



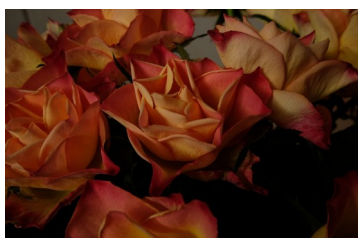
## Bildkombinationen

In dieser Erweiterung geht es darum, zwei Bilder auf geeignete Weise zu kombinieren. Man kann die Bilder addieren, voneinander subtrahieren oder miteinander multiplizieren.

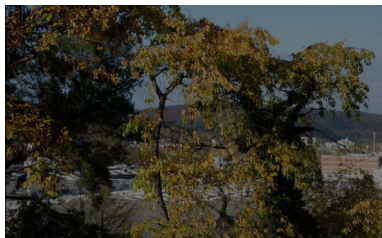
### Addition

Bei der Addition zweier Bilder werden die Intensitäten der Pixel addiert. Dadurch entsteht eine Überlagerung zweier Bilder. Da jede Farbkomponente einen Wert zwischen 0 und 255 haben kann, kommen am Ende Werte zwischen 0 und 510 heraus. Alle Werte über 255 müssen auf 255 abgeschnitten werden.

Bei der Addition zweier normaler Bilder kommt aufgrund dieses Effekts ein Bild heraus, das insgesamt zu hell ist und sehr viele weiße Pixel hat. Denkbar wäre es, die Helligkeit der Bilder zunächst zu reduzieren.



+



### Aufgaben

1. *Erstelle eine Klasse `Bildkombinationen`. Implementiere dort eine Methode `public Picture addition(Picture bild1, Picture bild2)`, die für jedes Pixel des neuen Bildes die Intensitäten der jeweiligen Farbkomponente der beiden Bilder addiert. Gehe zunächst davon aus, dass die Bilder gleich groß sind. Beachte, dass nach der Addition der Intensitäten der Bereich 0-255 einzuhalten ist. Reduziere alle Werte größer 255 auf den Wert 255. Teste deine Methode mit geeigneten Bildern.*
2. *Falls die Bilder nicht gleich groß sind, kommt es zu Fehlern. Nimmt man als Bildgröße für das neue Bild die minimale Breite und die minimale Höhe, löst man dieses Problem. Passe deine Implementation entsprechend an. Hinweis: Mit der Methode `Math.min(zahl1, zahl2)` kann man das Minimum zweier Zahlen bestimmen.*
3. *Füge dem Datei-Menü einen neuen Menüpunkt "Bild addieren ..." hinzu. Kopiere aus der Action-Methode für das Öffnen von Bildern die Befehle, um einen Datei-Öffnen-Dialog anzuzeigen und das ausgewählte Bild als `Picture`-Objekt zu öffnen. Addiere dieses Bild zum aktuell angezeigten Bild und zeige das Ergebnis im viewer als neues Bild an.*



## Subtraktion

Bei der Subtraktion von Bildern werden die Intensitäten voneinander subtrahiert. Durch die Differenzbildung der Intensitäten kann das Ergebnis negativ werden. Negative Werte werden zu Null, um im Definitionsbereich (0-255) zu bleiben.

Dabei ist es entscheidend, welches Bild von welchem abgezogen wird (siehe Abbildung: links unten = links oben - rechts oben; rechts unten = rechts oben - links oben).

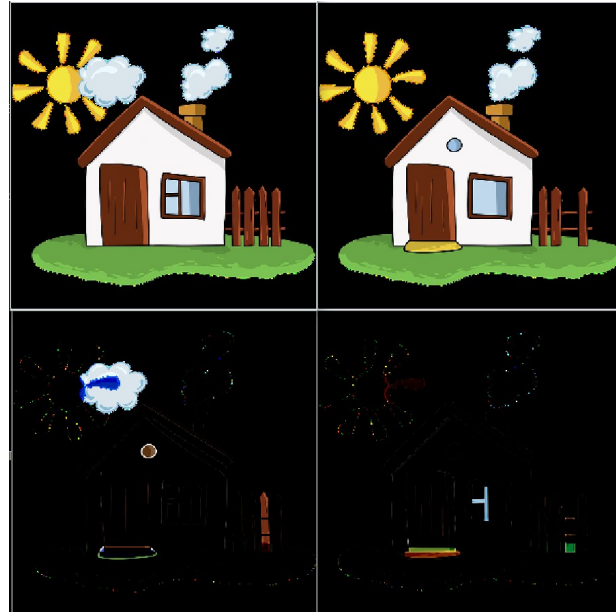


Abb. 1: Differenzbildung bei „Finde den Unterschied“, creozavr (Pixabay Lizenz) via Pixabay, URL: <https://pixabay.com/de/illustrations/haus-häuser-finde-den-unterschied-3208132/> (17.11.19)

## Aufgaben

### 4. Implementiere eine Methode `public`



`Picture` `subtraktion(Picture bild1, Picture bild2)` in der Klasse `Bildkombinationen`, die für jeden Pixel des neuen Bildes die Intensitäten der jeweiligen Farbkomponente der beiden Bilder subtrahiert. Gehe zunächst davon aus, dass die Bilder gleich groß sind. Beachte, dass nach der Subtraktion der Intensitäten der Bereich 0-255 einzuhalten ist. Ändere alle negativen Werte auf den Wert 0. Teste deine Methode mit den Bildern "unterschiedfinden1.png" und "unterschiedfinden2.png". Vertausche die Reihenfolge der Bilder.

