

Algorithmische Rekonstruktion von Funktionen eines Bildbearbeitungsprogramms Didaktische Hinweise

Didaktische Einordnung

Im Informatikunterricht oder im Kunstunterricht der Sekundarstufe I haben die Schüler*innen Fotos mithilfe einer Bildbearbeitungssoftware bearbeitet und verfremdet (vgl. [4] bzw. [5]). Da wir beispielsweise in Werbung und Nachrichten täglich der Wirkung von Bildern ausgesetzt sind und diese gezielt eingesetzt werden, um uns zu beeinflussen, ist es wichtig, um die Möglichkeiten der Manipulation von Bildern zu wissen. Die Verfremdung von Bildern durch Falschfarbendarstellungen wird in verschiedenen Wissenschaften zur Veranschaulichung von Sachverhalten verwendet und begegnet den Schüler*innen daher in unterschiedlichen Schulfächern.

Diese Einheit knüpft an die Bildbearbeitung aus Anwendersicht, welche die Bedienung eines Bildbearbeitungsprogramms erfordert, an. Durch die (vereinfachte) Rekonstruktion ausgewählter Algorithmen, die den Funktionen eines Bildbearbeitungsprogramms zugrunde liegen, soll das Verständnis für die digitale Bearbeitung von Bildern vertieft werden. Damit kann dann auch ein Übergang zur digitalen Verarbeitung von Bildern und einer Diskussion der technischen Möglichkeiten geschaffen werden.

Zielgruppe

Die vorliegenden Materialien richten sich an Schüler*innen in der Qualifikationsphase, die Erfahrungen mit dem grafischen Programmieren in Snap!¹ haben. Ein Materialpaket, welches den Einsatz der textbasierten Programmiersprache Processing² vorsieht, steht separat zur Verfügung.

Voraussetzungen

Idealerweise sollten die Schüler*innen mit den Funktionen eines Bildbearbeitungsprogramms, mit der Farbdarstellung im RGB-Modell und dem Aufbau von Rastergrafiken bzw. digitalen Fotos vertraut sein. Letzteres kann auch im Kontext dieser Einheit thematisiert werden.

Weiterhin sind Vorerfahrungen im algorithmischen Problemlösen mit der grafischen Programmiersprache Snap! oder einer ähnlichen Entwicklungsumgebung notwendig sowie im Umgang mit Listen.

Lernziele

Die vorliegenden Materialien verknüpfen Erfahrungen aus der Sekundarstufe I zur Bildbearbeitung und das RGB-Modell, das spätestens in der Einführungsphase im Rahmen des Moduls *Codierung und Übertragung von Daten* im Lernfeld „Informationen und Daten“ thematisiert wurde, mit dem Lernfeld „Algorithmen und Datenstrukturen“. Dabei werden unterschiedliche Kompetenzen gefestigt und vertieft. Der Schwerpunkt liegt dabei auf der Anwendung des RGB-Modells und der

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

² Die Programmierumgebung Processing wurde 2001 von Ben Fry und Casey Reas initiiert. Nähere Informationen finden Sie unter <https://processing.org/>

algorithmischen Auswertung und Veränderung einzelner RGB-Werte zur Manipulation von Bildern. Dadurch wird erfahrbar, dass ein Rechner ein Bild nicht als Ganzes „sieht“, sondern ein Bild aus vielen einzelnen Bildpunkten besteht, wobei die Farbe eines Bildpunktes mit drei bzw. vier Zahlen codiert wird. Die vierte Zahl gibt die Transparenz oder Deckkraft an. Sie ist im Rahmen dieser Einheit jedoch nicht relevant.

Die RGBA-Werte eines einzelnen Pixels speichert Snap! in einer Liste. Die Pixel eines Bildes speichert Snap! ebenfalls in einer Liste, so dass ein Bild mit einer Liste von Listen codiert wird und der Umgang mit dieser Datenstruktur geübt wird.

Die Materialien eignen sich auch für den im Kerncurriculum explizit geforderten projektorientierten Unterricht (vgl. [6])

Werkzeuge

Möchte man mit einer grafischen Programmiersprache arbeiten, so bietet Snap! im Gegensatz zu dem in der Sekundarstufe I verbreiteten Werkzeug Scratch³ geeignete Blöcke an, um auf die Pixel eines Bildes und die RGB-Werte zuzugreifen. Diese Möglichkeit ist entscheidend, um Algorithmen zu entwerfen, die Bilder durch Operationen auf den RGB-Werten manipulieren.

