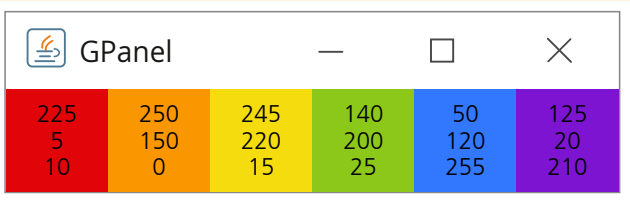
Themen

Farben in Graustufen umwandeln

Für das Umwandeln eines Farbbildes in ein Graustufenbild werden die RGB-Farbwerte verwendet, die für jedes einzelne Pixel mit Hilfe der Funktion `getPixelColor(x,y)` ❶ ermittelt werden.



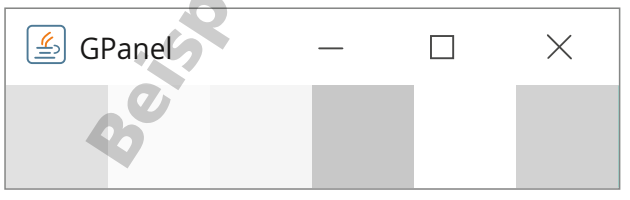
Anhand der drei Werte für rot, grün und blau wird dann ein Grauwert g ermittelt ❷, der mit der Funktion `makeColor()` zu einer Graustufe kombiniert wird ❸.

Es gibt unterschiedliche Methoden für das Ermitteln des Grauwerts. Im Beispiel wird jeweils der größte der drei RGB-Farbwerte verwendet, also 225, 250, 245 usw. ❷.

Das Ergebnis dieser besonders einfachen Methode ist ein Graustufenbild mit sehr hellen Farbflächen.

```
from gpanel import *
makeGPanel(Size(240, 40))
bild = getImage("farben.png")

graustufen = []
for x in range(240):
    for y in range(40):
        pixel = bild.getPixelColor(x,y) ❶
        rot = pixel.getRed()
        gruen = pixel.getGreen()
        blau = pixel.getBlue()
        g = max(rot, gruen, blau)  ❷
        grau = makeColor(g, g, g)  ❸
        if g not in graustufen:
            graustufen.append(g)
        bild.setPixelColor(x, y, grau)

image(bild, 0, 0)
print graustufen
```

```
[225, 250, 245, 200, 255, 210]
```



Die umgekehrte Methode – das Ermitteln des Grauwerts anhand des kleinsten der drei RGB-Farbwerte – führt entsprechend zu Graustufenbildern mit sehr dunklen Farbflächen.

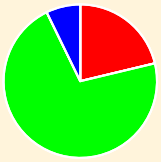
Bei einer weiteren Methode wird der Grauwert als Ganzzahl des Mittelwerts der drei RGB-Farbwerte berechnet ❺ (in RGB-Farbangaben sind nur ganze Zahlen zulässig).

Das Ergebnis ist hier ein relativ dunkles und kontrastarmes Graustufenbild.

```
g = int((rot + gruen + blau)/3)  ❺
```

```
[80, 133, 160, 121, 141, 118]
```

Da einfache Umwandlungsmethoden wie die drei genannten unnatürliche Graustufenbilder liefern, wird für das hochauflösende Fernsehen (HDTV) eine Formel genutzt, in der die drei Farbwerte unterschiedlich stark berücksichtigt werden:
grau = 0,2126 · rot + 0,7152 · grün + 0,0722 · blau.



Wie unterschiedlich die Gewichtung der Farben bei der Berechnung des Grauwertes ist, zeigt die Tortengrafik.

Die Formel berücksichtigt nicht nur die reinen Farbwerte, sondern auch die unterschiedliche Helligkeit der Farben, die das menschliche Auge besser erkennen kann als kleine Farbabweichungen.

