Übung 7 – Medientechnik

„Java Advanced Imaging“

Aufgaben:

Wir wollen uns heute mit den verschiedenen Operatoren des Java Advanced Imaging vertraut machen. Dazu soll wieder ein Frame mit 3 Panels erstellt werden. Je nach Aufgabenstellung sollen in den oberen Panels 1 bzw. 2 Originalbilder gezeigt werden, im untersten dann das Ergebnisbild.

1) Erstellen des Vorschaufensters

   ToDo:

   - Erzeugen eines JFrames
   - Erzeugen von 3 Panels
   - Setzen der Panelgröße
   - Layout des Frames festsetzen (Borderlayout)
   - Hinzufügen der Panels zum Frame

2) 2 Bilder aussuchen und im Projekt-Verzeichnis speichern

3) Jetzt sollen die beiden ausgesuchten Bilder in den ersten beiden Panels angezeigt werden (ROT markierte Begriffe müssen in der API-Doc nachgelesen werden!)

   ToDo:

   - Öffnen von zwei `FileSeekableStreams`, um die ausgesuchten Files dem Programm bereitstellen zu können. Möglichst mit Pfaden als Aufrufparameter.
   - Um auf die Bilder zugreifen zu können benötigt man RenderedOps der Bilder.
     „`RenderedOp image1 = JAI.create("stream", stream);`“
   - Dabei soll try-and-catch verwenden, um eventuelle IOExceptions abfangen zu können.

4) Erstellen der Main-Klasse

   Mit dieser Basisklasse sollen nun verschiedene Operatoren ausprobiert werden. Damit man nicht für jede neue Teilaufgabe den Code des vorigen Operators auskommentieren muss, ist es nützlich für alle folgenden Teilaufgaben jeweils einen „Ableger“ der bisher entstandenen Klasse zu basteln. (z.B. von meiner bisherigen Klasse JAI: JAI4a, JAI4b, ...)

a) Implementation von Addition, Subtraktion, Multiplikation, Division der 2 Originalbilder

```
ParameterBlock(pb):
Addieren der Quellen (Bild1, Bild2)
pb.addSource(src1);
pb.addSource(src2);

target = JAI.create("add", pb, null);
target = JAI.create("subtract", pb, null);
target = JAI.create("multiply", pb, null);
target = JAI.create("divide", pb, null);

create(string,parameterblock,Rendering Hints);
```

-> Ergebnis anschauen.

b) Jetzt soll nur 1 Originalbild mit Hilfe einer **LookupTable** farblich verändert werden.

ToDo:

-Erstellen der **Lookup Table**

```
"private byte lut[][];"
```

-Füllen der Lookuptable mit Werten (Bsp: rötlich)

```
for ( i = 0; i < 256; i++ ) {
                lut[0][i] = (byte)i;
                lut[1][i] = (byte)0;
                lut[2][i] = (byte)0;
            }
```

-Ausprobieren anderer Farbkombinationen

c) Convolution: Kantenglättung/Schärfung

- Erstellen eines **KernelJAI** 3x3 mit Übergabe des entsprechenden Floatarrays. Zum **RenderingBlock** hinzufügen ändern des Strings in der RenderedOp auf „convolve“. Danach das Bild zum Ergebnis-Panel adden.
  ➔ Ergebnis zu analysieren…

- Zu Implementieren sind:

  I) Low Pass Filter

  Matrix:

```
0.11F, 0.11F, 0.11F,
0.11F, 0.11F, 0.11F,
0.11F, 0.11F, 0.11F
```

  II) High Pass Filter

  Matrix:

```
-1.0F, -1.0F, -1.0F,
-1.0F, 9F, -1.0F,
-1.0F, -1.0F, -1.0F
```

III) Laplace Filter (optional)

Matrix:

```
1.0F, -2.0F, 1.0F,
-2.0F, 5.0F, -2.0F,
1.0F, -2.0F, 1.0F
```

IV) Emboss Effekt (optional)

Matrix:

```
-1.0F, -2.0F, 0F,
-2.0F, 0F, 2.0F,
0F, 2.0F, 1.0F
```

d) Für die etwas Schnelleren….

- Zur Teilaufgabe 4a) gibt es noch andere Kombinationsmöglichkeiten wie z.B. „xOr“ oder „Invert“ Experimentieren sie mit diesen (->JAI-API)
- Zur Teilaufgabe 4b) gibt es auch noch die Möglichkeit das ColorModel zu ändern und dann auf das ausgesuchte Bild anzuwenden.
- Zur Aufgabe 4c) kann man noch die optionalen Filter implementieren oder mit den Werten des Kernels „spielen“

Links/Literatur:

http://java.sun.com/products/java-media/jai/forDevelopers/jai-apidocs/
http://java.sun.com/products/java-media/jai/forDevelopers/jai1_0_1guide-unc/index.html
http://java.sun.com/developer/onlineTraining//javaai/

+ Vorlesungsunterlagen ;-)