Wenn man mit einer textbasierten Programmiersprache arbeiten möchte, ist Processing ein geeignetes Werkzeug, da das Erstellen von Programmen zum Erzeugen und Manipulieren von Bildern zu den Hauptanwendungsgebieten von Processing gehört und einfache Operationen zum Zugriff auf Bilder und einzelne Pixel zur Verfügung stehen. Processing basiert auf Java.

Die Abbildung eines Bildes auf eine lineare Datenstruktur ist in Snap! und Processing sehr ähnlich, so dass die Werkzeuge auch parallel eingesetzt werden können. Da sich die Umsetzung jedoch in einigen Details unterscheidet, gibt es ein gesondertes Materialpaket, das an Processing angepasst ist.

Didaktische Hinweise

Das Materialpaket setzt sich aus vier Einheiten zusammen, wobei die Punkte 1 bis 3 aufeinander aufbauen:

0. (Einstiegs-)Projekt: Portrait als Baustein-Mosaik
1. Algorithmen zur Manipulation von Bildern
2. Falschfarbendarstellungen
3. Das Zauberstab-Werkzeug

Es ist jedoch möglich, einzelne Aufgaben zu überspringen oder die Sequenz vorzeitig abzuschließen. Der Einstieg kann direkt mit dem Arbeitsblatt 1 „Algorithmen zur Manipulation von Bildern“ oder über das Projekt *Baustein-Mosaik* erfolgen. Letzteres kann auch isoliert durchgeführt werden.

Die Schüler*innen können bei der Bearbeitung der Arbeitsblätter 1 bis 3 auch individuelle Schwerpunkte setzen. Das arbeitsteilige Erarbeiten von Algorithmen für verschiedene Funktionen und das anschließende Zusammensetzen zu einem gemeinsamen Programm bieten sich ebenfalls an. Es folgen Hinweise zu den einzelnen Einheiten.

³ Scratch ist ein Projekt der Scratch Foundation in Zusammenarbeit mit der Lifelong Kindergarten Group des MIT Media Lab. Es ist kostenlos unter <https://scratch.mit.edu> erhältlich.

0. Projekt: Portrait als Baustein-Mosaik

Als (Einstiegs-)projekt wird das Zerlegen eines Portraitfotos in größere Bildpunkte in fünf verschiedenen Farben angeboten. Dabei handelt es sich um eine Farbe für den Hintergrund und vier Graustufen für das Gesicht der Person. Inspiriert ist dieses Projekt durch das Angebot eines Spielzeugherstellers ein Foto in ein Mosaik aus Bausteinen zu zerlegen und nachzubauen.

Dieses Projekt lässt sich in Snap! sehr komfortabel umsetzen, da ein Block zur Verfügung steht, mit dem ein Objekt den RGB-Wert der Bühne an seiner aktuellen Position abfragen kann. Außerdem kann ein Baustein-Objekt mit dem Block *stemple* Abdrücke auf der Bühne hinterlassen, so dass das Baustein-Mosaik direkt auf die Bühne gezeichnet werden kann, ohne das Bild selbst auf Pixelebene verändern zu müssen.

Im Materialpaket für die Sekundarstufe I wird dieses Vorgehen auch auf die Algorithmen zur Manipulation von Bildern im Allgemeinen übertragen. Für ein 1x1 großes Objekt dauert es jedoch auch unter Verwendung des Turbo-Modus relativ lange, die gesamte Bühne abzulaufen und dabei jedes Mal einen Abdruck in der passenden Farbe zu hinterlassen. Da für die Sekundarstufe II der Umgang mit zweidimensionalen Reihungen vorgesehen ist, was in Snap! Listen von Listen entspricht, wird für die Algorithmen in den Arbeitsblättern 1 bis 3 die interne Repräsentation eines Bildes als Liste von Listen genutzt, um die RGB-Werte dort direkt zu verändern und damit einen deutlichen Geschwindigkeitsvorteil zu erzielen.