Das Ergebnis der Formel sind plastische Graustufenbilder, die dank einer größeren Helligkeitsspanne deutlich kontrastreicher sind.

```
g = int(0.2126 * rot + 0.7152 * gruen
+ 0.0722 * blau)
```

```
[52, 160, 210, 174, 114, 56]
```

Farben in Graustufen umwandeln

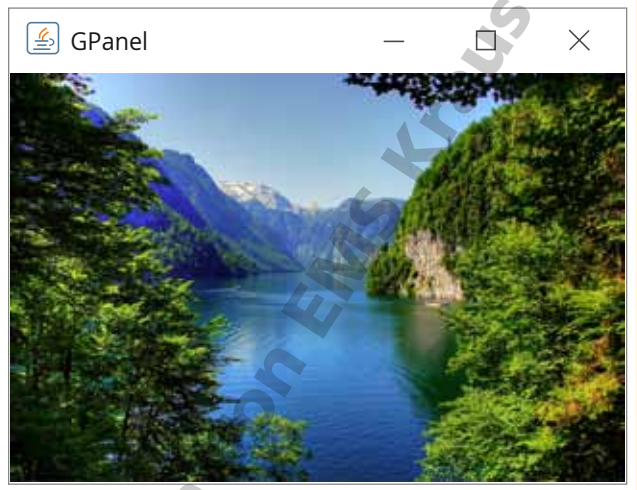
Aufgabe 1

Erstelle ein Programm, das

a) die Anzahl der unterschiedlichen Farben im Bild koenigssee.png ermittelt und ausgibt,

b) das Bild mit Hilfe der Formel für das hochauflösende Fernsehen (HDTV) in ein Graustufenbild umwandelt,

c) die Anzahl der unterschiedlichen Graustufen im umgewandelten Bild ermittelt und ausgibt



Beispiellösung

```
from gpanel import *
makeGPanel(Size(240, 160))
bild = getImage("bilder/koenigssee.png")
image(bild, 0, 0)

farben = []
graustufen = []
for x in range(240):
    for y in range(160):
        pixel = bild.getPixelColor(x, y)
        rot = pixel.getRed()
        gruen = pixel.getGreen()
        blau = pixel.getBlue()
        farbePixel = [rot, gruen, blau]
        if farbePixel not in farben:
            farben.append(farbePixel)
        g = int(0.2126 * rot + 0.7152 * gruen + 0.0722 * blau)
        grau = makeColor(g, g, g)
        if g not in graustufen:
            graustufen.append(g)

print "Anzahl Farben:",len(farben)
print "Anzahl Graustufen:",len(graustufen)
```

Anzahl Farben: 31439

Anzahl Graustufen: 237

Aufgabe 2

a) Wie viele unterschiedliche Farben sind im RGB-Farbraum möglich?

b) Wie viele unterschiedliche Graustufen sind in einem Graustufenbild möglich?

a) Der RGB-Farbraum entsteht durch die Mischung der drei Grundfarben Rot, Grün und Blau, die jeweils in 256 Farbtönen auftreten können. Daraus ergeben sich 256 x 256 x 256 = 16 777 216 Farben.

b) Graustufenbilder enthalten für Rot, Grün und Blau denselben Wert. Dadurch sind nur 256 unterschiedliche Werte, also 256 unterschiedliche Graustufen möglich.

