UNTERRICHTSVERLAUF

Die Unterrichtseinheit ist in zwei Teile gegliedert, die ineinander greifen. Zunächst werden Algorithmen zur digitalen Bildbearbeitung behandelt. Deren Prinzipien werden durch Aufgaben begleitet, die die SuS schriftlich lösen sollen. Diese sind mit gekennzeichnet.

Danach werden die Algorithmen in einer BlueJ-Umgebung implementiert, die es erlaubt, die Operationen direkt auf Beispielbilder anzuwenden und diese anzuzeigen. Dazu enthält der Ordner IMP einige Klassen, die dies ermöglichen. Aufgaben, die in dieser Umgebung bearbeitet werden sollen, sind auf den Aufgabenblättern mit gekennzeichnet. Oft schließen sich Experimente oder Tests mit den neu implementierten Methoden an.

Nun schließt sich die Erstellung einer graphischen Benutzeroberfläche (GUI) an, mit der diese Bildbearbeitungsalgorithmen komfortabel ausgeführt werden können. Die Gestaltung der Oberfläche wird im Gluon Scene Builder vorgenommen. (Kennzeichnung:). Es ist sinnvoll, das Erstellen der Benutzeroberfläche parallel zum ersten Teil als Hausaufgabe bearbeiten zu lassen, um Zeit während des Unterrichts zu sparen (vgl. 4. Doppelstunde).

Die Verbindung der Algorithmen und der Oberfläche geschieht durch Implementierung eines geeigneten Controllers. Wichtig ist dabei, dass dies im BlueJ-Projekt "bildbearbeitung" erfolgt. (Kennzeichnung:). Das Unterverzeichnis IMP enthält auch hier die gleichen Klassen wie beim ersten Projekt, diese sind aber teilweise anders implementiert. Der PictureViewer ist hier ein JavaFX-Control-Element, das in die GUI eingebunden werden kann. Die Klassen aus dem alten Projekt müssen mit "Add class from file ..." in BlueJ übernommen werden.

Da es eine Vielzahl von interessanten Algorithmen im Bereich der Bildbearbeitung gibt, lässt sich diese Unterrichtseinheit bei Bedarf gut erweitern. Es könnte sich eine umfangreichere Hausaufgabe für die Schülerinnen und Schüler (im Folgenden als SuS abgekürzt) anschließen, in der ein weiterer Algorithmus implementiert werden muss. Auch als GFS Thema sind sie gut geeignet. Die Aufgabenblätter zu den Erweiterungen sind deutlich weniger detailliert als der Pflichtteil. Sie sind daher schwerer zu bearbeiten. Die Vorgehensweise ist aber analog zum Pflichtteil: In jedem Arbeitsblatt werden die vier Arbeitsschritte durchlaufen. Auch hier sind die Aufgaben entsprechend gekennzeichnet.

Hinweis zu BlueJ und JavaFX:

1. Bei älteren BlueJ Versionen kann es zu folgender Fehlermeldung kommen:

Fehlermeldung:

```
java.lang.IllegalStateException: Toolkit not initialized
        at com.sun.javafx.application.PlatformImpl.runLater(PlatformImpl.java:273)
        at com.sun.javafx.application.PlatformImpl.runLater(PlatformImpl.java:268)
        at javafx.application.Platform.runLater(Platform.java:83)
        at bluej.runtime.ExecServer.runOnTargetThread(ExecServer.java:902)
        at bluej.runtime.ExecServer.access$700(ExecServer.java:78)
        at bluej.runtime.ExecServer$3.run(ExecServer.java:787)
```

Installieren Sie dann bitte die neuste Version oder wählen unter Werkzeuge->Einstellungen->Diverses->Run user code in this project on thread die Option "Default" (BlueJ 4.1.2).

2. Wiederherstellung von Objekten in BlueJ

Zum Testen der eigenen Implementierung müssen immer wieder die gleichen Objekte erzeugt werden. Mit Rechtsklick auf eine Klasse (z.B. GeometrsicheBildoperationen) kann man eine "Testklasse erzeugen". Dann erstellt man die benötigten Objekte und ruft bei der Testklasse "Objektzustand speichern" auf. Danach können jederzeit mit "Objektzustand wiederherstellen" sehr schnell alle Objekte neu erzeugt werden.

1. Doppelstunde: Spiegeln und Drehen

Ziele

Java: Deklaration und Verwendung von zweidimensionalen Arrays

Bildbearbeitung: geometrische Bildoperationen

Didaktische Überlegungen:

Verwendung von Arrays

Eindimensionale Arrays wurden schon in Klasse 9 behandelt. Die SuS müssten den Umgang beherrschen. Um an die Inhalte aus Klasse 9 anzuknüpfen, werden die grundlegenden Operationen im eindimensionalen Array wiederholt und auf ein zweidimensionales Color-Array übertragen. Dabei werden zunächst nur vordefinierte Farben (z.B. Color.BLACK) genutzt, um das Farbmodell noch nicht thematisieren zu müssen. Es wird auch nicht erwähnt, dass BLACK eine statische Konstante der Klasse Color ist. Ohne auf die Details einzugehen, werden aber von Anfang an die Begriff Klasse und Objekt korrekt benutzt.

Entscheidend für die Verarbeitung von zweidimensionalen Arrays sind verschachtelte For-Schleifen, die den SuSn auch schon aus Klasse 9 bekannt sein sollten. Zum Zeichnen der optischen Täuschungen wurden sie oftmals eingesetzt. Trotzdem sollte nochmals besprochen werden, wie diese Schleifen arbeiten. Durch die Aufgabe, x und y bei den Schleifendurchgängen anzugeben, wird der Ablauf nochmals verdeutlicht.