Trotzdem ergeben sich aus diesem Projekt Vorkenntnisse, die für die weiteren Aufgaben hilfreich sind. So wird hier bereits das Verständnis für die Codierung von Farben im RGB-Modell vertieft und es kann schon einmal der Zugriff auf den Rot-, Grün- und Blauanteil für den RGB-Wert eines Pixels, der als Liste gespeichert ist, geübt werden.

Abhängig von den Vorkenntnissen der Schüler*innen und den Ideen, die Sie in Aufgabe 1 entwickeln, muss entschieden werden, ob die erste Seite des Aufgabenblattes ausreicht und die Schüler*innen das Projekt weitgehend selbständig umsetzen können oder ob sie die weiteren Aufgaben und Hinweise benötigen, um schrittweise an die Umsetzung herangeführt zu werden.

Da es insbesondere für das systematische Ablaufen der Bühne unterschiedliche algorithmische Ansätze gibt, wurde versucht, die Hilfestellung möglichst allgemein zu halten, so dass die Schüler*innen hier individuelle Lösungen finden können. Der Lösungsvorschlag⁴ enthält ein Beispiel mit variablen Werten. Es gibt viele weitere denkbare Algorithmen, die hier nicht alle vorgestellt werden können. Im Unterricht kann an diesem Beispiel besonders gut die Gleichwertigkeit unterschiedlicher (Schüler*innen-)Lösungen hervorgehoben werden.

Zum Abfragen des aktuellen RGB-Wertes der Bühne an der Position eines Objekts stellt Snap! im Bereich *Fühlen* den Baustein *RGBA bei selbst* zur Verfügung. Der Block bietet noch weitere Optionen u. a. für die Arbeit mit dem HSV-Modell an und dementsprechend auch die Option Helligkeit. Die Idee für das Erstellen des Baustein-Mosaiks auf der Basis der RGB-Werte besteht darin, die Helligkeit aus dem Durchschnitt des Rot-, Grün- und Blauwertes zu berechnen und den vier Grautönen der Bausteine geeignete Helligkeitsintervalle zuzuordnen. Aufgrund der Helligkeit kann dann das passende Baustein-Kostüm ausgewählt werden. Einige Schüler*innen könnten zu Recht auf die Idee kommen, die Helligkeit mithilfe des Blocks *Helligkeit bei selbst* direkt auszugeben zu lassen, anstatt

⁴ Die Lösungsvorschläge sind nur in dem Materialpaket inklusive Musterlösungen enthalten.

sie aus den RGB-Werten zu berechnen. Dabei ist jedoch Folgendes zu beachten: Während sich als Durchschnitt aus dem Rot-, dem Grün- und dem Blauanteil ein Wert zwischen 0 und 255 ergibt, bewegen sich die Helligkeitswerte des HSV-Modells zwischen 0 und 100, so dass die Skalen nicht identisch sind. Eine eigene Berechnung hat zudem den Vorteil, dass das Vorgehen exemplarisch für die Auswertung des Rot-, Grün- und Blauwertes ist und damit die Rekonstruktion weiterer Funktionen eines Bildbearbeitungsprogramms vorbereitet. Es sollte daher im Vorfeld überlegt werden, wie mit dieser Möglichkeit umgegangen wird, so dass die Idee der Schüler*innen einerseits nicht abgewertet wird, andererseits aber der Zusammenhang mit den RGB-Werten bzw. die Unterschiede der Farbmodelle deutlich werden.

Der beiliegende Lösungsvorschlag enthält drei algorithmische Varianten zur Auswahl der Baustein-Kostüme. Tabelle 1 gibt einen Überblick. Weitere Varianten sind denkbar.

Eigener Block	Besonderheiten
waehle Kostuem 1	Verwendung des Blocks <i>RGBA bei selbst</i> . Die Intervalle für die Graustufen wurden experimentell ermittelt.
waehle Kostuem 2	Verwendung des Blocks <i>Helligkeit bei selbst</i> . Dabei wird der Bereich von 0 bis 100 rechnerisch in vier Intervalle eingeteilt und entsprechend der Intervallnummer das Kostüm ausgewählt. Dieser Ansatz basiert auf einer Idee von Jens Möning (vgl. [3])
waehle Kostuem 3	arbeitet wie <i>waehle Kostuem 1</i> , allerdings wird hier von einem Bühnenbild ausgegangen, bei dem der Hintergrund vorher bearbeitet wurde, so dass alle Pixel des Hintergrunds den gleichen RGB-Wert haben (hier 0, 0, 255). Als Bühnenbild sollte daher das dritte Hintergrundbild der Bühne ausgewählt werden.

