Course IPJava: Image Processing with Java
Chapter C5: The Convolution Project

Projekt convolution1 mit leerem Fenster

Im Directory `C:\temp` ein Sub-Directory `convolution1` anlegen.
TextPad starten `Datei Neu`
Leere Datei speichern: `Datei Speichern unter C:\temp\convolution1\Convolution1.java`
Schreiben Sie folgende Zeilen:

```
import java.awt.Container;
import java.awt.BorderLayout;
import java.awt.Color;
import javax.swing.JFrame;

public class Convolution1
{
  public static void main(String args[])
  {
    JFrame frame=new JFrame("Convolution and Ranking Filter");
    frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    frame.setSize(800,600);
    Container cp=frame.getContentPane();
    cp.setBackground(Color.WHITE);
    cp.setLayout(new BorderLayout());
    frame.setVisible(true);
  }
}
```

Übersetzen mit der Tastenkombination `Strg+1`. Ausführen mit der Tastenkombination `Strg+2`. Es öffnet sich ein Fenster, das sich nur minimieren, maximieren und schließen lässt. Die Klasse `Convolution1` dient also lediglich dem Aufbau unserer GUI und (später) allen notwendigen Objekt-Instantiierungen.

Zeichenfläche

Beenden Sie das Programm.
Legen Sie eine neue Datei in TextPad an: `Datei Neu`
Leere Datei speichern: `Datei Speichern unter C:\temp\convolution1\DrawArea.java`
Schreiben Sie folgende Klasse:

```
import java.awt.*;
import java.awt.event.*;
import javax.swing.JComponent;
import java.util.Arrays;
```

```
public class DrawArea extends JComponent
{
  int myWidth, myHeight;
  int pWidth, pHeight;
  final int X_SIZE=11;
  final int Y_SIZE=12;
  final int[][] conv3Matrix={{0,1,0},
                             {1,4,1},
                             {0,1,0}};
  final int[][] conv5Matrix={{0, 0, 0, 0, 0},
                             {0,-1,-1,-1, 0},
                             {0,-1,10,-1, 0},
                             {0,-1,-1,-1, 0},
                             {0, 0, 0, 0, 0}};
  byte[][] b=new byte[Y_SIZE][X_SIZE];
  Color[] color;
  public DrawArea()
  {
    Dimension size=getSize();
    myWidth=size.width;
    myHeight=size.height;
    addComponentListener(new ComponentAdapter()
    {
      public void componentResized(ComponentEvent ev)
      {
        Dimension size=getSize();
        myWidth=size.width;
        myHeight=size.height;
      }
    });
    fillColorArray();
    clear();
  }
  private void fillColorArray()
  {
    color=new Color[10];
    for(int i=0; i<color.length; i++)
      color[i]=new Color(i*25,i*25,i*25);
  }
  public void paintComponent(Graphics g)
  {
    super.paintComponent(g);
    setPixelMeasuresToArea();
    for(int y=0; y<Y_SIZE; y++)
      for(int x=0; x<X_SIZE; x++)
      {
        g.setColor(color[b[y][x]]);
        g.fillRect(x*pWidth, y*pHeight, pWidth, pHeight);
      }
  }
  /**
   * Sets the pWidth and pHeight to values relative to the current DrawArea
   * size.
   */
  private void setPixelMeasuresToArea()
  {
    pWidth=myWidth/X_SIZE;
    pHeight=myHeight/Y_SIZE;
  }
```

```
  public void reset()
  {
    byte[][] tmp={{9,0,0,0,0,0,0,0,0,0,9},
                  {0,0,0,0,9,9,9,0,0,0,0},
                  {0,0,0,0,9,5,9,0,0,0,0},
                  {0,0,0,0,9,9,9,0,0,0,0},
                  {0,0,0,0,0,9,0,0,0,0,0},
                  {0,9,9,9,9,9,9,9,9,9,0},
                  {0,0,0,0,9,9,9,0,0,0,0},
                  {0,0,0,0,9,9,9,0,0,0,0},
                  {0,0,0,0,9,0,9,0,0,0,0},
                  {0,0,0,0,9,0,9,0,0,0,0},
                  {0,0,0,0,9,0,9,0,0,0,0},
                  {9,0,0,0,0,0,0,0,0,0,9}};
    b=tmp;

  }
  public void convolution(int z)
  {
  }
  public void ranking()
  {
  }
  public void noise()
  {
  }
  public void clear()
  {
  }
}
```

