationen: Punkt im Kreis

Frage an die Mathematiker: Wann befindet sich ein Punkt in einem Kreis?



Animationen: Punkt im Kreis

Frage an die Mathematiker: Wann befindet sich ein Punkt in einem Kreis?

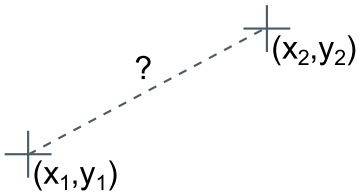


Antwort

Die Distanz zwischen dem Mittelpunkt des Kreises und dem angegebenen Punkt muss kleiner als der Radius des Kreises sein.

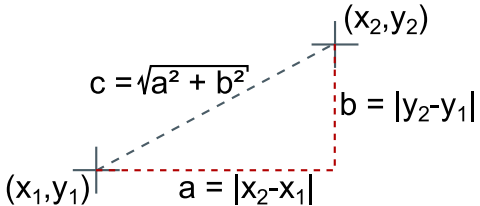
Animationen: Distanzbestimmung

Nächste Frage: Wie bestimmt man die Distanz zwischen zwei Punkten?



Animationen: Distanzbestimmung

Nächste Frage: Wie bestimmt man die Distanz zwischen zwei Punkten?

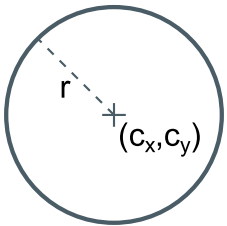


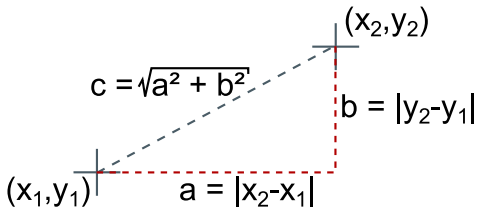
Antwort

Mit dem Satz des Pythagoras lassen sich Distanzen bestimmen. Dafür gibt es in Processing die vordefinierte Funktion `dist(x1,y1,x2,y2)`.

Aufgabe: Fang den Ball!

Ändere dein Programm zum springenden Ball, so dass der Ball sich jetzt nur noch bewegt, wenn die Maus sich außerhalb des Kreises befindet.





Erinnerung

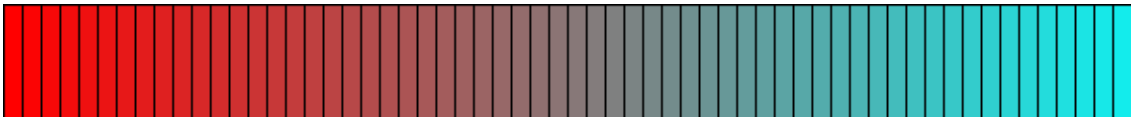
- Mausposition: `mouseX`, `mouseY`
- Distanzbestimmung: `dist(x1,y1,x2,y2)`

Inhalt

Schleifen

Aufgabe

Stelle einen Farbverlauf mit 60 einzelnen Rechtecken dar.

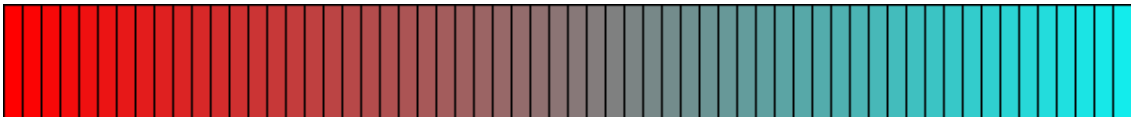


Bisher: Eine Programmzeile pro Rechenoperation.

Schleifen

Aufgabe

Stelle einen Farbverlauf mit 60 einzelnen Rechtecken dar.

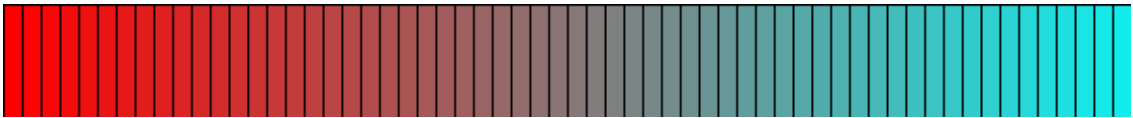


Bisher: Eine Programmzeile pro Rechenoperation.

Schleifen

Aufgabe

Stelle einen Farbverlauf mit 60 einzelnen Rechtecken dar.



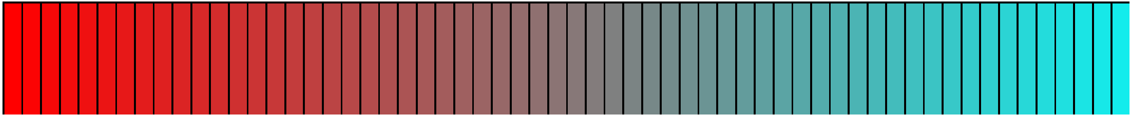
Bisher: Eine Programmzeile pro Rechenoperation.

```
int x = 0;
fill(255-x*4,x*4,x*4);
rect(x*10,0,10,60);
x = x + 1;
...
```

Schleifen

Aufgabe

Stelle einen Farbverlauf mit 60 einzelnen Rechtecken dar.



