Algorithmen zur Manipulation von Bildern
Einstiegsprojekt: Portrait als Baustein-Mosaik

Eine Firma, die bunte Bausteine verkauft, bietet seit einiger Zeit in ihren Geschäften die Möglichkeit, ein Selbstportrait aufzunehmen und dieses als Mosaik aus Bausteinen zusammenzusetzen. Das Foto wird dazu in eine Vorlage aus ca. 4500 Bausteinen in den Farben Weiß, Hellgrau, Dunkelgrau und Schwarz sowie einer Hintergrundfarbe (hier Gelb) umgewandelt. Abbildung 1 zeigt ein Beispiel. Die Firma behauptet, das geschehe mit ein bisschen Technologie und ganz viel Magie. Mit eurem Wissen über den Aufbau von Bildern und der Codierung von Farben seid ihr jedoch sicher in der Lage, diese Zauberei zu durchschauen und selbst ein Programm zu schreiben, das aus einem Foto eine entsprechende Vorlage erstellt. Habt ihr schon erste Ideen, wie der Zaubertrick funktioniert?

Abbildung 1: Vorlage für das Erstellen eines Baustein-Mosaiks

Grundidee

Wir verwenden die grafische Programmierumgebung Snap*!*[1], da sie uns geeignete Blöcke zur Verfügung stellt, um auf die Farbwerte der Pixel eines Bildes zuzugreifen. Das Foto, das in ein Mosaik umgewandelt werden soll, verwenden wir als Bühne. Außerdem benötigen wir ein Baustein-Objekt, das über fünf verschiedene Kostüme in den Farben des Mosaiks verfügt. Unser Algorithmus muss dann dafür sorgen, dass das Baustein-Objekt die Bühne systematisch abläuft und jeweils einen Abdruck in der passenden Steinfarbe hinterlässt.

Vorbereitung

Bevor ihr mit dem Programmieren beginnen könnt, benötigt ihr ein geeignetes Portrait als Hintergrundbild und Kostüme für das Baustein-Objekt. Erledigt deshalb als erstes mithilfe eines Bildbearbeitungsprogramms die Vorbereitungen in Aufgabe 1.

Aufgabe 1:

a) Nehmt ein Selbstportrait von euch vor einem einfarbigen Hintergrund auf. Überlegt, welche Hintergrundfarbe sich besonders gut eignet.
b) Erstellt mithilfe eines Bildbearbeitungsprogramms einen quadratischen Bildausschnitt eures Portraits. Überlegt euch sinnvolle Werte für die Breite und Höhe in Pixeln. Damit die Bearbeitung in Snap*!* nicht zu lange dauert, sollte das Bild nicht zu groß sein.
c) Stellt die Größe der Bühne in Snap*!* passend ein und importiert euer Bild als Hintergrund für die Bühne.
d) Wie groß sollen passend dazu die Bausteine sein? Entwerft mithilfe eines Bildbearbeitungsprogramms fünf gleich große, quadratische Kostüme in den Farben Weiß, Hellgrau, Dunkelgrau und Schwarz sowie der Hintergrundfarbe.
e) Importiert die Bilder als Kostüme für das Baustein-Objekt.

[1] Snap*!* wird von der University of California, Berkeley zur Verfügung gestellt: https://snap.berkeley.edu

Algorithmus zum Ablaufen des Bildes

Bevor wir anfangen das Bild zu verändern, benötigen wir einen Algorithmus für das Baustein-Objekt, so dass es die Bühne einmal vollständig abläuft. In Abbildung 2 stellt das gelbe Quadrat das Baustein-Objekt dar. Dieses muss nun jedes Feld des Gitters einmal besuchen.

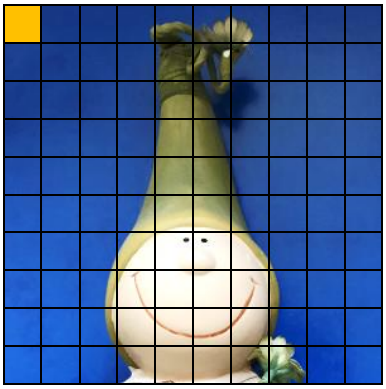

Abbildung 2: Raster für die Zerlegung eines Bildes in Bausteine

Aufgabe 2:

a) Skizziere in Abbildung 2 mithilfe von Pfeilen, in welcher Reihenfolge dein Objekt die Felder ablaufen soll.

b) Erstelle ein entsprechendes Skript für das Baustein-Objekt. Die Skizze aus Abbildung 2 kannst du verwenden, um dir benötigte Parameter des Baustein-Objektes oder der Bühne zu notieren.

