Programmieren: Bildbearbeitung

Das Thema der folgenden Aufgaben ist Bildbearbeitung. Sie erhalten dazu ein Rahmenprogramm, das bereits Bilder lesen und darstellen kann. Dieses Rahmenprogramm basiert auf einem Programm der Universität Georgia Tech. Ihre Aufgabe ist es, die Bilder zu bearbeiten.

Sie haben bereits bei JavaKara einiges an "Bildbearbeitung" kennengelernt: Bilder spiegeln, Bilder um 90° drehen, Bilder verschieben. Damit haben Sie schon Übung mit der **Struktur eines Bildes:** Es ist ein zwei-dimensionaler Array.

Allerdings waren diese Bilder sozusagen nur schwarz-weiss Bilder: Kleeblatt gesetzt oder nicht gesetzt. Bei den folgenden Aufgaben bearbeiten wir nun Farbbilder. Jeder Bildpunkt hat dabei eine eigene **Farbe**. Wie diese Farben im Programm beschrieben werden, betrachten wir unten in einem Beispiel.

Beispielprogramm: Bild vertikal spiegeln

Um das Rahmenprogramm und die Struktur eines Bildes kennenzulernen, betrachten wir die Aufgabe "Bild vertikal spiegeln".

Ein Bild wird durch die **Klasse Picture** repräsentiert. Diese Klasse entspricht bei JavaKara der Klasse des Objekt world. Ein Picture hat eine Breite (abfragbar mit Methode getWidth()) und eine Höhe (Methode getHeight()).

Ein Bildpunkt wird durch die **Klasse Pixel** repräsentiert. Ein Pixel kann mit der Methode getPixel(x, y) abgeholt werden. Bei JavaKara entsprach das zum Beispiel isLeaf(x, y).

Ein Pixel repräsentiert Farben in der sogenannten **RGB-Darstellung**: Eine Farbe ist eine Mischung aus Rot-, Grün- und Blau-Werten, wobei jeder Wert im Bereich 0..255 liegen muss.

Betrachten wir nun ein Programm, das Bilder vertikal spiegelt:

Originalbild

Bild vertikal gespiegelt

Das Programm muss sich in unser Rahmenprogramm einfügen. Kursive Schrift bedeutet, der Code muss genau so lauten, da er durch das Rahmenprogramm bedingt ist. Fett hervorgehoben sind Kernelemente des Programms.

```
public class BildVertikalSpiegeln implements BildBearbeitungInterface {
// den Klassennamen können Sie frei vergeben

        @Override
        public Picture bearbeite(Picture originalBild) {
        // das heisst, diese Methode erhält ein Picture als Eingabewert
        // und soll ein neues Picture zurückgeben

                int breite = originalBild.getWidth();
                int hoehe = originalBild.getHeight();
                Picture neuesBild = new Picture(breite, hoehe);
                // erstelle ein neues Bild mit gleichen Ausmassen wie Original
                // dieses neue Bild wird unten von unserer Methode zurückgegeben
                // zu Beginn sind alle Pixel schwarz (Farbwerte R=0, G=0, B=0)

                for (int y = 0; y < hoehe; y++) {
                        for (int x = 0; x < breite; x++) {
                                Pixel originalPixel = originalBild.getPixel(x, y);
                                // hole Pixel in Originalbild

                                Pixel neuerPixel = neuesBild.getPixel(x, hoehe - 1 - y);
                                // hole Pixel in neuem Bild, an gespiegelter y-Koordinate
```

```
                                kopiereFarben(originalPixel, neuerPixel);
                                // berechne die Farben des Pixels in neuem Bild
                        }
                }

                return neuesBild;
        }

        void kopiereFarben(Pixel originalPixel, Pixel neuerPixel) {
        // beim Spiegeln eines Bildes werden die Farben nicht verändert
        // daher werden die Originalwerte für die Farben einfach übernommen:

                processedPixel.setRed(neuerPixel.getRed());
                processedPixel.setGreen(neuerPixel.getGreen());
                processedPixel.setBlue(neuerPixel.getBlue());
        }

}
```