Bisher: Eine Programmzeile pro Rechenoperation.

```
int x = 0;
fill(255-x*4,x*4,x*4);
rect(x*10,0,10,60);
x = x + 1;
...
```

```
fill(255-x*4,x*4,x*4);
rect(x*10,0,10,60);
x = x + 1;
fill(255-x*4,x*4,x*4);
rect(x*10,0,10,60);
x = x + 1;
fill(255-x*4,x*4,x*4);
rect(x*10,0,10,60);
x = x + 1;
fill(255-x*4,x*4,x*4);
rect(x*10,0,10,60);
x = x + 1;
fill(255-x*4,x*4,x*4);
rect(x*10,0,10,60);
x = x + 1;
fill(255-x*4,x*4,x*4);
rect(x*10,0,10,60);
x = x + 1;
fill(255-x*4,x*4,x*4);
rect(x*10,0,10,60);
x = x + 1;
fill(255-x*4,x*4,x*4);
rect(x*10,0,10,60);
```

```
fill(255-x*4,x*4,x*4);
rect(x*10,0,10,60);
x = x + 1;
fill(255-x*4,x*4,x*4);
rect(x*10,0,10,60);
x = x + 1;
fill(255-x*4,x*4,x*4);
rect(x*10,0,10,60);
x = x + 1;
fill(255-x*4,x*4,x*4);
rect(x*10,0,10,60);
x = x + 1;
fill(255-x*4,x*4,x*4);
rect(x*10,0,10,60);
x = x + 1;
fill(255-x*4,x*4,x*4);
rect(x*10,0,10,60);
x = x + 1;
fill(255-x*4,x*4,x*4);
rect(x*10,0,10,60);
x = x + 1;
fill(255-x*4,x*4,x*4);
rect(x*10,0,10,60);
```

```
fill(255-x*4,x*4,x*4);
rect(x*10,0,10,60);
x = x + 1;
fill(255-x*4,x*4,x*4);
rect(x*10,0,10,60);
x = x + 1;
fill(255-x*4,x*4,x*4);
rect(x*10,0,10,60);
x = x + 1;
fill(255-x*4,x*4,x*4);
rect(x*10,0,10,60);
x = x + 1;
fill(255-x*4,x*4,x*4);
rect(x*10,0,10,60);
x = x + 1;
fill(255-x*4,x*4,x*4);
rect(x*10,0,10,60);
x = x + 1;
fill(255-x*4,x*4,x*4);
rect(x*10,0,10,60);
x = x + 1;
fill(255-x*4,x*4,x*4);
rect(x*10,0,10,60);
```

```
fill(255-x*4,x*4,x*4);
rect(x*10,0,10,60);
x = x + 1;
fill(255-x*4,x*4,x*4);
rect(x*10,0,10,60);
x = x + 1;
fill(255-x*4,x*4,x*4);
rect(x*10,0,10,60);
x = x + 1;
fill(255-x*4,x*4,x*4);
rect(x*10,0,10,60);
x = x + 1;
fill(255-x*4,x*4,x*4);
rect(x*10,0,10,60);
x = x + 1;
fill(255-x*4,x*4,x*4);
rect(x*10,0,10,60);
x = x + 1;
fill(255-x*4,x*4,x*4);
rect(x*10,0,10,60);
x = x + 1;
fill(255-x*4,x*4,x*4);
rect(x*10,0,10,60);
```

```
fill(255-x*4,x*4,x*4);
rect(x*10,0,10,60);
x = x + 1;
fill(255-x*4,x*4,x*4);
rect(x*10,0,10,60);
x = x + 1;
fill(255-x*4,x*4,x*4);
rect(x*10,0,10,60);
x = x + 1;
fill(255-x*4,x*4,x*4);
rect(x*10,0,10,60);
x = x + 1;
fill(255-x*4,x*4,x*4);
rect(x*10,0,10,60);
x = x + 1;
fill(255-x*4,x*4,x*4);
rect(x*10,0,10,60);
x = x + 1;
fill(255-x*4,x*4,x*4);
rect(x*10,0,10,60);
x = x + 1;
fill(255-x*4,x*4,x*4);
rect(x*10,0,10,60);
```

Schleifen: While

Definition

Eine Schleife (engl. *loop*) erlaubt die wiederholte Ausführung eines Anweisungsblocks solange eine Fortsetzungsbedingung erfüllt ist.

Syntax

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

Beispiel

```
int x = 0;
while(x < 60) {
  fill(255-x*4,x*4,x*4);
  rect(x*10,0,10,60);
  x = x + 1;
}
```

Aufgabe: Barcode

Erzeuge einen zufälligen Barcode mit einer Länge von 100 Pixeln.