Tabelle 1: Erläuterung zu den Unterschieden der drei Lösungsvarianten

Beim Erstellen der Portraitfotos, sollte die Hintergrundfarbe so gewählt werden, dass es möglichst keine Überschneidung mit den Farben des eigentlichen Portraits gibt. Im Beispiel ist dies nicht vollständig gelungen, da der blaue Hintergrund schwer vom Blau der Augen zu unterscheiden. Steht keine passende Farbe als Hintergrund zur Verfügung, kann beispielsweise auch ein weißer Hintergrund zuvor mit einem Bildbearbeitungsprogramm und dem Zauberstab-Werkzeug einheitlich gefärbt werden.

Das Ergebnis wird immer abhängig von der Helligkeit und dem Kontrast des verwendeten Fotos sein, so dass sich ein Algorithmus ggf. nicht auf beliebige Fotos anwenden lässt. Ziel sollte daher nur sein, dass die Schüler*innen einen Algorithmus entwerfen, der ihr eigenes Foto in ein passendes Baustein-Mosaik transformiert.

1. Algorithmen zur Manipulation von Bildern

Die Grundidee aller weiteren im Rahmen dieser Einheit entwickelten Algorithmen besteht darin, die Liste, welche die Farbwerte zu jedem Bildpunkt eines Kostüms als weitere Liste enthält, zu durchlaufen und je nach gewünschter Manipulation des Bildes zu verändern. Dazu muss zunächst eine Vorstellung entwickelt werden, wie ein Bild, das aus Zeilen und Spalten von Farbpunkten besteht und damit erst einmal zweidimensional ist, auf eine eindimensionale Datenstruktur

abgebildet werden kann. Nimmt man die Liste für die Farbanteile als weitere Dimension hinzu, wird eine dreidimensionale Struktur auf eine zweidimensionale abgebildet.

Der Block, der zum Auslesen der RGB-Werte eines Bildes bzw. Kostüms notwendig ist, wird vorgestellt. Ebenso der Block, der die veränderten RGB-Werte als Kostüm auf der Bühne darstellt. Da davon ausgegangen wird, dass es sich bei der programminternen Darstellung des Bildes als Liste von Listen um eine bekannte Datenstruktur handelt, können die Schüler*innen die Skripte zur Manipulation der RGB-Werte von Anfang an selbst entwickeln.

Als erste Funktion zur Manipulation eines Bildes wird das Invertieren der Farbwerte gewählt, weil die mathematische Berechnung hier relativ einfach und eindeutig ist. Da es sich um das erste Beispiel handelt, ist der Hinweis zur Berechnung direkt im Arbeitsblatt enthalten.

Die Lösungen der Schüler*innen sollten gründlich besprochen werden, da die Implementierung des Algorithmus zum Invertieren der Farbwerte die Grundlage für alle weiteren Algorithmen zur Bearbeitung von Bildern ist. Dies kann von den Schüler*innen in einer abschließenden Aufgabe auch selbst reflektiert werden.

Solange jeder Pixel in der Liste bearbeitet werden soll, kann dies am einfachsten mit der Schleife *für jedes Element von*, die speziell für Listen zur Verfügung steht, umgesetzt werden. Um die Anwendung von Funktionen auf Teilbereiche vorzubereiten und das Verständnis für die zweidimensionale Struktur der Liste von Listen zu vertiefen, wird in Aufgabenteil b) von Aufgabe 3 die Funktion nur auf die obere Hälfte des Bildes angewendet. Dazu muss in einer Zählschleife auf die passenden Elemente der äußeren Liste zugegriffen werden.

