



Medientechnik

Übung 5

Java Bildbearbeitung - Teil 2

Planung

Nr	Zeitraum	Thema
1	20.04. – 25.04.	Bildretusche mit Gimp
2	27.04. – 01.05.	GUI Programmierung
3	04.05. – 08.05.	Model-View Controller
4	18.05. – 22.05.	Bildfilter – Teil 1
5	26.05. – 29.05.	Bildfilter – Teil 2
6	01.06. – 03.06.	Video & Film Theorie
7	29.06. – 02.07	Audio-Aufnahme und –Bearbeitung

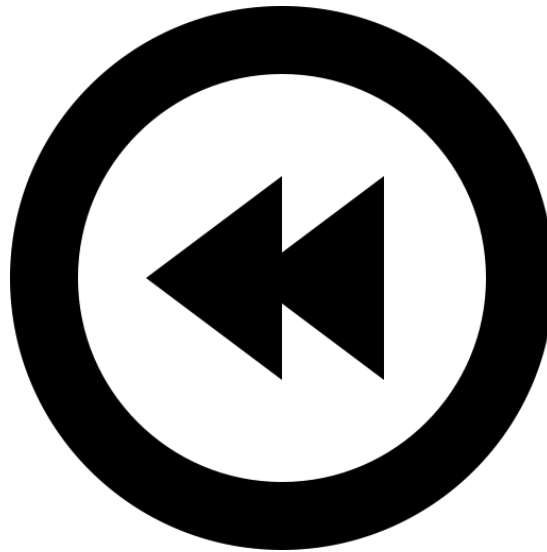
Keine Übungen zw. 03.06. und 29.06.!

Java2D – Bildbearbeitung

- Teil 1 (letzte Woche):
 - Bilder laden
 - Konvolution
 - Filter von JH Labs
- Teil 2:
 - Eigene Filter

Rewind

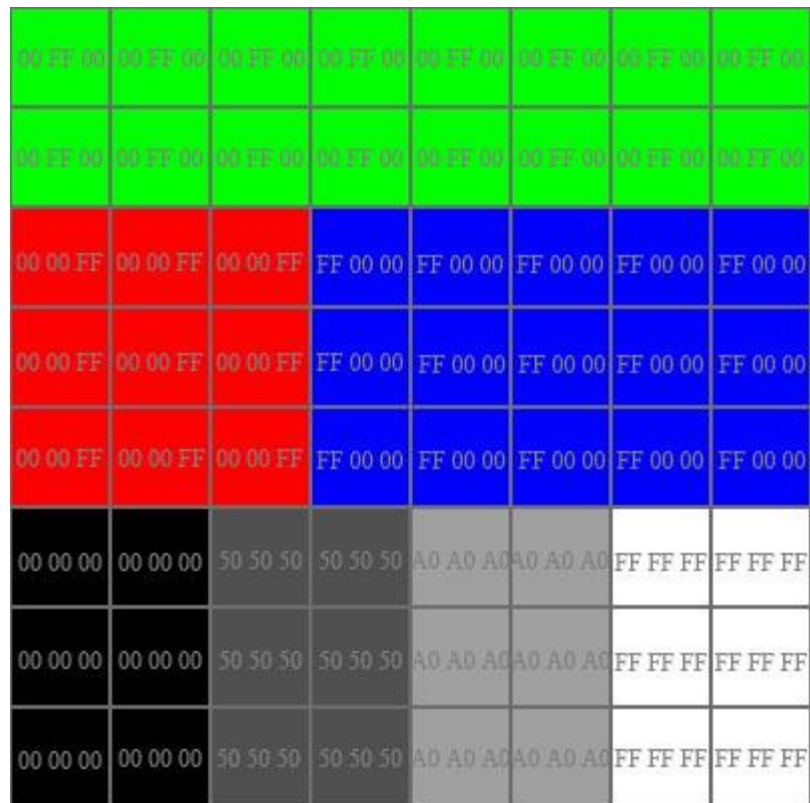
- Was ist Konvolution?
- Was ist ein Kernel?
- Warum benötigen wir ein BufferedImage?