Farben in Graustufen umwandeln

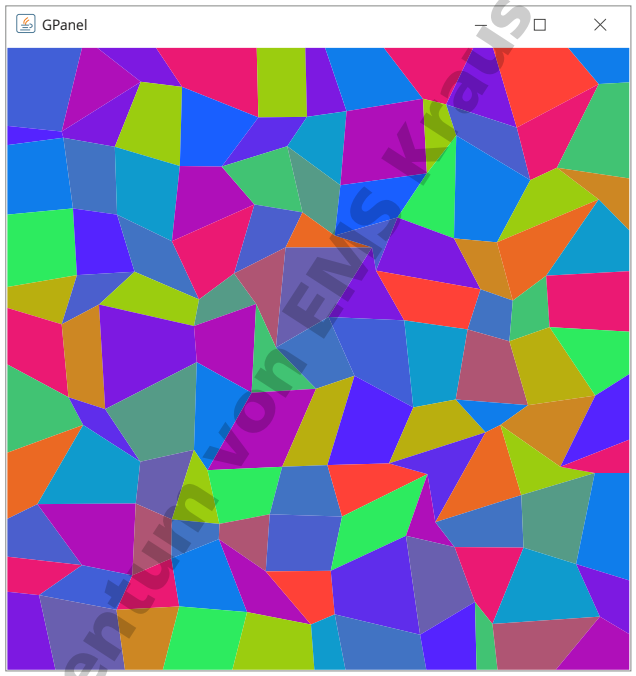
Aufgabe 3

a) Erstelle ein Programm, das das Bild polygon.png mit der Mittelwertmethode in ein Graustufenbild umwandelt.

b) Was fällt dir auf? Erkläre deine Beobachtung.

Beispiellösung

```
from gpanel import *
makeGPanel(Size(500,500))
bild = getImage("bilder/polygon.png")
image(bild, 0, 0)

for x in range(500):
    for y in range(500):
        pixel = bild.getPixelColor(x,y)
        rot = pixel.getRed()
        gruen = pixel.getGreen()
        blau = pixel.getBlue()
        g = int((rot + gruen + blau)/3)
        grau = makeColor(g, g, g)
        bild.setPixelColor(x,y,grau)
image(bild, 0, 0)
```



Alle Farben des Polygons enthalten Farbwerte, deren Mittelwert denselben Wert ergibt. Dadurch verwandelt sich das Bild bei der Umwandlung in Graustufen in eine homogene graue Fläche.

Aufgabe 4

Notiere für die vier Methoden der Graustufenumwandlung jeweils drei Farben mit ihren Farbwerten, die nach der Umwandlung zu demselben Grauton werden.

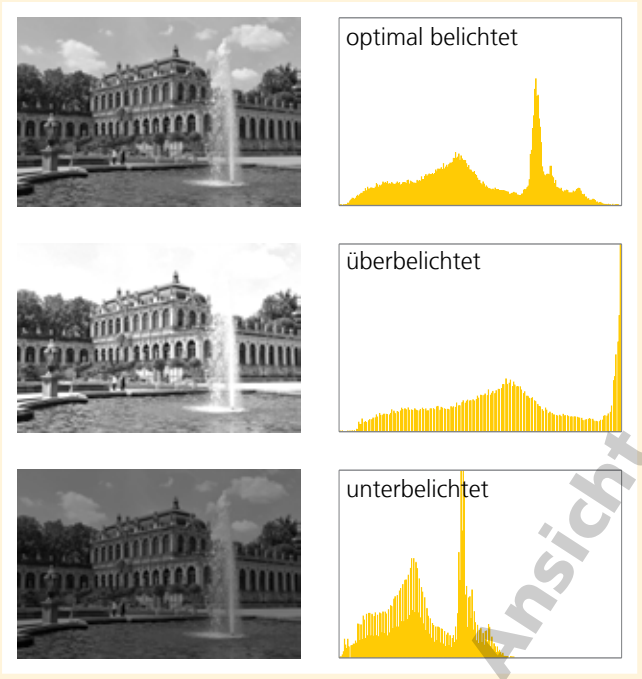
Beispiellösung

Methode		rot	grün	blau	grau
kleinster Farbwert	1	150	255	195	150
	2	165	150	235	150
	3	240	200	150	150
größter Farbwert	1	25	80	150	150
	2	150	35	0	150
	3	140	150	90	150
Mittelwert	1	65	255	130	150
	2	195	0	255	150
	3	240	115	95	150
Formel HDTV	1	255	110	240	150
	2	50	195	0	150
	3	130	145	255	150

Histogramme

Um die Belichtung von Fotos zu beurteilen, nutzen Profis so genannte Histogramme. Darin ist die Häufigkeit, mit der die einzelnen Graustufen in einem Foto vorkommen, als Balkendiagramm dargestellt, vom Grauwert 0 (schwarz) ganz links bis zum Grauwert 255 (weiß) ganz rechts. Je mehr Pixel einer Graustufe es gibt, desto höher ist der jeweilige Balken.

