# Übung 2.1

## Lernziele

* Farbbilder einlesen und Kanäle interpretieren
* Operationen auf einzelnen Kanälen
* Umrechnung int/uint8/float
* Umrechnung von Farb- zu Graubilder

## Schritte

* Letzte Woche haben wir schon kennengelernt, wie man Bilder einliest. Mit dem gleichen Befehl können wir jetzt auch Farbbilder einlesen.
* Zeigen, dass RGB-Bilder aus drei Kanälen bestehen
* RGB-Bild visualisieren und zeigen, dass es nicht mehr uint8, sondern float ist
* Einzelne Kanäle ausblenden
* Einzelne Kanäle verstärken
* RGBA-Bild einlesen 🡪 Was ist der 4. Kanal?
* Wir fassen also zusammen: Wir haben gesehen, wie man Farbbilder einliest und auf die jeweiligen Kanäle zugreift. Wir haben gesehen, dass es RGB- und RGBA-Bilder gibt. Ausserdem haben wir kennengelernt, dass Pixel nicht nur mit 8-bit uint8, sondern auch 32-bit float32 kodiert sein können und dann eine andere Skala einnehmen.
* Uint8 haben den grössten Vorteil, dass sie weniger Speicher einnehmen. Das heisst aber nicht, dass *uint8* die Lösung für alles sind. Wir schauen uns jetzt an, was das Problem mit *uint8* Variablen sind.
* Graubild-Konvertierung Formel zeigen und besprechen, was da das Problem ist

## Aufgaben

* Programmieren Sie eine Funktion, die ein Bild mit float32 Kodierung (0 bis 1) zu einem Bild mit uint8 Kodierung (0 bis 255) umwandelt.
* Lesen Sie das Bild “Bunte\_Formen.png” ein und geben Sie sie in Ihre soeben geschriebene Funktion.
* Erstellen Sie eine Funktion “rgb\_to\_grayscale”, die ein RGB- oder RGBA-Bild mit uint8 Kodierung als Input nimmt und als Graubild ausgibt (float oder uint8). Hinweis: Beachten Sie die Limitierung von uint8 Bildern. Geben Sie dann das konvertierte uint8 Bild in die Funktion.

# Übung 2.2

## Lernziele

* Kennwerte interpretieren (Mittelwert, Varianz)
* Kontrast im Histogramm erhöhen
* Kontrast im Bild erhöhen
* Rauschen mit Mittelung reduzieren
* (Weissabgleich?)

## Schritte

* Funktion schreiben, die Mittelwert und Varianz ausgibt
* Bilder einlesen und Werte ausgeben lassen
* Bilder plotten und von Studierenden zuordnen lassen
* Entropie programmieren
* Histogramm aufmachen, Screenshot und dann Punktoperationsfunktion zeichnen (helle Sachen mehr Kontrast)
* Kontrastverstärkung umsetzen

## Aufgaben

* Laden Sie das Bild “cameraman.bmp” als Graubild ein und ploten Sie das Histogramm. Ermitteln Sie den Wertebereich für den Mantel des Kameramanns.
* Erhöhen Sie den Kontrast spezifisch nur für diesen Wertebereich mittels einer Fensterung.
* Laden Sie die Bilder “Saturn\_1.jpg”, “Saturn\_2.jpg” und “Saturn\_3.jpg” ein und entfernen Sie das Rauschen, indem sie das durchschnittliche Bild aus den drei Bildern berechnen.

## Bonus

* Laden Sie das Bild «Saturn\_RGB» ein und wenden Sie einen Weissabgleich auf das Bild an. Benutzen sie als Referenz-Pixel Zeile 170, Spalte 340.
  + Schritt 1: RGB-Werte des Referenz-Pixels, was hinterher maximal weiss sein soll, extrahieren.
  + Schritt 2: Für jeden Kanal den individuellen Korrekturfaktor berechen. Bsp: Referenz-Pixel hat Rot-Wert von 250. Weiss hat einen Wert von 255. Daher muss der Faktor 255/250 sein. Das heisst, alle Pixel müssen mit diesem Faktor multipliziert werden.
  + Schritt 3: Ein leeres Bild der gleichen Grösse wie das Original-Bild erstellen.
  + Schritt 4: Durch das Original-Bild iterieren (doppelte for-Schleife) und bei jedem Pixel die korrigierten Pixelwerte berechnen und in das leere Bild eintragen.
  + Schritt 5: Das neue Bild in uint8 Kodierung umwandeln.