Eigene Filter

Theorie: Wie kann man Bilder auf Pixelbasis manipulieren?

- Spezielle Datenstruktur speichert Werte der einzelnen Pixel
- Zugriff und Manipulation einzelner Pixel durch Bit-Operatoren (<<, >>, &, |)



Eigener Filter

Beispiel: BufferedImage (RGB):

- Zugriff auf einzelnes Pixel:

```
public int getRGB(int x, int y)
```

- Laden der Pixelwerte in ein RGB Array:

```
public int[] getRGB(  
    int startX, int startY,  
    int w, int h,  
    int[] rgbArray, int offset,  
    int scansize)
```

Iteration über Pixel

- Zwei for-Schleifen

```
for (int x = 0; x < image.getWidth(); x++){
    for (int y = 0; y < image.getHeight(); y++) {
        int pixelValue = image.getRGB(x, y);
        // do something...
    }
}
```

- 1-D Array

```
int[] allPixels =
    new int[image.getWidth()*image.getHeight()];

image.getRGB(0, 0,
    image.getWidth(), image.getHeight(),
    allPixels, 0, 0);

for (int pixelValue : allPixels) {
    // do something...
}
```

RGB Farbwerte interpretieren

Dezimal: **R** **G** **B**
 (74, 156, 122)

Hexidezimal: (4A, 9C, 7A) \rightarrow $0x4A9C7A_{16}$

$$0x4A9C7A_{16} \\ = 4889722_{10}$$

Das ist der RGB-Wert des Pixels,
Der von `.getRGB(...)` zurückgegeben wird

Farbwerte zurückrechnen

$$= \underbrace{01001010}_R \quad \underbrace{10011100}_G \quad \underbrace{01111010}_B$$

4889722_{10}

Wie kommt man jetzt an den Grün-Wert?

Grün-Wert herausfiltern (1)

01001010 10011100 01111010₂

Wir wollen an die zweiten 8 Bits.

Idee:

große Bitmaske mit “logischem Und”

```
    01001010 10011100 01111010
& 00000000 11111111 00000000
-----
    00000000 10011100 00000000
```

Grün-Wert herausfiltern (2)

00000000 10011100 00000000₂

Idee:

Bits um 8 Stellen nach rechts “verschieben”

00000000 10011100 00000000

>> 8

00000000 00000000 10011100

Farbwerte herausfiltern

- Bitshift kann auch vor der Bit-Maske kommen

- Operationen können gemeinsam ausgeführt werden:

```
int green = (pixelValue >> 8) & 0b11111111;
```

- Bitmasken funktionieren auch mit Hexadezimal-Werten:

```
int green = (pixelValue >> 8) & 0xff;
```

Farbwert zurückschreiben

Nach Berechnung / Modifikation müssen wir wieder einen eindeutigen Integer Wert erzeugen. → “logisches Oder”

```
int newPixelValue = (red << 16) | (green << 8) | blue;
```

→ **r = 01001010** **g = 10011100** **b = 01111010**

```
      01001010  00000000  00000000
      | 00000000  10011100  00000000
      | 00000000  00000000  01111010
      -----
      01001010  10011100  01111010
```

```
image.setRGB(x, y, newPixelValue);
```

Color Klasse statt Arithmetik

- Man kann auch Color Objekte erzeugen, und darüber and die Werte der Farbkanäle gelangen:

```
Color color = new Color(image.getRGB(x, y));  
int red = color.getRed();  
int green = color.getGreen();  
int blue = color.getBlue();  
  
// modify red/green/blue, then:  
int newPixelValue = (red << 16) | (green << 8) | blue;  
image.setRGB(x, y, newPixelValue);
```

- Nachteile:
 - man kan die Werte nicht mehr ins Color-Objekt zurückschreiben.
 - Overhead durch neue Objekte (potenziell Performanz-Verlust)

Farbkanal extrahieren



Bild: Tobias Stockinger, CC-BY-NC

Farbkanal extrahieren: Idee

- Wähle aus 3 Farbkanälen diejenigen, die übrig bleiben sollen, z.B. **rot**.
- Setze die anderen Farbwerte jeweils auf 0 → kein Farbanteil am Pixel,
rot : 100% - **grün** : 0% - **blau**: 0%
- Die angezeigten Pixel unterscheiden sich nur in der Helligkeit des **roten** Kanals!

