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. [2] bzw. [3]). 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 Bildbearbeitungsprogramm zugrunde liegen, soll das Verständnis für die digitale Bearbeitung von Bildern vertieft werden. Damit kann dann auch ein Übergang zur digitalen *Ver*arbeitung 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 textbasierten Programmieren in Processing[1] haben. Ein Materialpaket, welches den Einsatz der graphischen Programmiersprache Snap*!*[2] vorsieht, steht ebenfalls 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 ggf. auch im Kontext dieser Einheit thematisiert werden.

Weiterhin sind Vorerfahrungen im algorithmischen Problemlösen mit Processing notwendig. Zur Erarbeitung der Grundlagen können beispielsweise die Materialien aus dem Paket *Einstieg in die textbasierte Programmierung mit Processing* verwendet werden. Hier sind insbesondere die Einheiten *1_Einstieg* und *4_Eigene Methoden* relevant. Ergänzend dazu müssen die Schüler*innen mit Reihungen umgehen können.

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*

[1] Die Programmierumgebung Processing wurde 2001 von Ben Fry und Casey Reas initiiert. Nähere Informationen finden Sie unter https://processing.org/

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

Übertragung von Daten im Lernfeld „Informationen und Daten“ thematisiert wurde, mit dem Lernfeld „Algorithmen und Datenstrukturen“. Dabei werden unterschiedliche Kompetenzen aus diesem Bereich gefestigt und vertieft. Der Schwerpunkt liegt dabei auf der Anwendung des RGB-Modells und der 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 Zahlen codiert wird. Die Farbwerte der einzelnen Pixel eines Bildes speichert Processing in einer eindimensionalen Reihung. Im Zusammenhang mit der Implementierung des Zauberstab-Werkzeugs können darüber hinaus optional die Datenstrukturen *Dynamische Reihung* oder *Schlange* eingesetzt werden.

Die Materialien eigenen sich auch für den im Kerncurriculum explizit geforderten projektorientierten Unterricht (vgl. [4])

Werkzeuge

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. Wurde bisher mit Java gearbeitet, ist ein Umstieg auf Processing vermutlich einfacher als vergleichbare Algorithmen mit den reinen Java-Bibliotheken umzusetzen. Alternativ kann die grafische Programmiersprache Snap*!* verwendet werden. 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 Snap*!* angepasst ist.

Didaktische Hinweise

Das Materialpaket setzt sich aus vier Einheiten zusammen, die aufeinander aufbauen:

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

Es ist jedoch möglich, einzelne Aufgaben zu überspringen oder die Sequenz vorzeitig abzuschließen. Die Schüler*innen können bei der Bearbeitung 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.

1. Algorithmen zur Manipulation von Bildern

Die Grundidee aller im Rahmen dieser Einheit entwickelten Algorithmen besteht darin, die eindimensionale Reihung, welche die Farbwerte zu jedem Bildpunkt des Bildes im Programmfenster 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.

Die Operationen, die zum Laden eines Bildes in das Programmfenster und Zugriff auf die Farbwerte notwendig sind, werden an einem Beispiel eingeführt. Hier sind die Operationen *loadImage()* und *loadPixels()* zu unterscheiden. *loadImage()* lädt ein Bild aus einer Datei, so dass es in einer Variablen vom Typ *PImage* gespeichert werden kann und im Programm zur Verfügung steht. Mit *loadPixels()* werden hingegen die Farbwerte des Bildes, das aktuell im Programmfenster zu sehen ist, in die Reihung `pixels` übernommen. Die Werte in der Reihung `pixels` können beliebig verändert und das Ergebnis mit der Operation *updatePixels()* im Programmfenster sichtbar gemacht werden. Das in einer Variablen vom Typ *PImage* gespeicherte Bild bleibt von den Veränderungen unberührt.

Der Umgang mit diesen Operationen wird in Aufgaben gefestigt, bevor Algorithmen zur gezielten Manipulation des Bildes im Programmfenster entworfen werden.

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.

Da davon ausgegangen wird, dass die Schüler*innen Erfahrungen im Umgang mit eindimensionalen Reihungen haben und die algorithmischen Konzepte beherrschen, die notwendig sind, um einen Algorithmus zum Invertieren der Farbwerte zu implementieren, wird die Umsetzung hier nicht vorgegeben. Stattdessen werden lediglich die benötigten Operationen vorgestellt. Hier ist zu beachten, dass Processing für die Farbwerte einen speziellen Datentyp *color* verwendet, in dem der Rot-, der Grün- und der Blauwert zu einer Ganzzahl zusammengefasst sind. Der Zugriff auf die einzelnen Farbanteile erfolgt über entsprechende Operationen, die es erleichtern den Überblick zu behalten.