Hinweis: Wenn du mit festen Werten arbeitest, reichen die dunkelblauen Blöcke aus dem Bereich *Bewegung* aus. Möchtest du dein Skript flexibel gestalten, so dass es auch für andere Größen der Bühne oder der Bausteinkostüme funktioniert, findest du im Bereich *Fühlen* weitere hilfreiche Blöcke. Einige davon zeigt Tabelle 1.

Blöcke	Mögliche Einstellungen	Bedeutung
Attribut Nachbarn	Attribut Höhe Attribut Breite	Mit diesem Block lassen sich Höhe und Breite eines Objekts herausfinden
Kostüm Nr. von	Rand links von Bühne Rand rechts von Bühne Rand oben von Bühne Rand unten von Bühne Breite von Bühne Höhe von Bühne	Mit diesem Block lassen sich die Koordinaten der Bühnenränder sowie Höhe und Breite der Bühne abfragen.

Tabelle 1: Blöcke zum Abfragen von Eigenschaften eines Objektes bzw. der Bühne

Hilfe: Wenn du nicht weiterkommst, findest du in den Dateien *AB1_Aufg2_Hinweis1, AB1_Aufg2_Hinweis2 und AB1_Aufg2_Hinweis3* schrittweise Hinweise für einen möglichen Ansatz.

Baustein-Mosaik erstellen

Anstatt das Original des Bühnenbildes zu verändern, übermalen wir es einfach mit Abdrücken eines Kostüms des Baustein-Objekts in der jeweils passenden Farbe. Das so entstandene Bild können wir dann mit einem Rechtsklick auf die Bühne abspeichern.

Um einen Abdruck eines Kostüms zu hinterlassen, muss das Objekt das entsprechende Kostüm anziehen. Anschließend kann der Block *stemple* aus dem Bereich *Stift* verwendet werden, um einen Abdruck zu hinterlassen.

Aufgabe 3: Erweitere dein Skript zunächst so, dass das Baustein-Objekt an jeder Position ein beliebiges Kostüm anzieht und einen Abdruck hinterlässt. Das Ergebnis könnte beispielsweise wie in Abbildung 3 aussehen.

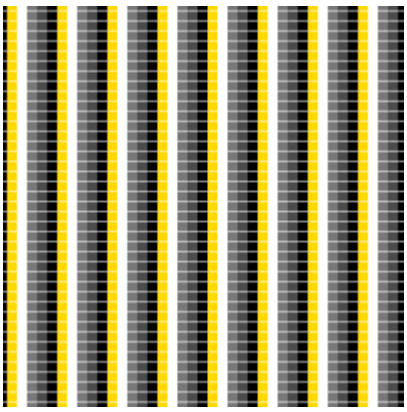

Abbildung 3: Bühne gefüllt mit Abdrücken wechselnder Baustein-Kostüme

Hinweis: Beim Hinterlassen eines Abdrucks werden die äußersten Pixel des Kostüms teilweise heller dargestellt als der Rest. Dadurch ergeben sich die hellen Streifen im Bild. Da wir die einzelnen Bausteine in der Vorlage dadurch sogar besser unterscheiden können, ist das aber unproblematisch.

Nun müssen wir nur noch ein zum Foto passendes Kostüm auswählen, um das Baustein-Mosaik von unserem Portrait zu erzeugen. Dazu ist euer Wissen über die Codierung und Darstellung von Farben im RGB-Modell gefragt.

Im Bereich *Fühlen* steht dir der Block *Farbton bei …* zur Verfügung (s. Abbildung 4). Diesen kannst du so einstellen, dass er dir eine Liste mit dem Rot-, dem Grün und dem Blauwert des Bühnenbildes an der aktuellen Position des aufrufenden Objektes zurückgibt. Kombiniere diesen Block mit dem Block *Element … von …* aus dem Bereich *Variablen* (s. Abbildung 5), um auf den Rot-, den Grün- und den Blauwert einzeln zugreifen zu können.