Das Erstellen eines Graustufenbildes unterscheidet sich algorithmisch nur in der Berechnung der neuen Farbwerte. Die Berechnung des passenden Grautons zu einem Farbton kann jedoch unterschiedlich definiert werden. Da der Grauton die Helligkeit des Bildpunktes wiedergibt, kann er aus dem Durchschnitt des Rot-, Grün- und Blauwertes berechnet werden. Beim Berechnen der Helligkeit eines Pixels können die Farbanteile jedoch auch unterschiedlich gewichtet oder der Median der drei Farbanteile bestimmt werden (vgl. z. B. [1]). Basierend auf dem RGB-Modell können die Schüler*innen hier eigene Vorschläge machen. Um diese zu vergleichen, bietet sich eine arbeitsteilige Implementierung verschiedener Vorschläge an.

Anschließend sollten die Schüler*innen selbständig Algorithmen für verschiedene weitere Funktionen erstellen können. Dieser Teil kann auch als arbeitsteiliges Projekt angelegt werden.

Je nach Komplexität der Funktion kann es dabei nicht darum gehen, die exakten Berechnungen eines Bildbearbeitungsprogramms zu rekonstruieren, sondern vereinfachte rechnerische Ansätze zu finden, die das Bild in die gewünschte Richtung verändern. Auch die Lösungsvorschläge sind als Vereinfachungen in diesem Sinne zu verstehen.

Bevor die Hinweise zum Einsatz kommen, sollten die Schüler*innen daher zu einem gezielten Experimentieren mit den RGB-Werten auf der Basis ihres Wissens über die Bedeutung der Farbwerte angeregt werden.

Bei diesem Arbeitsblatt ergeben sich vielfältige Möglichkeiten zur Differenzierung. Sowohl hinsichtlich der Quantität der Funktionen, die umgesetzt werden, als auch der Komplexität. Auch einzelne Funktionen lassen sich unterschiedlich komplex umsetzen. So kann z. B. beim Verändern der Helligkeit mit einem festen Wert gearbeitet werden oder dem Anwender die Eingabe über einen

Schieberegler oder ein Eingabefeld ermöglicht werden. Weiterhin kann der Schwerpunkt auf das kreative Experimentieren mit den RGB-Werten oder auf das gezielte Rekonstruieren einzelner Effekte gelegt werden.

Ab Version 6 von Snap! wird das Anwenden einer Funktion auf alle Elemente einer Liste weiter vereinfacht. Das Skript in Abbildung 1 zum Invertieren der RGB-Werte kann damit auf das Skript in Abbildung 2 reduziert werden.

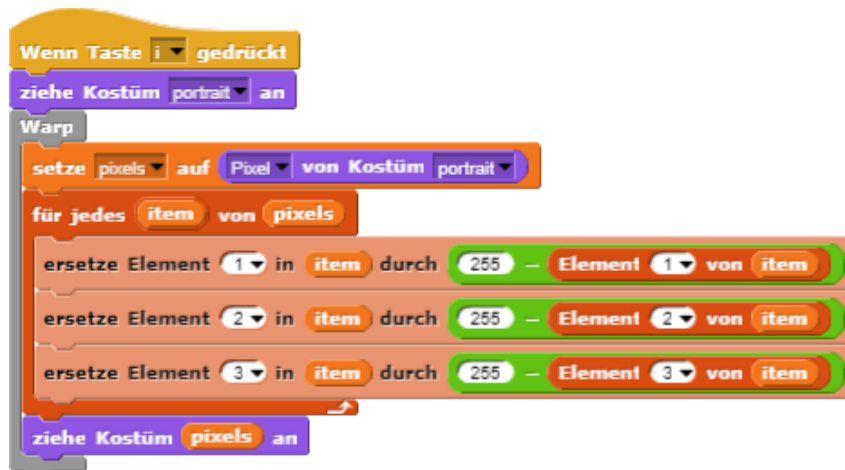


Abbildung 1: Skript zum Invertieren der RGB-Werte eines Bildes

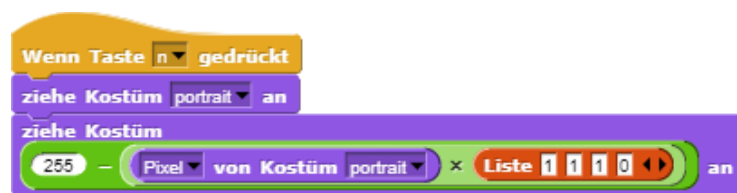


Abbildung 1: Skript zum Invertieren der RGB-Werte eines Bildes ab Version 6 von Snap!