Farbkanal extrahieren: Code

```
BufferedImage targetImage = new BufferedImage(  
    image.getWidth(),  
    image.getHeight(),  
    BufferedImage.TYPE_INT_RGB);  
  
for(int x = 0; x < image.getWidth(); x++){  
    for(int y = 0; y < image.getHeight(); y++){  
        int pixel = image.getRGB(x, y);  
        int red = (pixel >> 16) & 0xff;  
        int targetPixel = red << 16;  
        targetImage.setRGB(x, y, targetPixel);  
    }  
}
```



Wrap-up Quiz

1. Was ist eine „Bit-Maske“
2. Warum benötigen wir überhaupt Bit-Shift Operationen?
3. Was ist der Vorteil von Bit-Shift Operationen?
4. Wie ist die Grund-Idee zum Einfärben von Bildern?



Übungsblatt 3

- Bildfilter Funktionalität implementieren
- Bildverarbeitung (Theorie)


 LMU München, Übungen zur Vorlesung/Medientechnik im Sommersemester 2015 

Übungsblatt 3: Bildfilter


Aufgabe 1: **4 Punkte**



a) Erklären Sie kurz den Aufbau eines Histogramms (z. B. Achsenbeschriftungen). Welche Informationen kann man aus einem Histogramm ablesen? (1 Punkt)

b) Deuten Sie das folgende Histogramm. Wie sieht das Bild in etwa aus? Welche Probleme gibt es? (1 Punkt)



c) In der Vorlesung haben Sie die „Rule of Thirds“ kennengelernt. Das folgende Bild wurde entsprechend in 3 horizontale Abschnitte aufgeteilt. Zeichnen Sie zwei Histogramme (grobe Zeichnung genügt) für das obere und mittlere Drittel und erklären Sie Ihre Überlegungen dazu. (2 Punkte)



 LMU München, Übungen zur Vorlesung/Medientechnik im Sommersemester 2015 

Aufgabe 2: Konvolution

3 Punkte

a) Erklären Sie die Funktionsweise einer Konvolution (Faltung) mit Hilfe einer Kernelmatrix auf Bildern. Welche Probleme treten an den Rändern auf? (1,5 Punkte)

b) Was passiert bei der Anwendung der folgenden Filter? Das f steht für „float“. (je 0,5 Punkte)

a)
$$\begin{bmatrix} 1/9f & 1/9f & 1/9f \\ 1/9f & 1/9f & 1/9f \\ 1/9f & 1/9f & 1/9f \end{bmatrix}$$

b)
$$\begin{bmatrix} 1f & 1f & 1f \\ 1f & -8f & 1f \\ 1f & 1f & 1f \end{bmatrix}$$

c)
$$\begin{bmatrix} -1f & -1f & -1f \\ -1f & 9f & -1f \\ -1f & -1f & -1f \end{bmatrix}$$


Aufgabe 3: Bildfilter in JAVA

19 Punkte

a) Der vorhandene Bildbetrachter aus den letzten Übungsblättern soll nun mit einigen Fotobearbeitungseffekten ausgestattet werden. Dabei sollen die Checkboxes im Menü „Filters“ der De-Aktivierung der Effekte dienen, wobei weiterhin MVC und das Observer-Pattern eingesetzt werden sollen. Das übrige Programmverhalten wurde in Übungsblatt 2 definiert.

Aufbauend auf den bereits angelegten GUI-Elementen sollen folgende Filter selbst implementiert und nutzbar gemacht werden (falls Parameter erforderlich sind, finden Sie die passenden heraus!). Fügen Sie ggf. weitere Schaltflächen zur GUI hinzu und passen Sie die Funktionalität der Buttons an! (je 2 Punkte pro Filter)

- Black/White/Threshold-Filter





Vielen Dank!

WELCHE FRAGEN GIBT ES? 😊