Tipps:

- Zufallsentscheidung an jeder Stelle
- 50/50: `random(1) < 0.5`

Erinnerung: While-Syntax

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

While: Endlosschleife

Frage: Was passiert, wenn die Fortsetzungsbedingung immer erfüllt wird?

Processing

```
boolean gleichgewichtOk = true;
int alkoholpegel = 0;
while (gleichgewichtOK) {
  alkoholpegel = alkoholpegel + 1;
}
```

While: Endlosschleife

Frage: Was passiert, wenn die Fortsetzungsbedingung immer erfüllt wird?

Processing

```
boolean gleichgewichtOk = true;
int alkoholpegel = 0;
while (gleichgewichtOK) {
  alkoholpegel = alkoholpegel + 1;
}
```

Antwort: **Der Computer hört nie auf zu rechnen!**

Schleifen: For

Oft möchte man in einer Schleife einen Zähler verwenden und man weiß von Anfang an schon, wie oft die Anweisungen in der Schleife ausgeführt werden sollen.

⇒ Bauen wir doch einen Zähler direkt in das Schleifenkonstrukt ein!

Syntax: For-Schleife

```
for (<Initialisierung>; <Bedingung>; <Schrittanweisung>) {
  <Anweisungsblock>
}
```

For: Beispiel

Beispiel: For

```
for(int x = 0; x < 60; x++) {
  fill(255-x*4,x*4,x*4);
  rect(x*10,0,10,60);
}
```

Vergleich: While

```
int x = 0;
while(x < 60) {
  fill(255-x*4,x*4,x*4);
  rect(x*10,0,10,60);
  x = x + 1;
}
```

Weiterer Vorteil:

- Schleifenkopf zeigt Anzahl der Durchläufe
- Endlosschleife sofort erkennbar

Inhalt

Arrays

Was macht man, wenn ein Programm viele Werte speichern muss?

- viele Werte speichern $\Rightarrow$ viele Variablen definieren
- genauso unpraktikabel wie Schleifen auszuschreiben
- Anzahl benötigter Variablen evtl. zu Beginn unbekannt

$\Rightarrow$ Variable von variabler Größe benötigt

Arrays: Definition, Zuweisung, Zugriff

Ein Array ist eine Variable, die mehrere Werte vom gleichen Typ enthält.

Definition

```
float[] a = new float[3];
int[] b = new int[]{1,2,4};
```

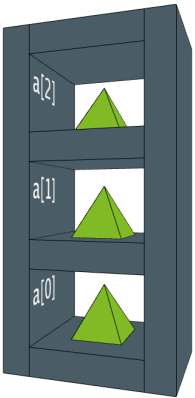
Zuweisung

```
b[0] = 1;
a[1+1] = 3.4;
```

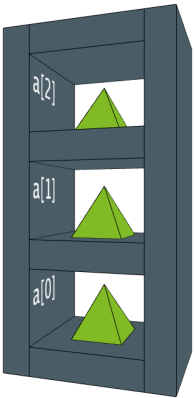
Zugriff

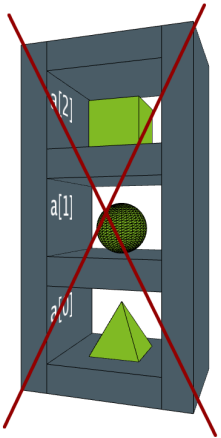
```
int x = b[0];
println(a[x+1]);
int n = a.length;
```

Array als Regal

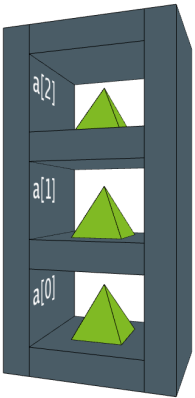


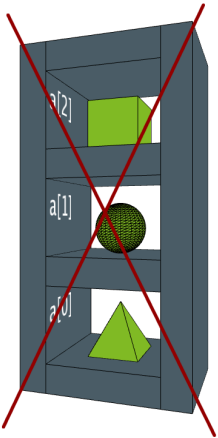
Array als Regal

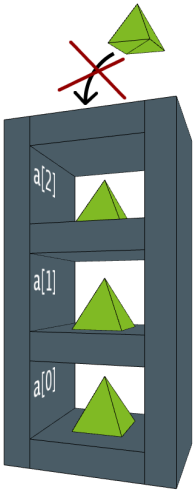




Array als Regal





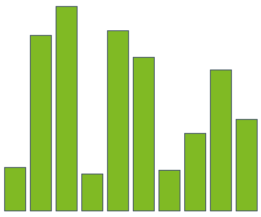


Aufgabe: Balkendiagramm

Erstelle ein Array mit 10 positiven ganzen Zahlen und stelle es als Balkendiagramm dar.

Allgemeine Syntax

```
int[] ar = new int[10];
ar[0] = 4;
int x = ar[2];
```



Zusammenfassung - I

Beispiel: For-Schleife

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

Syntax: While-Schleife

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

Zusammenfassung - II

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];
```