Die Multiplikation in Abbildung 2 wird auf die RGBA-Liste jedes Pixels angewendet, indem der erste, zweite und dritte Wert in jeder RGBA-Liste mit 1 und der letzte Wert mit 0 multipliziert werden. Anschließend wird jeder einzelne Werte in der Liste durch die Differenz von 255 und dem aktuellen Wert ersetzt. Dadurch werden der Rot-, Grün- und Blauwert für jeden Bildpunkt invertiert. Um den Alphawert bei 255 zu belassen, wird er durch die Multiplikation zuerst auf 0 gesetzt, damit sich bei der Subtraktion wieder 255 ergibt. Dieses Beispiel und weitere Details zu den Neuerungen in Snap! 6.0 finden Sie in [2].

2. Falschfarbendarstellungen

Das Erzeugen von Falschfarbenbildern stellt die Schüler*innen algorithmisch vor keine neuen Herausforderungen. Das Arbeitsblatt eignet sich daher gut zum selbständigen Weiterarbeiten der Schüler*innen nach Arbeitsblatt 1. Die Falschfarbendarstellung kann aber auch alternativ zu den verschiedenen Manipulationen eines Bildes in Aufgabe 5 von Arbeitsblatt 1 eingesetzt werden.

Da die Schüler*innen aus dem vorangegangenen Arbeitsblatt auf Erfahrungen im Umgang mit den RGB-Werten zurückgreifen können, sieht das Arbeitsblatt vor, dass sie die Grundlagen für das Erstellen von Falschfarbenbildern, Rot-, Grün- und Blauauszug und Farbcodierungen anhand von Beispielen selbst erarbeiten.

Da das Arbeitsblatt entsprechend wenig Erklärungen enthält, ist für Möglichkeiten der Zwischen-
sicherung zu sorgen, insbesondere wenn Schüler*innen das Arbeitsblatt weitgehend selbständig
bearbeiten.

Im Zusammenhang mit der Falschfarbendarstellung gibt es viele Anwendungsgebiete in
verschiedenen Wissenschaften, die sich auch in den entsprechenden Schulfächern wiederfinden
lassen. Hier bestehen daher verschiedene Möglichkeiten des Fächerübergriﬀs beispielsweise zu
Biologie oder Erdkunde, indem entsprechende Kontexte gewählt werden, in denen diese
Darstellungen zum Einsatz kommen.

3. Das Zauberstab-Werkzeug

Bei der Rekonstruktion des Zauberstab-Werkzeuges kommen das Auswählen eines Referenzwertes
und der Vergleich von RGB-Werten hinzu.

Da vom Anwender nicht erwartet werden kann, dass er den Referenzwert als RGB-Wert eingibt,
bietet sich die Auswahl eines Bildpunktes mit der Maus an, dessen Farbwert als Referenz dienen soll.
Das lässt sich in Snap! sehr einfach umsetzen, da ein passender Block zur Verfügung steht, der den
RGB-Wert an der Position der Maus als Liste zurückgibt.

Da ein Farbton aus drei Werten besteht, ist weiterhin zu erarbeiten, dass zwei RGB-Werte verglichen
werden können, indem der Rot-, der Grün- und der Blauanteil verglichen werden. Hinzu kommt, dass
es bei Fotos viele Nuancen eines Farbtons gibt, so dass RGB-Werte auch als gleich eingestuft werden
müssen, wenn der Rot-, Grün- und Blauanteil nur im gleichen Bereich liegen. Programme zur
Bildbearbeitung ermöglichen dem Anwender hier in der Regel die Toleranz festzulegen. Aufgabe 3
soll entsprechende Überlegungen bei den Schüler*innen motivieren.

Das Umfärben aller Pixel, die zu dem Referenzpunkt passen, lässt sich mit diesen Vorüberlegungen
relativ einfach und ganz ähnlich zu den zuvor betrachteten Funktionen implementieren.