Damit Sie dieses Programm laufen lassen können, müssen Sie im Hauptprogramm in der **Klasse BildBearbeitungsProgramm** die fettmarkierten Teile anpassen:

```
final String[] bildDateiNamen = { "goldengate.jpg" };
// geben Sie hier den Namen oder die Namen der zu bearbeitenden Bilder an
// mehrere Dateien können Sie komma-getrennt angeben:
// “goldengate.jpg”, “hallodu.jpg”
// die Dateien müssen im gleichen Ordner liegen wie goldengate.jpg

final BildBearbeitungInterface[] bildBearbeiter = { new BildVertikalSpiegeln() };
// hier müssen Sie Ihre Klasse angeben, die ein Bild bearbeitet
// auch hier können Sie komma-getrennt mehrere Bild-Bearbeitungs-Klassen nennen:
// new BildVertikalSpiegeln(), new BildHorizontalSpiegeln()
```

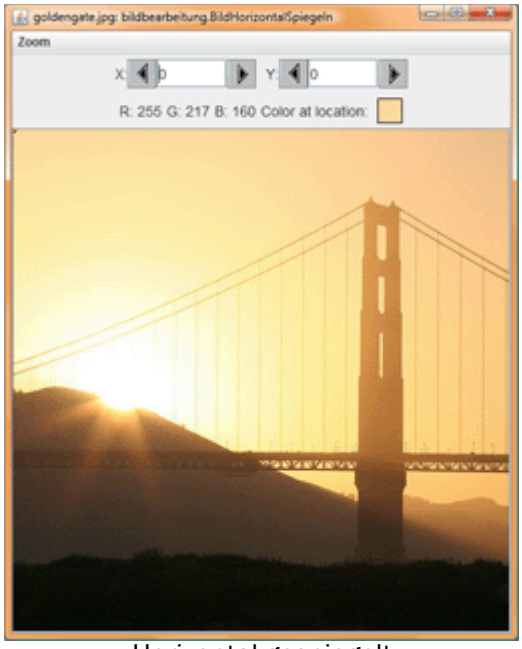
Jetzt können Sie wie gehabt mit “Run” die Klasse BildBearbeitungsProgramm starten.

Vorbereitende Aufgabe: Bild horizontal spiegeln

Diese Aufgabe wird in der Klasse gelöst. Ziel ist es, das Rahmenprogramm und die Klassen Picture und Pixel kennenzulernen.

Erstellen Sie eine Klasse, die ein Bild horizontal spiegelt:

Originalbild



Horizontal gespiegelt

Verwenden Sie dazu eine Kopie der Klasse BildVertikalSpiegeln. Denken Sie daran, diese Klasse im Hauptprogramm in der Klasse BildBearbeitungsprogramm einzufügen.

Sie können auch eigene Bilder verwenden, um Ihre Lösung zu testen.

Beispielprogramm: Farbe Rot entfernen

Farben werden in sehr vielen Programmen wie Adobe Photoshop, Microsoft Powerpoint u.v.m. häufig als Mischung von Rot-, Grün- und Blau-Bestandteilen angegeben (RGB). Jeder Farbwert liegt dabei im Bereich 0..255. Das ergibt 2^8 * 2^8 * 2^8 = 2^24 = 16'777'216 mögliche Farben.