### 5. Falls die Bilder nicht gleich groß sind, kommt es zu Fehlern. Nimmt man als Bildgröße für das neue Bild die minimale Breite und die minimale Höhe, löst man dieses Problem. Passe deine Implementation entsprechend an.



Hinweis: Mit der Methode `Math.min(zahl1, zahl2)` kann man das Minimum zweier Zahlen bestimmen.

### 6. Füge dem Datei-Menü einen neuen Menüpunkt "Unterschiede finden ..." hinzu. Kopiere aus der Action-Methode für das Öffnen von Bildern die Befehle, um einen Datei-Öffnen-Dialog anzuzeigen und das ausgewählte Bild als `Picture`-Objekt zu öffnen (`bild1`). Subtrahiere dieses Bild vom aktuell angezeigten Bild (`bild2`) und speichere das Ergebnis als `diff1`. Subtrahiere dann `bild2` von `bild1` und speichere das Ergebnis als `diff2`. Addiere `diff1` und `diff2` und zeige das Ergebnis im viewer an.



Teste deine Methode mit den Bildern "baum1.jpg" und "baum2.jpg". Beide Bilder wurden im Abstand von wenigen Millisekunden aufgenommen. Was kann man nun erkennen? Man kann den Effekt noch deutlicher machen, wenn man durch lineare Histogrammanpassung den Bereich auf 40 bis 100 einschränkt. Wie könnte man diese Operation für das autonome Fahren nutzen?

### 7. Öffne die Datei "zonenplatte\_cosinus.png". Wende mehrmals die Operation



"Weichzeichnen" an, bis die dünnen Linien außen verschwinden. Weichzeichnen wird auch als "Low-Pass-Filter" bezeichnet, da schnelle Helligkeitsänderungen (dünne Linien) im Bild verschwinden und langsame Änderungen (dicke Linien) erhalten bleiben (passieren können). Speichere das Ergebnis als neue Datei.

Wende "Unterschiede finden ..." auf das Originalbild mit dem neuen Bild an. Beschreibe, was man nun sieht. Addiere zu diesem Bild das Originalbild. Beschreibe den Effekt, den alle diese Operationen zusammengekommen auf das Originalbild haben.





## Multiplikation

Bei der Multiplikation von zwei Bildern werden die Intensitäten multipliziert. Durch die Multiplikation entstehen Werte zwischen 0 und  $255^2 = 65025$ . Sinnvoller wäre hier eine Normierung der Intensitätswerte vor der Multiplikation auf einen Wertebereich von 0 bis 1. Dadurch bleibt der Wertebereich auch nach der Multiplikation erhalten. Danach wird der Bereich wieder auf 0-255 erweitert.

$$I_{\text{neu}} = \frac{I_{\text{Bild 1}}}{255} \cdot \frac{I_{\text{Bild 2}}}{255} \cdot 255 = I_{\text{Bild 1}} \cdot I_{\text{Bild 2}} / 255$$

## Aufgaben

8.  **Implementiere eine Methode** `public Picture multiplikation(Picture bild1, Picture bild2)` **in der Klasse** `Bildkombinationen`, **die für jedes Pixel des neuen Bildes die Intensitäten der jeweiligen Farbkomponente der beiden Bilder multipliziert. Gehe zunächst davon aus, dass die Bilder gleich groß sind. Beachte, dass nach der Multiplikation der Intensitäten der Bereich 0-255 einzuhalten ist. Wende daher die oben genannte Formel an. Teste deine Methode mit den Bildern "katze.jpg" und "maske1.jpg" bzw. "maske2.jpg". Erstelle mit einem Bildbearbeitungsprogramm eine eigene Maske. Erkläre, wie eine Maske funktioniert.**
9.  **Falls die Bilder nicht gleich groß sind, kommt es zu Fehlern. Nimmt man als Bildgröße für das neue Bild die minimale Breite und die minimale Höhe, löst man dieses Problem. Passe deine Implementation entsprechend an. Hinweis: Mit der Methode `Math.min(zahl1, zahl2)` kann man das Minimum zweier Zahlen bestimmen.**
10.  **Füge dem Kunst-Menü die neuen Menüpunkte "Helligkeitsverlauf" und "Runder Ausschnitt" hinzu. Wende dazu die entsprechende Maske auf das aktuelle Bild an, um den Effekt zu realisieren. Lade dazu die Maske zunächst in ein Picture-Objekt. Beachte, dass beim Dateinamen das Unterverzeichnis mit angegeben werden muss:**  
`Picture bild2 = new Picture("images/maske1.jpg");`
11.  **Wende eine Maske auf ein Bild an. Wende danach "Unterschiede finden ..." an, um Unterschiede zwischen dem Originalbild und dem veränderten Bild zu finden. Welchen Effekt hat das?**
12.  **Die Masken haben eine feste Größe, die nicht unbedingt zum angezeigten Bild passt. Erzeuge erst beim Aufruf der Action-Methode eine entsprechende Maske der richtigen Größe. Erstelle dazu zunächst ein Picture-Objekt der richtigen Größe und zeichne darauf einen Farbverlauf mit Hilfe von Linien oder Kreisen.**