Ist ein Foto ausgewogen belichtet, verteilen sich die Balken über die gesamte Breite, berühren aber weder links noch rechts den Rand. Das Foto enthält also keine schwarzen und rein weißen Pixel.



Das Histogramm eines überbelichteten Fotos liegt am rechten Rand an. Ein solches Foto enthält zahlreiche rein weiße Pixel. Bei einem unterbelichteten Foto sammeln sich die Balken hingegen im linken Bereich des Histogramms. Dunkle Bereiche des Fotos können auch schwarze Pixel enthalten.

Die in weißen und schwarzen Bereichen eines Bildes vorhandenen Details lassen sich auch durch nachträgliches Retuschieren nicht mehr zum Vorschein bringen.

Möchte man ein Histogramm für ein Graustufenbild erstellen, muss man zunächst eine Liste schreiben, die für jedes einzelne Pixel den Grauwert enthält. Anschließend wird gezählt, wie häufig die einzelnen Graustufen von 0 bis 255 in dieser Liste vorkommen.

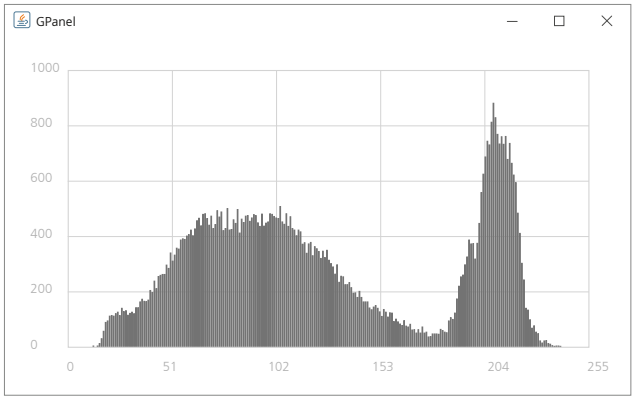
```
#Häufigkeit Graustufen zählen
histogramm = []
zaehler = 0
for n in range(256):
    for i in range(len(graustufen)):
        if graustufen[i] == n:
            zaehler = zaehler + 1
    haeufigkeit = [n,zaehler]
    histogramm.append(haeufigkeit)
    zaehler = 0
```

Das Ergebnis ist eine mehrdimensionale Liste – in unserem Beispiel heißt sie **histogramm** –, die in jeder der 256 Teillisten einen Grauwert mit seiner Häufigkeit enthält.

Diese Liste wird verwendet, um das Histogramm zu zeichnen. Im Grafikfenster wird dafür mit der Funktion **drawGrid()** ein Koordinatensystem gezeichnet. In unserem Beispiel reicht die x-Achse des Koordinatensystems von 0 bis 255, die y-Achse von 0 bis 1000, beide Achsen enthalten fünf Abschnitte, alles ist silbergrau gefärbt.

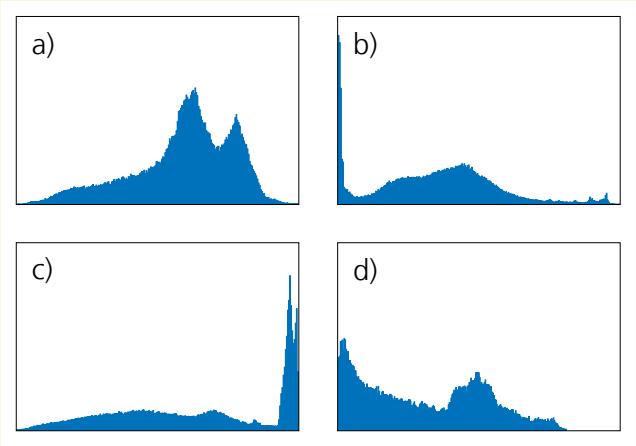
Die 256 Balken des Histogramms werden als Rechtecke in das Koordinatensystem gezeichnet. Die linke untere Ecke jedes einzelnen Rechtecks befindet sich bei x = n / y = 0. Der y-Wert der rechten oberen Ecke ist der zweite Wert der jeweiligen Teilliste aus der mehrdimensionalen Liste **histogramm**.

```
#Histogramm zeichnen
makeGPanel(-30,275,-100,1100)
drawGrid(0,255,0,1000,5,5,"silver")

setColor("DimGray")
for n in range(256):
    fillRectangle(n,0,n+1,histogramm[n][1])
```