Eine kurze Einleitung zu RGB finden Sie bei Wikipedia:
http://de.wikipedia.org/wiki/RGB-Farbraum

Zwei Online-Programme, um Farben zu mischen, finden Sie unter:
http://www.wackerart.de/mixer.html
http://www.kurztutorial.info/programme/farbenrechner/cmyk-rgb.htm

Betrachten wir ein Programm, das die Farbe rot aus einem Bild entfernt:

Orginalbild

Rot entfernt

Das Programm lässt die Bildstruktur unverändert: Die Koordinaten werden nicht wie beim Programm Bild spiegeln neu berechnet, sondern unverändert übernommen. Auf jedem Pixel werden die Farbwerte von Grün und Blau übernommen, aber Rot wird immer auf 0 gesetzt:

```
public class BildRotEntfernen implements BildBearbeitungInterface {

        @Override
        public Picture bearbeite(Picture originalBild) {
                int breite = originalBild.getWidth();
                int hoehe = originalBild.getHeight();
                Picture neuesBild = new Picture(breite, hoehe);

                for (int y = 0; y < hoehe; y++) {
```

```
                        for (int x = 0; x < breite; x++) {
                                Pixel originalPixel = originalBild.getPixel(x, y);
                                Pixel neuerPixel = neuesBild.getPixel(x, y);
                                bearbeitePixel(originalPixel , neuerPixel);
                        }
                }

                return neuesBild;
        }

        void bearbeitePixel(Pixel originalPixel , Pixel neuerPixel) {
                neuerPixel.setRed(0);
                neuerPixel.setGreen(originalPixel .getGreen());
                neuerPixel.setBlue(originalPixel.getBlue());
        }

}
```

Die zugehörige Anpassung im Hauptprogramm in BildBearbeitungsProgramm:

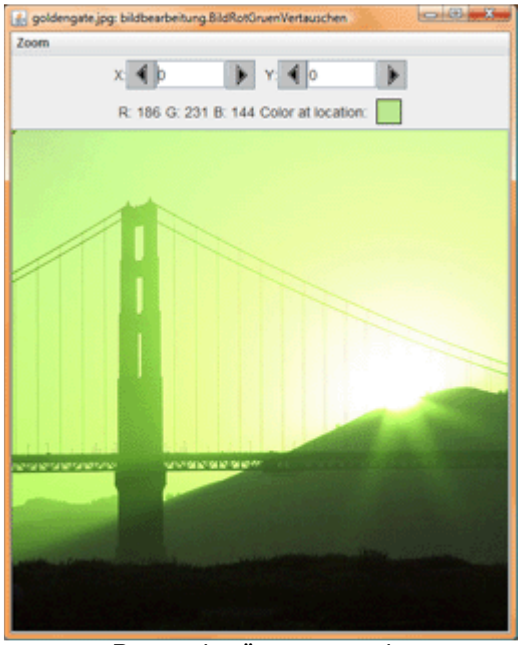
```
final BildBearbeitungInterface[] bildBearbeiter = {
  new BildVertikalSpiegeln(),
  new BildRotEntfernen()
};
```

Vorbereitende Aufgabe: Rot und Grün vertauschen

Diese Aufgabe wird in der Klasse gelöst. Ziel ist es, das Rahmenprogramm und die Klassen Picture und Pixel kennenzulernen.

Schreiben Sie ein Programm, das für jeden Bildpunkt die Farben Rot und Grün vertauscht (oder ähnliche Farbveränderungen vornimmt):

Originalbild



Rot und grün vertauscht

Verwenden Sie dazu eine Kopie der Klasse BildRotSpiegeln. Denken Sie daran, diese Klasse im Hauptprogramm in der Klasse BildBearbeitungsprogramm einzufügen.

Sie können auch eigene Bilder verwenden, um Ihre Lösung zu testen.

Bildbearbeitung: Aufgaben

Lösen Sie insgesamt drei Aufgaben der fünf Aufgaben:

> Lösen Sie **Aufgabe 1** und **Aufgabe 2**.