Übersetzen mit `Strg+1`.
Es wäre ebenso möglich die neue Klasse in die Datei `Convolution1.java` zu schreiben und nicht in ein extra File zu speichern. Allerdings ist eine zusätzliche Datei wesentlich übersichtlicher.
Nun müssen wir unserem `JFrame` noch eine Instanz der neuen Klasse als GUI-Komponente zuweisen.
Wechseln Sie dazu in TextPad zur Datei `Convolution1`: `Fenster 1 C:\...\Convolution1.java`
Ergänzen Sie die Methode `public static void main(String[] args)` von `Convolution1`, so dass sie so aussieht:

```
    public static void main(String[] args)
    {
      JFrame frame=new JFrame("Faltung und nichtlinearer Rangordnungsfilter");
      DrawArea drawArea=new DrawArea();
      frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
      frame.setSize(800,600);
      Container cp=frame.getContentPane();
      cp.setBackground(Color.WHITE);
      cp.setLayout(new BorderLayout());
      cp.add(drawArea);
      frame.setVisible(true);
    }
```

Übersetzen mit `Strg+1`, ausführen mit `Strg+2`.

Wichtig:
Wenn die Java Runtime Environment eine Java-Klasse ausführt, dann sucht sie automatisch nach der `main()`-Methode in dieser Klasse. Führen Sie immer nur die Klasse mit der `main()`-Methode aus - in unserem Fall also immer nur die Klasse `Convolution1` - ansonsten erhalten Sie eine Fehlermeldung.

Buttonleiste

Beenden Sie das Programm.
Legen Sie eine neue Datei in TextPad an: `Datei Neu`
Leere Datei speichern: `Datei Speichern unter C:\temp\convolution1\ButtonBar.java`
Schreiben Sie folgende Klasse:

```
import java.awt.GridLayout;
import java.awt.event.ActionListener;
import java.awt.event.ActionEvent;
import java.util.HashMap;
import javax.swing.JPanel;
import javax.swing.JButton;

public class ButtonBar extends JPanel
{
  static public final String[] BUTTON_NAME={"Homunculus", "Convolution 3x3",
      "Convolution 5x5", "Ranking Filter 3", "Noise", "Clear"};
  static public final String[] BUTTON_CMD={"drawHomunculus", "useConvolution3",
      "useConvolution5", "useRankingFilter", "useNoise", "clearArea"};
  JButton[] button=new JButton[BUTTON_NAME.length];
  DrawArea drawArea;
  int lpMidWeight=1;
  public ButtonBar(DrawArea draw)
  {
    drawArea=draw;
    setLayout(new GridLayout(0,1));
    ActionListener listener=new ButtonListener();
    for(int i=0; i<button.length; i++)
    {
      button[i]=new JButton(BUTTON_NAME[i]);
      button[i].setActionCommand(BUTTON_CMD[i]);
      button[i].addActionListener(listener);
      add(button[i]);
    }
  }
  //-------------------------------------------------------------------------
  class ButtonListener implements ActionListener
  {
    HashMap commandMap=new HashMap(BUTTON_CMD.length);
    public ButtonListener()
    {
      for(int i=0; i<BUTTON_CMD.length;i++)
        commandMap.put(BUTTON_CMD[i], new Integer(i));
    }
    public void actionPerformed(ActionEvent ev)
    {
      int cmd=((Integer)commandMap.get(ev.getActionCommand())).intValue();
      switch(cmd)
      {
        case 0:
          drawArea.reset();
          drawArea.repaint();
          break;
      }
    }
  }
}
```

Übersetzen mit `Strg+1`.
Nun müssen wir (wie vorher die `DrawArea`) unserem `JFrame` noch die Buttonleiste zuweisen.
Wechseln Sie dazu in TextPad zur Datei `Convolution1`: `Fenster 1 C:\...\Convolution1.java`
Fügen Sie die folgenden Zeilen als vorletzte Anweisungen (vor die Zeile `frame.setVisible(true);`) in die Methode `public static void main(String[] args)` von `Convolution1` ein:

```
    ButtonBar buttonBar=new ButtonBar(drawArea);
    cp.add(buttonBar, BorderLayout.EAST);
```

Übersetzen mit `Strg+1`, ausführen mit `Strg+2`. Erproben Sie die Buttonleiste. Allerdings funktioniert bisher nur der Button `Homunculus`.

