

## Algorithmen zur Manipulation von Bildern

Neben dem Zeichnen verschiedener Formen gehört zu den typischen Funktionen von Bildbearbeitungsprogrammen die Manipulation von Bildern, insbesondere Fotos, in Hinblick auf Helligkeit und Kontrast, das Verändern zu einem Graustufenbild, das Entfernen eines Rotstichs, das Entfernen oder Umfärben eines Bildbereichs usw. Ziel dieses Arbeitsblattes ist es, für einige dieser Funktionen geeignete Algorithmen zu entwerfen.

Entsprechende Algorithmen basieren darauf, den RGB-Wert jedes einzelnen Pixels des Bildes mithilfe geeigneter Rechenoperationen passend zu verändern. Vielleicht haben Sie in Snap! <sup>1</sup> bereits den Block *RGBA bei selbst* verwendet, um einzelne RGB-Werte eines Bildes abzufragen. Mit diesem Block den RGB-Wert für jeden einzelnen Bildpunkt auszulesen und zu übermalen würde jedoch sehr lange dauern. Deshalb schauen wir uns als Grundlage für die Algorithmen zur Manipulation von Bildern an, wie ein Bild in Snap! codiert und gespeichert wird, so dass wir die Veränderungen direkter und damit effizienter vornehmen können.

### Codierung digitaler Fotos

Digitale Fotos sind Rastergrafiken, die aus vielen einzelnen, winzigen Quadraten, den Pixeln, bestehen. Jedes Pixel wird durch seinen RGB-Wert codiert.

**Aufgabe 1:** Erläutern Sie den Zusammenhang zwischen dem grob gerasterten Bild und der Zahlenmatrix in Abbildung 1.

	(255, 192, 0)	(180, 198, 231)	(180, 198, 231)	(180, 198, 231)	(180, 198, 231)
	(180, 198, 231)	(180, 198, 231)	(180, 198, 231)	(255, 0, 0)	(180, 198, 231)
	(0, 176, 80)	(180, 198, 231)	(255, 0, 0)	(255, 0, 0)	(255, 0, 0)
	(0, 176, 80)	(0, 176, 80)	(191, 143, 0)	(191, 143, 0)	(191, 143, 0)
	(131, 60, 11)	(180, 198, 231)	(191, 143, 0)	(0, 0, 0)	(191, 143, 0)
	(131, 60, 11)	(180, 198, 231)	(191, 143, 0)	(0, 0, 0)	(191, 143, 0)

Abbildung 1: Aufbau und Codierung einer Rastergrafik

<sup>1</sup> Snap! wird von der University of California, Berkeley zur Verfügung gestellt: <https://snap.berkeley.edu>

Wie Abbildung 1 zeigt können wir uns die Codierung eines Bildes also zunächst als eine Matrix von RGB-Werten vorstellen. Da der Speicher eines Rechners linear aufgebaut ist, bildet Snap! die Zeilen und Spalten eines Bildes jedoch auf eine lineare Liste ab.

### Aufgabe 2:

- Erläutern Sie anhand von Abbildung 2, wie das Bild aus Abbildung 1 in einer Liste codiert werden kann.
- Welche zusätzlichen Informationen werden benötigt, um aus der Liste in Abbildung 2 das ursprüngliche Bild in Abbildung 1 zu rekonstruieren.
- Die gesamten Daten des Bildes werden in Snap! als Liste von Listen zur Verfügung gestellt. Welche Werte würden die folgenden Blöcke für die Liste `pixels` in Abbildung 2 liefern?

Element 1 von pixels

Element 9 von pixels

Element 2 von Element 1 von pixels

Element 3 von Element 11 von pixels

- Zu jedem Bildpunkt wird zusätzlich zu dem Rot-, Grün- und Blauwert noch ein vierter Wert, der Alpha-Wert gespeichert. Recherchieren Sie, welche Bedeutung dieser Wert hat.

Abbildung 2: Abbildung eines zweidimensionalen Bildes auf eine Liste

### Auslesen und Verändern der Farbanteile eines Pixels

Um die RGB-Werte der einzelnen Pixel zu verändern, müssen wir auf die Liste mit den RGB-Werten eines Bildes zugreifen können. Das Bild, das wir bearbeiten möchten, importieren wir zunächst als Kostüm für ein Objekt oder als Hintergrund für die Bühne. Sie kennen bereits den Block *ziehe Kostüm ... an*, mit dem das Kostüm, also in diesem Fall das Bild, auf der Bühne angezeigt werden kann.

Mit dem Block *Pixel von Kostüm ...* können wir die Liste der RGB-Werte eines Kostümbildes ausgeben lassen. Wir finden den entsprechenden Block im Bereich *Aussehen* als *Breite von Kostüm aktuell* (s. Abbildung 3). Statt der Eigenschaft *Breite* wählen wir im vorderen Teil *Pixel* aus und im hinteren Teil den Namen des Kostüms, zu dem wir die Liste der RGB-Werte auslesen möchten.

Breite von Kostüm aktuell

Pixel von Kostüm portrait

Abbildung 3: Block für das Auslesen der RGB-Werte eines Bildes.