> Lösen Sie **eine der Aufgaben 3, 4, 5.**

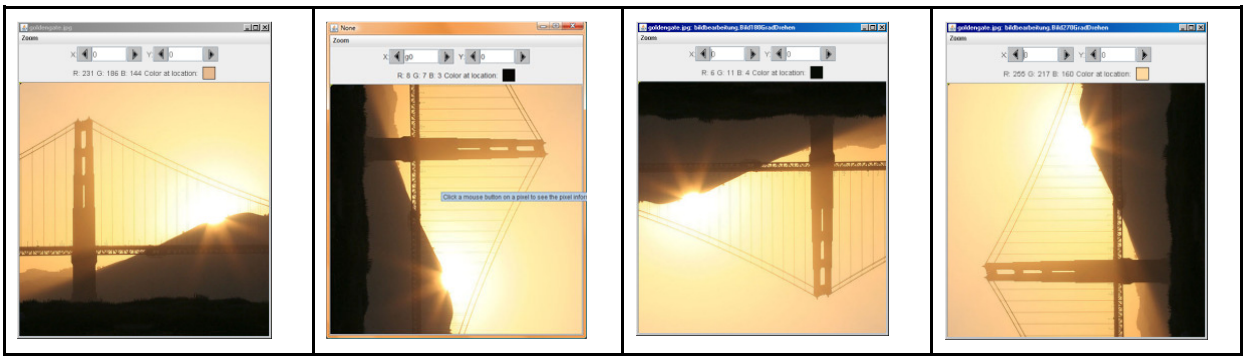
Aufgabe 1: Bild um 90°, 180° und 270° drehen

Schreiben Sie drei Bildbearbeitungsprogramme:

1. Drehung um 90° nach rechts [1 Punkt]

2. Drehung um 180° (entspricht Punktspiegelung am Zentrum) [3 Punkte]

3. Drehung um 270° nach rechts (entspricht Drehung um 90° nach links) [1 Punkt]



Das Hauptprogramm in der Klasse BildBearbeitungsProgramm sieht dann wie folgt aus:

```
final String[] bildDateiNamen = { "goldengate.jpg" };

final BildBearbeitungInterface[] bildBearbeiter = {
                        new Bild90GradDrehen(),
                        new Bild180GradDrehen(),
                        new Bild270GradDrehen() };
```

Selbstverständlich können Sie auch andere Bilder verwenden! Achten Sie einfach darauf, dass die Bilder maximal ca. 600x600 Pixel gross sind.

Aufgabe 2: Farbveränderungen am Bild vornehmen

1. Schreiben Sie ein Programm, das die Farben eines Bildes in Graustufen umwandelt: [2 Punkte]

Originalbild

Bild in Graustufen

Hintergrundsinformationen zu Graustufen:
http://de.wikipedia.org/wiki/Graustufen

Die Formel für Umrechnung von RGB-Farben in Graustufen:
http://de.wikipedia.org/wiki/Grauwert

Sie können auch mit anderen Formeln für die Grauwerte experimentieren - je nach Bild liefert eine andere Formel anschaulichere Grauwerte.

Wenn Sie mit Farbwerten rechnen, können Ihnen folgende Hinweise hilfreich sein:

* Farbwerte sind ganzzahlig. Wenn Sie nun zum Beispiel den Rotanteil eines jeden Pixels auf 20% des Orginalwertes setzen möchten, können Sie das wie folgt berechnen:

```
int neuerRotWert = (int) (originalPixel.getRed() * 0.2);
neuerPixel.setRed(neuerRotWert);
```

Die Multiplikation von originalPixel.getRed() mit 0.2 = 20/100 liefert als Resultat eine sog. **Fliesskommazahl**. War der Rotwert des Pixels zum Beispiel 87, wäre das Resultat 17.4. Aber für den neuen Rotwert braucht es wieder eine Ganzahl. Die Angabe von (int) nach der Zuweisung mit = sorgt dafür, dass die Fliesskommazahl in eine Ganzzahl umgewandelt wird.

In diesem Beispiel könnten Sie alternativ auch schreiben:

```
int neuerRotWert = originalPixel.getRed() * 20 / 100;
neuerPixel.setRed(neuerRotWert);
```

So wird eine ganzzahlige Multiplikation und Division durchgeführt. Das Resultat ist in beiden Formeln dasselbe.