Histogramme

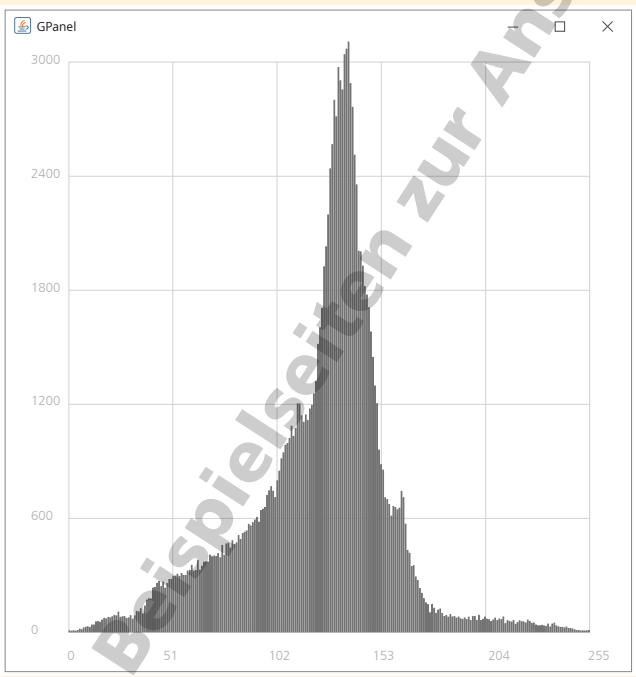
Aufgabe 1
Notiere für die folgenden vier Histogramme, ob die Fotos, zu denen sie gehören, unterbelichtet, ausgewogen belichtet oder überbelichtet sind.



a) ausgewogen belichtet
b) unterbelichtet
c) überbelichtet
d) unterbelichtet

Aufgabe 2
Erstelle ein Programm, das für das Bild moewen.png ein Histogramm zeichnet.

Probiere aus, welche Parameter du für die Höhen von Grafikfenster und Koordinatensystem einstellen musst, damit die Häufigkeitsbalken des Histogramms ins Koordinatensystem passen.



Beispiellösung

```
from gpanel import *

#Grauwerte Bild einlesen
bild = getImage("bilder/moewen.png")
graustufen = []
for x in range(450):
    for y in range(300):
        pixel = bild.getPixelColor(x,y)
        grau = pixel.getRed()
        graustufen.append(grau)

#Häufigkeit Graustufen zählen
histogr = []
zaehler = 0
for n in range(256):
    for i in range(len(graustufen)):
        if graustufen[i] == n:
            zaehler = zaehler + 1
    haeufigkeit = [n,zaehler]
    histogr.append(haeufigkeit)
    zaehler = 0

#Histogramm zeichnen
makeGPanel(-30,275,-200,3100)
drawGrid(0,255,0,3000,5,5,"silver")

setColor("DimGray")
for n in range(256):
    fillRectangle(n,0,n+1,histogr[n][1])
```

Histogramme

Aufgabe 3
Ermittle im Bild fachwerk.png die überbelichteten Bereiche, in denen sich weiße Pixel befinden, und färbe sie blau ein.

Beispiellösung

```
from gpanel import *
makeGPanel(Size(400, 620))
window(0, 400, 0, 620)
bild = getImage("bilder/fachwerk.png")
image(bild, 0, 320)

for x in range(400):
    for y in range(300):
        pixel = bild.getPixelColor(x,y)
        g = pixel.getRed()
        if g == 255:
            farbe = makeColor(0,114,188)
        else:
            farbe = makeColor(g,g,g)
        bild.setPixelColor(x,y,farbe)
image(bild, 0, 0)
```

Aufgabe 4
Ermittle im Bild burg.png die unterbelichteten Bereiche, in denen sich schwarze Pixel befinden, und färbe sie gelb ein.