Convolution

Beenden Sie das Programm.
Wechseln Sie in TextPad zur Datei `DrawArea`: `Fenster 2 C:\...\DrawArea.java`
Version 2: Erweitern Sie die Methode `public void convolution(int z)` der Klasse `DrawArea`, bis sie so aussieht:

```
    public void convolution(int z)
    {
      int[][] kernel;
      if(z==3)
        kernel=conv3Matrix;
      else if(z==5)
        kernel=conv5Matrix;
      else
        return;
      int d=z/2;
      int sum, divisor, weight, x, y, xx, yy, xxx, yyy;
      float min=Float.MAX_VALUE;
      float max=Float.MIN_VALUE;
      float gray, factor;

      float[][] temp=new float[Y_SIZE][X_SIZE];
      for(y=0; y<Y_SIZE; y++)
        for(x=0; x<X_SIZE; x++)
        {
          sum=divisor=0;
          for(yy=0; yy<=2*d; yy++)
          {
            yyy=y+yy-d;
            if(yyy<0 || yyy>=Y_SIZE)
              continue;
            for(xx=0; xx<=2*d; xx++)
            {
              xxx=x+xx-d;
              if(xxx<0 || xxx>=X_SIZE)
                continue;
              weight=kernel[yy][xx];
              sum+=b[yyy][xxx]*weight;
              divisor+=weight;
            }
          }
          if(divisor<=1)
            gray=(float)sum;
          else
            gray=(float)sum/(float)divisor;
          if(gray>max)
            max=gray;
          if(gray<min)
            min=gray;
          temp[y][x]=gray;//can be <0 or even >9
        }
      //map the calculated values to the interval from 0 till 9
      factor=9.f/(max-min);
      for(y=0; y<Y_SIZE; y++)
        for(x=0; x<X_SIZE; x++)
          b[y][x]=(byte)(factor*(temp[y][x]-min));
    }
```

Übersetzen mit `Strg+1`.
Wechseln Sie in TextPad zur Datei `ButtonBar`: `Fenster 3 C:\...\ButtonBar.java`

Schreiben Sie folgende neuen Cases unterhalb des letzten `break;` von `case 0:` in der Methode `public void actionPerformed(ActionEvent ev)` der Event Handler Klasse `ButtonListener`:

```
    case 1:
      drawArea.convolution(3);
      drawArea.repaint();
      break;
    case 2:
      drawArea.convolution(5);
      drawArea.repaint();
      break;
```

Übersetzen mit `Strg+1`. Wechseln Sie in TextPad zur Datei `Convolution1`: `Fenster 1 C:\...\Convolution1.java`. Ausführen mit `Strg+2`. Erproben Sie die 2 Faltungen.

Rangordnungsfilter

Beenden Sie das Programm.
Wechseln Sie in TextPad zur Datei `DrawArea`: `Fenster 2 C:\...\DrawArea.java`
Version 3: Erweitern Sie die Methode `public void ranking()` der Klasse `DrawArea`, bis sie so aussieht:

```
    public void ranking()
    {
      int x, y, xx, yy, n;
      byte[] rankingList=new byte[9];
      byte[][] temp=new byte[Y_SIZE][X_SIZE];
      for(y=0; y<Y_SIZE; y++)
        for(x=0; x<X_SIZE; x++)
        {
          n=0;
          for(yy=y-1; yy<=y+1; yy++)
          {
            if(yy<0 || yy>=Y_SIZE)
              continue;
            for(xx=x-1; xx<=x+1; xx++)
            {
              if(xx<0 || xx>=X_SIZE)
                continue;
              rankingList[n++]=b[yy][xx];
            }
          }
          Arrays.sort(rankingList, 0, n);
          temp[y][x]=rankingList[n/2];
        }
      for(y=0; y<Y_SIZE; y++)
        for(x=0; x<X_SIZE; x++)
          b[y][x]=temp[y][x];
    }
```

Übersetzen mit `Strg+1`.
Wechseln Sie in TextPad zur Datei `ButtonBar`: `Fenster 3 C:\...\ButtonBar.java`
Schreiben Sie folgenden neuen Case unterhalb des letzten `break;` von `case 2:` in der Methode `public void actionPerformed(ActionEvent ev)` der Event Handler Klasse `ButtonListener`:

```
    case 3:
      drawArea.ranking();
      drawArea.repaint();
      break;
```

Übersetzen mit `Strg+1`. Wechseln Sie in TextPad zur Datei `Convolution1`: `Fenster 1 C:\...\Convolution1.java`. Ausführen mit `Strg+2`. Erproben Sie den Rangordnungsfilter.