* Farbwerte sind ganze Zahlen im **Bereich 0..255**. Vielleicht verwenden Sie eine Berechnung, die einen Wert ausserhalb dieser Grenzen liefert. Sie könnten mit if-Anweisungen sicherstellen, dass der neue Wert nie unter Null oder über Null liegt. Angenommen, wir möchten den Rotanteil um 45% erhöhen, dann könnte ja der neue Rotwert über 255 sein:

```
int neuerRotWert = (originalPixel.getRed() * 145 / 100;
if (neuerRotWert > 255)  {
        neuerRotWert = 255;
}
neuerPixel.setRed(neuerRotWert);
```

Kürzer lässt sich das schreiben, wenn man eine Methode verwendet, die das Minimum zweier Werte zurückgibt:

```
int neuerRotWert = Math.min(255, originalPixel.getRed() * 145 / 100);
neuerPixel.setRed(neuerRotWert);
```

Analog kann auch Math.max(0, irgendEineBerechnungFuerNeuenFarbwert) verwendet werden, um sicherzustellen, dass nie ein negativer Farbwert berechnet wird.

2. Schreiben Sie ein zweites Programm, das die Farbwerte der einzelnen Pixel neu berechnet: [3 Punkte]

* Zum Beispiel ein Bild in Rot-, Grün-, oder Blaustufen (also nur eine Grundfarbe verwenden statt wie bei den Graustufen alle drei mit gleichem Wert).

* Zum Beispiel den Anteil einer Farbe erhöhen, den Anteil ein anderen Farbe verringern.

* Zum Beispiel Farben vertauschen, Rot zu Grün, Grün zu Blau, Blau zu Rot werden lassen.

* ...

Sie können selbst entscheiden, wie Sie die Farbwerte neu berechnen möchten!

Aufgabe 3: Bildgrösse halbieren [5 Punkte]

Schreiben Sie ein Programm, das die Grösse eines Bildes halbiert, horizontal und vertikal.

Originalbild

Bildgrösse halbiert mit Interpolation

Abstrakt dargestellt würde ein Bild mit vier Pixel reduziert auf ein Bild mit einem Pixel:

Bild voher	
Pixel 1	Pixel 2
Pixel 3	Pixel 4

Bild nachher
Pixel

Eine einfache Lösung ist dabei, jede zweite Spalte und jede zweite Zeile wegzulassen. Die Farben von “Pixel” in der Abbildung oben rechts wären dann einfach die Farben von “Pixel 1” in der Abbildung oben links. Allerdings leidet die Bildqualität erheblich darunter.

1. Schreiben Sie ein Programm, das dieses einfache Verfahren umsetzt.

Eine anspruchsvollere Lösung berechnet die Farben von “Pixel” in der Abbildung oben rechts unter Berücksichtigung von mehreren Pixeln im Orginalbild, zum Beispiel unter Berücksichtigung der vier abgebildeten Pixel oder unter Berücksichtigung aller acht Nachbarpixel berücksichtigen.

2. Finden Sie ein Verfahren, das mehrere Pixel des Originalbildes verwendet, um so die Bildqualität zu steigern. Am besten, Sie experimentieren dazu mit verschiedenen Formeln für die Berechnung der Farben der neuen Pixel.

Aufgabe 4: Farben in Abhängigkeit von Position verändern [5 Punkte]

Schreiben Sie ein Programm, das die Farbwerte der Pixel in Abhängigkeit Ihrer Position verändert.