Diese Liste können wir zur weiteren Verarbeitung in einer Variablen speichern (s. Beispiel 1). Mithilfe der Listenoperationen kann dann auf die einzelnen Werte zugegriffen und es können Werte in der Liste verändert werden. Um die veränderte Liste wieder als Bild zu visualisieren, übergeben wir dem Block *ziehe Kostüm ... an* die Variable, welche die Liste mit den RGB-Werten enthält (s. Beispiel 2).

setze pixels auf Pixel von Kostüm portrait

Beispiel 1 Speichern der Liste der RGB-Werte des Kostüms portrait in der Variablen pixels:

ziehe Kostüm pixels an

Beispiel 2: Anzeigen einer Liste von RGB-Werten als Kostümbild

### Aufgabe 3:

- a) Erstellen Sie ein Skript, das die RGB-Werte aller Bildpunkte eines Kostüms invertiert (s. Abbildung 4).

#### Hinweise:

- Einen RGB-Wert zu invertieren bedeutet, jeden Farbanteil durch die Differenz zwischen 255 und dem ursprünglichen Farbanteil zu ersetzen. Aus dem RGB-Wert (30, 100, 255) wird so der invertierte RGB-Wert (225, 155, 0).
- Die Blöcke in Abbildung 5 können beim Erstellen des Skriptes hilfreich sein.
- Schließen Sie die gesamte Bearbeitung der RGB-Werte in einen Wrap-Block ein, da die Ausführung des Skriptes ansonsten sehr lange dauern kann.



Abbildung 4: Original und Bild mit invertierten Farben



Abbildung 5: Blöcke zur Implementierung von Aufgabe 6a

- b) Verändern Sie ihr Skript aus Aufgabenteil a) so, dass nur die RGB-Werte der oberen Hälfte des Bildes invertiert werden.

**Hinweis:** Statt des Blocks *für jedes Element von ...* muss hier eine geeignete Schleife aus dem Bereich *Steuerung* verwendet werden.

## Rekonstruktion ausgewählter Funktionen eines Bildbearbeitungsprogramms

Digitalkameras erzeugen in aller Regel Farbfotos. Um eine andere Wirkung zu erzielen, möchte man die Farbinformation manchmal auf Grauwerte reduzieren.

Zu einem Farbfoto das passende Graustufenbild wie in Abbildung 6 zu erzeugen, bedeutet im Prinzip, jeden einzelnen Bildpunkt auf seine Helligkeit zu reduzieren und den zur Helligkeit passenden Grauton auszuwählen. Die Helligkeit eines Farbtons lässt sich unterschiedlich definieren.



Abbildung 6: Umwandlung eines Farbbildes in ein Graustufenbild

### Aufgabe 4:

- a) Diskutieren Sie unterschiedliche Möglichkeiten, die Helligkeit eines Bildpunktes anhand des RGB-Wertes zu definieren. Ergänzend können Sie anschließend die Vorschläge in der Datei *AB1\_Aufg4\_Hinweis* betrachten.
- b) Ergänzen Sie in Ihrem Programm aus Aufgabe 3 die Möglichkeit, das Bild in ein Graustufenbild umzuwandeln. Gehen Sie dabei ggf. arbeitsteilig vor, um verschiedene Definitionen bzw. Berechnungen der Helligkeit zu implementieren und die Ergebnisse zu vergleichen.

**Aufgabe 5:** Ergänzen Sie in Ihrem Programm weitere Funktionen zur Bearbeitung eines Bildes. Hier sind einige Ideen. Ihnen fallen aber sicher auch noch andere Möglichkeiten ein!

- Verändern der Helligkeit
- Erhöhen des Kontrastes
- Entfernen eines Rotstichs
- Sepia-Effekt
- Schwarz-Weiß-Bild
- Posterisation

**Hinweise:** Überlegen Sie zunächst selbst, wie die RGB-Werte rechnerisch so verändert werden können, dass sich der gewünschte Effekt ergibt.

Wenn Sie keine Idee haben, welche Rechenoperationen notwendig sind, können Sie ...

- ... verschiedene Rechnungen ausprobieren und sich anschauen, wie sie das Bild verändern.
- ... die Hinweise in den entsprechenden Dateien *AB1\_Aufg5\_Hinweis\_...* zur Hilfe nehmen.

Das Ergebnis muss nicht perfekt sein! Für manche Funktionen eines Bildbearbeitungsprogramms werden komplexe mathematischen Funktionen verwendet, während man einen ähnlichen Effekt schon mit einfachen Rechenoperationen erreichen kann.

**Aufgabe 6:** Erläutern Sie, in wie weit man bei dem Programm, das Sie in Aufgabe 3 erstellt haben, von einem Grundgerüst für Aufgabe 4 und Aufgabe 5 sprechen kann. Gehen Sie dabei darauf ein, welche Gemeinsamkeiten und welche Unterschiede die Algorithmen zur Implementierung der unterschiedlichen Funktionen eines Bildbearbeitungsprogramms aufweisen.

Dieses Werk ist lizenziert unter einer [Creative Commons Namensnennung - Nicht-kommerziell - Weitergabe unter gleichen Bedingungen 4.0 International Lizenz](#). Von der Lizenz ausgenommen ist das InfSII-Logo.

**Bildnachweis:** Die Abbildungen wurden von der Autorin selbst erstellt. Die abgebildeten Blöcke und die Liste in Abbildung 2 wurden der Entwicklungsumgebung Snap! in der Version 7.2.5 entnommen. Snap! wird von der University of California, Berkeley zur Verfügung gestellt:

<https://snap.berkeley.edu>