Beim Zauberstab-Werkzeug kommt es allerdings besonders häufig vor, dass man die Anwendung auf
einen Bereich des Bildes oder auf zusammenhängende Pixel beschränken möchte. Wenn ein Objekt
freigestellt werden soll, möchte man beispielsweise nur die Pixel des Hintergrunds weiß färben, nicht
aber gleichfarbige Pixel, die zu dem Objekt gehören. Im Kontext des Zauberstab-Werkzeuges wird
daher sowohl die Beschränkung auf einen rechteckigen Auswahlbereich als auch auf
zusammenhängende Pixel betrachtet. In beiden Fällen muss ein Zusammenhang zwischen den
Koordinaten eines Bildpunktes auf der Bühne und dem Index in der Liste aller Pixel hergestellt
werden. Dieser Zusammenhang ist in Snap! komplexer als in Processing, da das Koordinatensystem
der Bühne seinen Ursprung im Zentrum der Bühne hat und die Koordinaten daher nicht wie in
Processing mit den Spalten und Zeilennummern der Bildpunkte gleichgesetzt werden können. Da der
Zusammenhang zwischen den Koordinaten eines Bildpunktes und dem Index in der Liste der Pixel
ansonsten nicht benötigt wird, sollte daher gut überlegt werden, ob dieser Teil des Arbeitsblattes im
Unterricht thematisiert wird oder ggf. nur den leistungsstarken Schüler*innen angeboten wird.

Die Transformation wird hier in zwei Schritten motiviert und durchgeführt. Zunächst werden die
Koordinaten der Bühne in die Spalten- und Zeilennummer der Bildpunkte transformiert. Dabei wird
davon ausgegangen, dass der Bildpunkt in der linken, oberen Ecke die Spalten und Zeilennummer 0
hat. Das entspricht dem Koordinatensystem des Programmfensters in Processing. Es wäre auch
möglich die Spalten und Zeilen mit 1 beginnend zu nummerieren. Dann müssten die folgenden
Rechnungen entsprechend angepasst werden. Um das Zentrum in die linke, obere Ecke zu

verschieben und die Richtung der y-Achse umzukehren, muss zu der x-Koordinate die halbe Bühnenbreite addiert werden. Von der y-Koordinate wird die halbe Bühnenbreite subtrahiert und vom Ergebnis der Betrag genommen.

Das Bild ist zeilenweise in der Reihung `pixels` gespeichert. Da die transformierten x- und y-Koordinaten nun die Spalte bzw. Zeile angeben und die Zählung bei 0 beginnt, erhält man den Index des ersten Pixels einer Zeile, indem man die y-Koordinate mit der Zeilenlänge also der Breite des Bildes multipliziert und 1 addiert. Da die x-Koordinate die Spalte in dieser Zeile angibt muss schließlich noch die x-Koordinate addiert werden.

Transformiert man die Koordinaten so, dass die Nummerierung der Spalten und Zeilen bei 1 beginnt, kann der Index mit der Formel $(y - 1) \cdot \text{Zeilenlänge} + x$ berechnet werden.

Ist der Zusammenhang zwischen den Koordinaten eines Bildpunktes und dem Index in der Liste aller Pixel hergestellt, kann die Beschränkung auf den Auswahlbereich relativ einfach implementiert werden, indem zwei geschachtelte Zählschleifen zwischen den Grenzen des Auswahlbereichs für die x- und die y-Koordinate iterieren.

Bei der Auswahl zusammenhängender Pixel müssen rekursiv oder iterativ solange die Nachbapixel betrachtet werden, bis ein Pixel nicht mehr die Auswahlkriterien erfüllt. Im Falle des Zauberstab-Werkzeugs werden also solange die Nachbapixel hinzugenommen, bis ein Pixel keinen ähnlichen Farbwert mehr aufweist. Auch beim Bestimmen der Nachbapixel ist es einfacher zunächst von der x- und y-Koordinate auszugehen. Da hier die Koordinaten nur um jeweils ± 1 variiert werden müssen. Es lässt sich auch leicht ermitteln, ob die Koordinaten noch in den Grenzen des Bildes liegen, da die Koordinaten dazu Werte zwischen Null und der Breite bzw. der Höhe des Bildes haben müssen. Aus den Koordinaten lässt sich dann auf bekannte Weise der Index in der Reihung `pixels` bestimmen. Je nach Kenntnisstand der Schüler*innen können sie entsprechende Ansätze selbst entwickeln und vergleichen.