Noise und Clear

Beenden Sie das Programm.
Wechseln Sie in TextPad zur Datei `DrawArea`: `Fenster 2 C:\...\DrawArea.java`
Version 4: Erweitern Sie die Methode `public void noise()` der Klasse `DrawArea`, bis sie so aussieht:

```
public void noise()
{
  for(int y=0; y<Y_SIZE; y++)
    for(int x=0; x<X_SIZE; x++)
    {
      int noise=(int)(Math.random()*3)-1;
      noise+=b[y][x];
      if(noise<0)
        b[y][x]=0;
      else if (noise>9)
        b[y][x]=9;
      else
        b[y][x]=(byte)noise;
    }
}
```

Erweitern Sie die Methode `public void clear()` der Klasse `DrawArea`, bis sie so aussieht:

```
public void clear()
{
  for(int y=0; y<Y_SIZE; y++)
    for(int x=0; x<X_SIZE; x++)
      b[y][x]=0;
}
```

Übersetzen mit `Strg+1`.
Wechseln Sie in TextPad zur Datei `ButtonBar`: `Fenster 3 C:\...\ButtonBar.java`
Schreiben Sie folgende neuen Cases unterhalb des letzten `break;` von `case 3:` in der Methode `public void actionPerformed(ActionEvent ev)` der Event Handler Klasse `ButtonListener`:

```
case 4:
  drawArea.noise();
  drawArea.repaint();
  break;
case 5:
  drawArea.clear();
  drawArea.repaint();
```

Übersetzen mit `Strg+1`. Wechseln Sie in TextPad zur Datei `Convolution1`: `Fenster 1 C:\...\Convolution1.java`. Ausführen mit `Strg+2`. Erproben Sie `Noise` (auch mehrfach drücken) und `Clear`.

Experimente

(1) Erproben Sie andere 3x3 Filterkerne:

```
-2, 1, 1                              1, 1,-2
-2, 1, 1 = Filter für linke Kanten    1, 1,-2 = Filter für rechte Kanten
-2, 1, 1                              1, 1,-2

-2,-2,-2                              1, 1, 1
 1, 1, 1 = Filter für obere Kanten    1, 1, 1 = Filter für untere Kanten
 1, 1, 1                             -2,-2,-2

-1,-1,-1                              1, 1, 1
-1, 8,-1 = Filter für einsame Pixel   1,-8, 1 = Filter für einsame Lö;cher
-1,-1,-1                              1, 1, 1
```

(2) Entwerfen Sie weitere 3x3 Filterkerne speziell für die linke Hand, für die rechte Hand und 5x5 Filterkerne für beide Füße und für den Hals des Homunculus. Bauen Sie diese in die Buttonleiste ein.
(3) Variieren Sie den Homunculus und beobachten Sie die Ergebnisse Ihrer Filterkerne.
(4) Verschieben Sie im Rangordnungsfilter das ausgewählte Pixel (Befehl `temp[y][x]=rankingList[n/2];`) nach hö;heren oder nach niederen Rangordnungen. Beobachten Sie die Änderungen.

Weitere Aufgaben

Lesen sie die JavaTM 2 SDK, Standard Edition Documentation zu den Sprachelementen, die Sie geschrieben haben (siehe: http://java.sun.com/j2se/1.4.1/search.htm). Schauen Sie sich vor allem die Dokumentation zu den Klassen `java.awt.GridLayout`, `java.awt.BorderLayout`, `javax.swing.JButton`, `javax.swing.JPanel` und `java.util.HashMap` an. Schauen Sie sich auch die Dokumentation zur Mehtode `sort(byte[],int,int)` der Klasse `java.util.Arrays` an.
Erarbeiten Sie sich die Logik der 4 verschachtelten `for`-Schleifen in der Methode `public void convolution(int z)`.
Programmieren Sie eine vereinfachte Methode `myConvolution` der festen Grö;ße 3x3.
Erfinden und erproben Sie neue Varianten des Programms in Form von neuen Projekten `convolution2`, `convolution3` usw. nach obigem Muster.