Beispiellösung

```
from gpanel import *
makeGPanel(Size(450, 620))
window(0, 450, 0, 620)
bild = getImage("bilder/burg.png")
image(bild, 0, 320)

for x in range(450):
    for y in range(300):
        pixel = bild.getPixelColor(x,y)
        g = pixel.getRed()
        if g == 0:
            farbe = makeColor(255,204,0)
        else:
            farbe = makeColor(g,g,g)
        bild.setPixelColor(x,y,farbe)
image(bild, 0, 0)
```

Histogramme

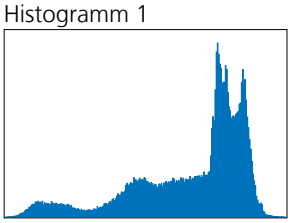
Aufgabe 5

Ordne diese vier Bilder und Histogramme paarweise einander zu.

Überprüfe deine Zuordnung, indem du mit dem Programm aus Aufgabe 2 Histogramme für die vier Bilder erstellst.

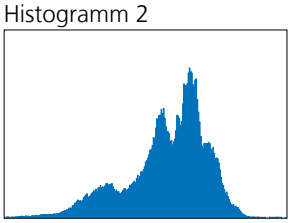
valencia.png

Histogramm 1



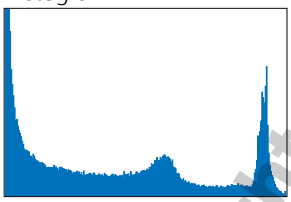
fluss.png

Histogramm 2



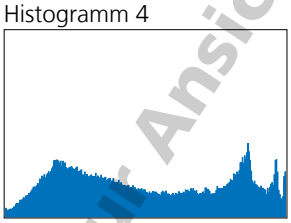
taj-mahal.png

Histogramm 3



schottland.png

Histogramm 4



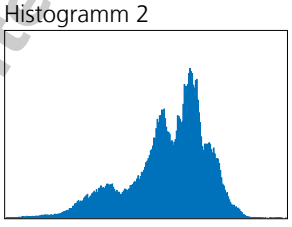
Beispiellösung

```
from gpanel import *

#Grauwerte Bild einlesen
bild = getImage("bilder/valencia.png")
#bild = getImage("bilder/fluss.png")
#bild = getImage("bilder/taj-mahal.png")
#bild = getImage("bilder/schottland.png")
graustufen = []
for x in range(450):
    for y in range(300):
        pixel = bild.getPixelColor(x,y)
        grau = pixel.getRed()
        graustufen.append(grau)

#Häufigkeit Graustufen zählen
histogr = []
zaehler = 0
for n in range(256):
    for i in range(len(graustufen)):
        if graustufen[i] == n:
            zaehler = zaehler + 1
    haeufigkeit = [n,zaehler]
    histogr.append(haeufigkeit)
    zaehler = 0

#Histogramm zeichnen
makeGPanel(-30,275,-200,3100)
drawGrid(0,255,0,3000,5,5,"silver")

setColor("DimGray")
for n in range(256):
    fillRectangle(n,0,n+1,histogr[n][1])
```

Richtige Zuordnung der Histogramme zu den Fotos

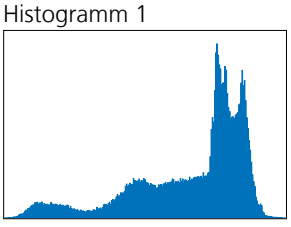
valencia.png

Histogramm 2



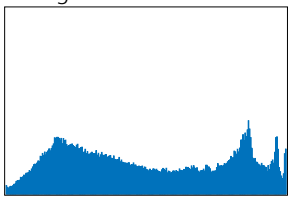
taj-mahal.png

Histogramm 1



fluss.png

Histogramm 4



schottland.png

Histogramm 3