Da der rekursive Ansatz ggf. die maximale Rekursionstiefe bzw. den für die Rekursion zur Verfügung stehenden Speicher übertrifft, wird in der Musterlösung ein iterativer Ansatz verfolgt. Dazu werden alle Nachbapixel, die farblich ähnlich sind, weiß gefärbt und die Koordinaten in einer Snap/-Liste abgelegt. Für alle Pixel, deren Koordinaten der Liste entnommen werden, werden ebenfalls die farblich passenden Nachbarn ermittelt, weiß gefärbt und in die Liste aufgenommen. Die Bearbeitung endet, wenn die Liste leer ist. Alternativ kann hierfür auch die im KC vorgesehene Datenstruktur Schlange verwendet werden.

Ausblick

Im Internet sind prominente Beispiele zu finden, bei denen Bilder in der Presse gezielt manipuliert wurden (s. [7] und [8]). Im Kontext dieser Einheit bietet es sich an, über die Motive für solche Manipulationen, aber auch die technischen Grundlagen zu sprechen, die solche Manipulationen ermöglichen. Daraus ergibt sich dann u. a. die Frage, wie Manipulationen eines Bildes erkannt werden können. Neben Unstimmigkeiten im Bild, die auf Manipulationen hinweisen, wäre hier auch eine Art Prüfcode denkbar, der nicht mehr zu dem Bild passt, wenn es manipuliert wird. Hier können die Schüler*innen eigene Ideen entwickeln bzw. Verbindungen zum Modul *Kryptologie* herstellen.

Denkbar ist auch eine Vertiefung mit Aspekten aus den Bereichen *Computer sehen* und *automatisierte Bildverarbeitung*.

Literaturverzeichnis

- [1] Janecke, M. (2013). Farbfotos in Graustufen umwandeln. <https://prlbr.de/2013/farben-in-graustufen-umwandeln/> [Datum des Zugriffs: 29.05.2020]
- [2] Mönig, J. (2020). Snap! 6.0 <https://www.youtube.com/watch?v=GZkQC3nWGnk> [Datum des Zugriffs: 25.06.2020]
- [3] Mönig, J. (2020). *Dithering - Dice – Hasso* <https://snap.berkeley.edu/project?user=jens&project=Dithering%20-%20Dice%20-%20Hasso> [Datum des Zugriffs: 13.05.2020]
- [4] Niedersächsisches Kultusministerium (2014). *Kerncurriculum für die Schulformen des Sekundarbereichs I Schuljahrgänge 5 – 10. Informatik*. Hannover: Unidruck
- [5] Niedersächsisches Kultusministerium (2016). *Kerncurriculum für das Gymnasium Schuljahrgänge 5 – 10. Kunst*. Hannover: Unidruck
- [6] Niedersächsisches Kultusministerium (Hrsg.) (2017) *Kerncurriculum für das Gymnasium - gymnasiale Oberstufe, die Gesamtschule – gymnasiale Oberstufe, das Kolleg. Informatik*. Hannover: unidruck
- [7] Museum für Kommunikation. (2007). *X für U Bilder, die lügen. Didaktische Materialien*. <https://www.mfk.ch/lehrmittel/lehrmittel-bilder-die-luegen> [Datum des Zugriffs: 26.02.2022]
- [8] Spiegel Geschichte (Hrsg.) (2008). *Bildmanipulationen Mehr Blut, mehr Rauch, weniger Speck*. <https://www.spiegel.de/fotostrecke/manipulierte-bilder-fotostrecke-107186.html> [Datum des Zugriffs: 29.05.2020]

Lizenz

Dieses Werk ist lizenziert unter einer [Creative Commons Namensnennung - Nicht kommerziell - Keine Bearbeitungen 4.0 International Lizenz](#).

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

Für die korrekte Ausführbarkeit der beiliegenden Quelltexte wird keine Garantie übernommen. Auch für Folgeschäden, die sich aus der Anwendung der Quelltexte oder durch eventuelle fehlerhafte Angaben ergeben, wird keine Haftung oder juristische Verantwortung übernommen.