Abbildung 4: Block Farbton bei (oben) mit passender Einstellung zur Abfrage des aktuellen RGB-Wertes (unten).

Abbildung 5: Block für den Zugriff auf ein Listenelement

Aufgabe 4:

a) Überlegt euch Kriterien, nach denen ihr abhängig vom Rot-, Grün- und Blauwert das passende Kostüm für das Baustein-Objekt auswählen könnt.

b) Erweitere dein Skript so, dass beim Durchlaufen der Bühne an jeder Position das passende Kostüm ausgewählt und ein Abdruck hinterlassen wird.

Aufgabe 5: Um das Mosaik aus Bausteinen zusammenzusetzen, benötigt man entsprechend viele Bausteine der jeweiligen Farbe. Erweitere dein Programm so, dass es beim Erstellen der Vorlage für jede Farbe zählt, wie viele Bausteine benötigt werden.

Optimierung des Ergebnisses

Das Ergebnis ist nicht nur abhängig von eurem Algorithmus, sondern auch von eurem Ausgangsbild. Möglicherweise ist es daher sinnvoll, das Ausgangsbild zuvor mit einem Bildbearbeitungsprogramm vorzubereiten. Neben dem Verändern von Helligkeit und Kontrast kann es auch hilfreich sein, den Hintergrund einheitlich zu färben. Wenn du beispielsweise einen blauen Hintergrund gewählt hast und die Person blaue Augen hat, so ist es schwer anhand der RGB-Werte zwischen dem Hintergrund und der Augenfarbe zu unterscheiden, um das passende Kostüm auszuwählen. Ein fotografierter Hintergrund weist außerdem immer Schattierungen auf, so dass die RGB-Werte von Position zu Position leicht variieren. Wenn wir den Hintergrund in einer Farbe färben, die ansonsten nicht im Bild vorkommt, können wir in unserem Algorithmus anhand dieses einen RGB-Wertes einen Bildpunkt eindeutig dem Hintergrund zuordnen.

Des Weiteren kannst du überprüfen, ob du eine passende Zuordnung der Helligkeitswerte zu den Bausteinkostümen gewählt hast und ob die Bausteinkostüme eine gute Größe haben.

Um die einzelnen Bausteine in der Vorlage besser erkennen und tatsächlich danach bauen zu können, sind Bausteinkostüme mit einer Umrandung hilfreich. In Abbildung 6 wurden beispielsweise Bausteinkostüme mit grüner Umrandung verwendet.

Wenn du schon einmal einen Eindruck von dem Mosaik aus Bausteinen erhalten möchtest, kannst du statt der einfarbigen Kostüme auch Fotos von Bausteinen in den verschiedenen Farben verwenden. Ein Beispiel zeigt Abbildung 7.

Abbildung 6: Umrandung der Bausteine zur besseren Unterscheidung der Bausteine.

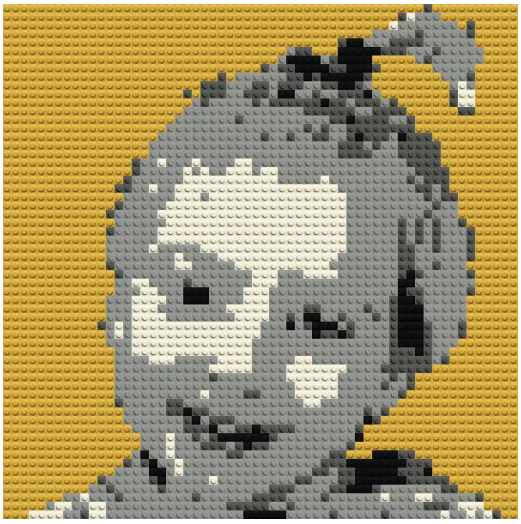

Abbildung 7: Mosaik aus Fotos von Bausteinen als Vorschau

Hast du noch weitere Ideen, wie du das Ergebnis deines Programms verbessern kannst?

Lizenz

Bildnachweis: Die Abbildungen 1, 2, 3 und 5 wurden von der Autorin selbst erstellt. Die abgebildeten Blöcke wurden der Entwicklungsumgebung Snap*!* in der Version 5.1.0 entnommen. Snap*!* wird von der University of California, Berkeley zur Verfügung gestellt: https://snap.berkeley.edu