Medientechnik – Vorbereitung ÜB3/Übung 4 – Tutorial

Ausgangsprojekt ist auf der Homepage zu finden, runterladen lassen und in Eclipse einbinden.

1. Ordnerstruktur und Klassen erklären:

- Images-Ordner mit Bild(ern)

- Lib-Ordner mit externer Library *Filters.jar* (von JH Labs, siehe Folien)
    - Wird in der .classpath-Datei gesetzt
    - Kann alternativ in Eclipse/Netbeans/… über die Projekteigenschaften -> Java Build Path verwaltet werden (neue Libraries hinzufügen/entfernen etc.)
- ImageRead.java: beinhaltet verschiedene statische Methoden, um ein Bild zu laden (einfach mal ausprobieren, was funktioniert)
- ImageView.java: beinhaltet bereits Methoden zum Laden und Zeichnen von einem Original-Bild und einem gefilterten Bild ->laufen lassen

2. Bild laden:

2 Möglichkeiten anschauen, ausprobieren

Falls es später beim Filtern zu Problemen kommt, die andere Variante probieren

3. Java Filter:

- ConvolveOp (javaFilter1):
    - Neue Kernelmatrix anlegen

```
float[] values = {

      1/9f, 1/9f, 1/9f,
      1/9f, 1/9f, 1/9f,
      1/9f, 1/9f, 1/9f
};
```

    - Neue ConvolveOp erzeugen

```
ConvolveOp cOp= new ConvolveOp(kernel);
```

    - Filtern

cOp.filter(srcImg, dstImg);

- Was bewirkt welche Matrix? → Alle einfach mal ausporbieren

- ColorConvertOp
  - Neue ColorConvertOp erzeugen

    ```
    ColorConvertOp ccOp = new ColorConvertOp(
    ColorSpace.getInstance(ColorSpace.CS_GRAY), null);
    ```

  - Filtern

    cc0p.filter(srcImg, dstImg);

4. Eigener Filter:

- Höhe und Weite des Bildes in Variablen speichern
- Durch die Pixel des Bildes laufen und jeweils RGB-Wert auslesen:

```
int w = srcImg.getWidth();
int h = srcImg.getHeight();

for (int x = 1; x <w; x++){
      for (int y=1; y<h; y++){
            int p = srcImg.getRGB(x, y);
            int alpha = (p >> 24) & 0xff;
            int red   = (p >> 16) & 0xff;
            int green = (p >> 8) & 0xff;
            int blue  = (p)  & 0xff;
      }
}
```

- Je nur den Rot-/Grün-/Blaukanal im dstImg sichtbar machen (noch in der for-Schleife)

```
int neu  = red << 16;
dstImg.setRGB(x,y, neu);
```

oder

```
int neu  = green << 8;
dstImg.setRGB(x,y, neu);
```

oder

```
int neu  = blue;
dstImg.setRGB(x,y, neu);
```

- Für schnellere Bildbearbeitung nicht zwei `for`-Schleifen, sondern ein Array mit allen Pixeln (s. Source-Code)

5. JHLabs Filter:

- Filter eines Drittanbieters, der bereits konfigurierte Filter anbietet:

  http://www.jhlabs.com/ip/filters/index.html

- Filter lassen sich aber über gewisse Parameter an eigene Bedürfnisse anpassen (s.u. ... einfach mit den Parametern ein wenig rumprobieren)

SolarizeFilter (oder anderen) Filter einbinden und filtern:

```
SolarizeFilter sOp = new SolarizeFilter();
sOp.filter(srcImg, dstImg);
```

oder

```
PointillizeFilter pOp = new PointillizeFilter();
pOp.setFuzziness(0.9f); // oder 0.1f
pOp.setScale(10f); // oder 1f, 50f;
pOp.filter(srcImg, dstImg);
```

oder

```
InvertFilter iOp = new InvertFilter();
iOp.filter(srcImg, dstImg);
```

6. repaint():

Für das Übungsblatt wichtig: Wenn das „gefilterte“ Bild verändert wurde, d.h. im Model wurden Filter (de-)aktiviert, muss die GUI beim update() neu gezeichnet werden... → repaint()