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.

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 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.

2. Falschfarbendarstellungen

Das Erzeugen von Falschfarbenbilden 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 9 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 Zwischensicherung 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. Die Koordinaten der Maus entsprechen dabei den Koordinaten des Bildpunktes im Koordinatensystem des Programmfensters. Um den Farbwert des Bildpunktes aus der Reihung `pixels` auszulesen, muss daher zunächst eine Rechenvorschrift hergeleitet werden, mit der aus den x- und y-Koordinaten eines Bildpunktes der passende Index in der Reihung `pixels` berechnet werden kann. Entsprechende Überlegungen sollen mithilfe von Abbildung 2 in Aufgabe 2 motiviert werden. Das Bild ist zeilenweise in der Reihung `pixels` gespeichert. Da die y-Koordinate die Zeile angibt und die Zählung bei 0 beginnt, erhält man den Index des ersten Pixels der Zeile, indem man die y-Koordinate mit der Zeilenlänge also der Breite des Bildes multipliziert. Da die x-Koordinate die Spalte in dieser Zeile angibt muss die x-Koordinate noch addiert werden.

Da ein Farbton aus drei Werten besteht, ist zunächst 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 4 soll entsprechende Überlegungen bei den Schüler*innen motivieren.

Beim Zauberstab-Werkzeug kommt es besonders häufig vor, dass man die Anwendung auf einen Bereich des Bildes oder auf zusammenhänge 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-Werkzeugs wird daher sowohl das Aufziehen eines rechteckigen Auswahlbereichs mit der Maus als auch das Bearbeiten zusammenhängender Pixel betrachtet.

Eine einfache Variante zum Aufziehen von Rechtecken mit der Maus ist bereits in den Einstiegsmaterialien zu Processing enthalten. Die entsprechende Aufgabe in diesem Arbeitsblatt ist so aufgebaut, dass dem Anwender mehr Flexibilität beim Aufziehen des Rechtecks gegeben werden kann, indem die Maus vom Startpunkt aus in alle Richtungen bewegt werden kann. Dazu müssen die Koordinaten der Maus am Startpunkt und am aktuellen Punkt innerhalb des Programms der linken, oberen und der rechten, unteren Ecke zugeordnet werden. Daraus können dann die Parameter zum Zeichnen des entsprechenden Rechtecks abgeleitet werden. Um dem Anwender während des Ziehens nur die aktuelle Auswahl anzuzeigen, kann die vorherige Vorschau mithilfe der Funktion *updatePixels()* auf einfache Weise übermalt werden.

Da bereits erarbeitet wurde, wie den x- und y- Koordinaten eines Bildpunktes der Index in der Reihung `pixels` zugeordnet werden kann, können die bereits implementierten Algorithmen so verändert werden, dass zwei geschachtelte Zählschleifen jeweils durch die y- bzw. die x-Koordinate iterieren und der dazu passenden Index der Reihung `pixels` betrachtet wird.

Bei der Auswahl zusammenhängender Pixel müssen rekursiv oder iterativ solange die Nachbarpixel betrachtet werden, bis ein Pixel nicht mehr die Auswahlkriterien erfüllt. Im Falle des Zauberstab-Werkzeugs werden also solange die Nachbarpixel hinzugenommen, bis ein Pixel keinen ähnlichen Farbwert mehr aufweist. Auch beim Bestimmen der Nachbarpixel 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 0 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 beim Testen die von Processing vorgesehene maximale Rekursionstiefe bzw. den für die Rekursion zur Verfügung stehenden Speicher übertroffen hat, wird in der Musterlösung ein iterativer Ansatz verfolgt. Dazu werden alle Nachbarpixel, die farblich ähnlich sind, weiß gefärbt und in einer dynamischen linearen Datenstruktur abgelegt. Für alle Pixel, die der Datenstruktur entnommen werden, werden ebenfalls die farblich passenden Nachbarn ermittelt, weiß gefärbt und in die Datenstruktur aufgenommen. Dazu können beispielsweise die im KC vorgesehenen Datenstrukturen Schlange oder dynamische Reihung verwendet werden. Solange dazu noch keine Bibliotheken zur Verfügung stehen, verwendet der Lösungsvorschlag die von Processing bereitgestellte Datenstruktur *ArrayList*. Dies ist jedoch nicht als Empfehlung zu verstehen. Die Datenstruktur kann problemlos durch die im Unterricht verwendete Implementierung der Datenstrukturen *Schlange* oder *dynamische Reihung* mit den entsprechenden Operationen ersetzt werden.