Sie könnten zum Beispiel zum Beispiel einen Farbwert von links nach rechts ausblenden, d.h. den neuen Farbwert von 100% des Originalfarbwertes ganz links kontinuierlich auf 0% ganz rechts reduzieren:

Originalbild

Rotanteil wird nach rechts reduziert

Hilfreich für solche Berechnungen sind folgende Formeln (analoge Formeln können Sie für Y-Richtung verwenden):

```
double deltaX = orginalBild.getWidth() - orginalPixel.getX();
// wie weit ist Pixel vom rechten Rand entfernt?
double prozentDeltaX = deltaX / orginalBild.getWidth();
// Abstand des Pixel vom rechten Rand in Prozent
```

Sie könnten auch ein Bild von der Mitte her “ausdunkeln”:

Originalbild

Bild von Mitte her dunkler gemacht

Hilfreiche Formeln dazu:

```
double deltaX = Math.abs(originalBild.getWidth() / 2.0 - x);
// wie weit ist Pixel von der Mitte entfernt?
double prozentDeltaX = 1 - deltaX / (originalBild.getWidth() / 2.0);
// wie weit ist Pixel prozentual von der Mitte entfernt?
```

Sie könnten das Bild auch in horizontaler und vertikaler Richtung von der Mitte her ausdunkeln.

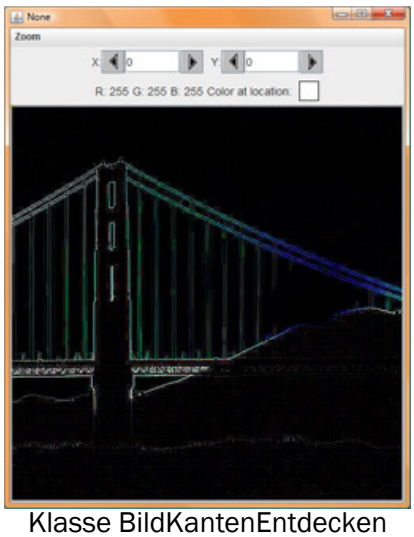
Sie könnten auch die Helligkeit eines Bilder in der Mitte am stärksten erhöhen, gegen den Bildrand hin immer weniger.

Sie könnten aber auch nur einen bestimmten Bildausschnitt verändern. Wenn Sie zum Beispiel ein Foto einer Person mit roten Augen nehmen, könnten Sie in den Bereichen der roten Augen den Rotwert reduzieren. Die Koordinaten können Sie mit Hilfe des Rahmenprogramms bestimmen.

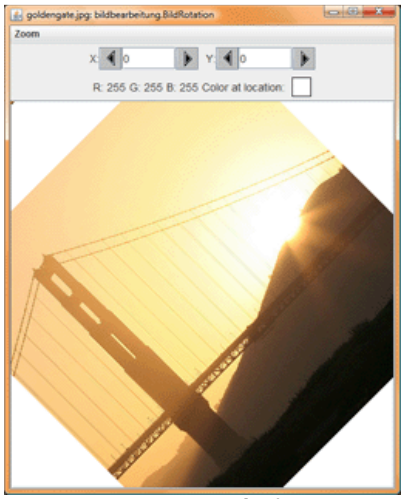
Sie können selber wählen, welchen Effekt Sie berechnen wollen!

Aufgabe 5: Kreative Bildbearbeitung [5 Punkte]

Mit dem Rahmenprogramm haben Sie einige Bildbearbeitungseffekt erhalten. Starten Sie die Klasse BildBearbeitungsProgrammAlleEffekte, um alle bereits vorgegebenen Effekte zu betrachten. Hier eine Auswahl:

<table>
<tr>
<td> 

Klasse BildKantenEntdecken</td>
<td> 

Klasse BildSwirl</td>
</tr>
<tr>
<td> 

Klasse BildRotation (45° nach links)</td>
<td> 

Klasse BilderNacheinanderBearbeiten mit den Effekten BildRotEntfernen und BildVertikalSpiegeln</td>
</tr>
</table>

Suchen Sie selber ein geeignetes Bild und schreiben Sie einen Bildbearbeitungseffekt, so dass das Resultat im weitesten Sinne visuell ansprechend ist!