Im Lösungsvorschlag werden x- und y- Koordinate jeweils in einer Reihung der Länge zwei zusammengefasst. Alternativ können die x- und die y-Koordinate parallel in zwei eindimensionalen Schlangen bzw. dynamischen Reihungen gespeichert werden.

4. Projekt: Portrait als Baustein-Mosaik

Als Abschlussprojekt wird das Zerlegen eines Portraitfotos in gröbere 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.

Für dieses Projekt werden verschiedene algorithmische Ansätze benötigt, welche die Schüler*innen in den vorangegangenen Arbeitsblättern bereits erarbeitet haben:

- Iteration durch die Reihung `pixels` bzw. die Koordinaten der Pixel des Bildes
- Zugriff auf Nachbarpixel
- Bestimmen der Helligkeit eines Pixels
- Einteilen der Farbwerte in Intervalle
- Vergleichen von RGB-Werten mit einem Referenzwert
- Umfärben von Pixeln

Die Aufgabe wurde daher nicht in Teilaufgaben zerlegt, sondern überlässt es den Schüler*innen hier die bekannten Ansätze zu kombinieren. Neu ist lediglich das Zusammenfassen mehrerer Pixel zu einem größeren Bildpunkt. Um hierzu entsprechende Überlegungen zu motivieren, erhalten die Schüler*innen einen stark vergrößerten Ausschnitt von 25 Pixeln. Daran können die Schüler*innen sich einerseits verdeutlichen, wie die Koordinaten einer quadratischen Pixelgruppe zusammenhängen. Anderseits ist hier zu sehen, dass die RGB-Werte benachbarter Pixel nur kleine Abweichungen aufweisen. Damit lässt sich begründen, vereinfachend nur ein Pixel der Gruppe zum Bestimmen der Helligkeit und Auswahl der Bausteinfarbe zu betrachten. Ergänzend kann hier ein Foto so stark vergrößert werden, dass die einzelnen Pixel zu erkennen sind. Dabei wird man auch Beispiele finden, in denen die Farbwerte benachbarter Pixel stärker abweichen. Die Schüler*innen können abwägen, ob sie den Mehraufwand für das Berücksichtigen aller Pixel beim Bestimmen der Helligkeit einer Pixelgruppe als lohnend erachten.

Im Lösungsvorschlag wird für das Erstellen der Vorlage das Pixel in der Mitte der Pixelgruppe betrachtet. Beim Erstellen der Vorschau wird hingegen das Pixel oben links betrachtet, da diese Koordinaten als Parameter für das Zeichnen des Baustein-Bildes angegeben werden müssen. Dadurch unterscheiden sich die Ergebnisse geringfügig voneinander.

Abschließend werden einige Anregungen für Optimierungen gegeben. Zusätzlich könnte hier das Einbeziehen aller Pixel in die Auswahl der Bausteinfarbe in Betracht gezogen werden.

Ausblick

Im Internet sind prominente Beispiele zu finden, bei denen Bilder in der Presse gezielt manipuliert wurden (s. [5] und [6]). 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] Niedersächsisches Kultusministerium (2014). *Kerncurriculum für die Schulformen des Sekundarbereichs I Schuljahrgänge 5 – 10. Informatik*. Hannover: Unidruck

[3] Niedersächsisches Kultusministerium (2016). *Kerncurriculum für das Gymansium Schuljahrgänge 5 – 10. Kunst*. Hannover: Unidruck

[4] Niedersächsisches Kultusministerium (Hrsg.) (2017) *Kerncurriculum für das Gymnasium - gymnasiale Oberstufe, die Gesamtschule – gymnasiale Oberstufe, das Kolleg. Informatik.* Hannover: unidruck

[5] Museum für Kommunikation. (2007). *X für U Bilder, die lügen. Didaktische Materialien.* https://www.mfk.ch/fileadmin/user_upload/zzz_Dateiliste_alte_Seite/pdfs/Bildung_Vermittlung/Materialien/Ausstellungen/Bdl/Bdl_didakt_Materialien.pdf [Datum des Zugriffs: 29.05.2020]

[6] 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