Geometrische Bildoperationen

Der Einstieg mit einem Beispielprogramm vermeidet das aufwändige Erklären der zur Verfügung stehenden Methoden der Klasse Picture. Aus dem Programmschnipsel wird die Benutzung sofort klar. Die Struktur dieser Methode wird für alle weiteren Bildbearbeitungsoperationen übernommen. Außerdem fördert die Analyse von fremdem Code das Verständnis für Algorithmen, ohne auf Syntaxprobleme zu stoßen.

Bei den geometrischen Bildoperationen (hier horizontale Spiegelung) wären zwei Vorgehensweisen denkbar:

1. x und y geben die Position des neuen Pixels an:
   ```
   pixelNeu[x][y] = pixel[(breite-1)-x][y];
   ```
2. x und y geben die Position des Pixels im Originalbild an.
   ```
   pixelNeu[(breite-1)-x][y] = pixel[x][y];
   ```

Variante 2 passt besser zu der Vorstellung, dass die alten Pixel auf eine neue Positionen verschoben werden. Dies setzt aber voraus, dass jedes Pixel auf genau eine neue Position abgebildet wird. Das ist bei der oben zu sehenden Spiegelung der Fall, aber im Allgemeinen nicht. Bei einer Vergrößerung oder Verkleinerung des Bildes oder auch beim Drehen eines Bildes gilt dies nicht. Dort muss für jedes Pixel entschieden werden, wie die Farbe ermittelt wird. Das kann z.B. mit nearest Neighbor-Verfahren oder mit bilinearer Interpolation geschehen. Daher wird hier von Anfang an die Variante 1 bevorzugt. Durch geeignete Aufgaben sollen die SuS sich diese Zusammenhänge selbst erschließen. Insbesondere bei den Drehungen ist dies nicht ganz so einfach. Daher wird zunächst die Spiegelung behandelt. Hier wird die Idee "(breite-1)-x" schon angelegt.

Implementation in BlueJ

In BlueJ arbeiten die SuS mit einem vorgegebenen Projekt. Dieses Projekt stellt die Klassen Beispielbild und GeometrischeBildoperationen bereit. Beispielbild ist eine Unterklasse von Picture. Die Klassen Picture, PictureViewer, HSB und einige weitere hier nicht benötigte Klassen sind ein Paket von Klassen die von der ZPG IMP bereit gestellt werden (Unterordner IMP).

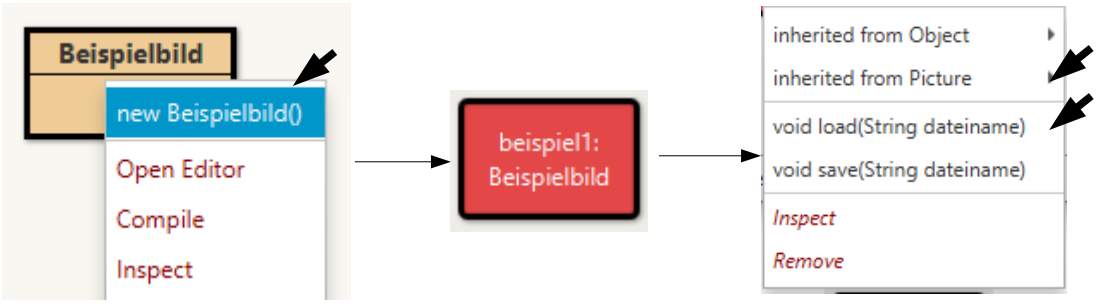
Die Klasse Picture wurde schon in Klasse 9 für das Zeichnen von optischen Täuschungen eingesetzt und stellt die meisten Zeichenbefehle zur Verfügung, die es auch in Processing gibt. Zusätzlich gibt es die Methoden getPixelArray und setPixelArray, die das Bild als zweidimensonales Array zurückgeben bzw. das Bild gemäß dem Pixelarray setzen. In Processing liefern die entsprechenden Methoden ein eindimensionales Pixelarray. Sollte weiterhin Processing eingesetzt werden, müsste den SuSn eine Methode zum Umwandeln in ein zweidimensionales Array zur Verfügung gestellt werden.

Zusätzlich gibt es eine Klasse PictureViewer, die ein Objekt der Klasse Picture in einem Frame darstellen kann. Bei Erstellen eines neuen Bildes wird automatisch ein PictureViewer-Objekt erzeugt und das Bild angezeigt. Durch das Scrollrad kann das Bild vergrößert oder verkleinert dargestellt werden. Später wird bei der Erstellung eines graphischen User-Interfaces (GUI) eine veränderte PictureViewer-Klasse verwendet, die ein Bild in einer GUI anzeigen kann, ansonsten aber die gleichen Methoden bereitstellt.

Die Klasse HSB vereinfacht die Umrechnung von Color-Objekten in HSB-Werte (vgl. Farbmodell).

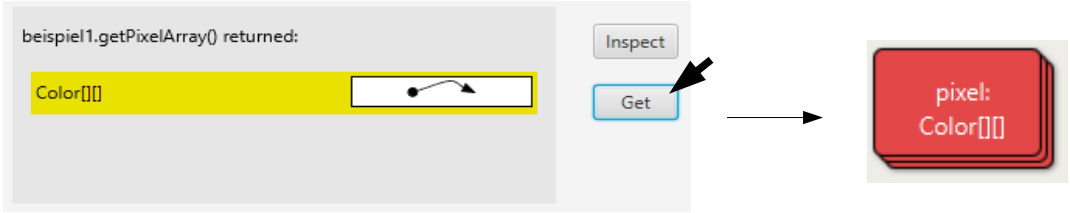
Die SuS arbeiten mit BlueJ nach folgendem Schema:

Sie erzeugen ein Objekt der Klasse Beispielbild (Rechtsklick auf den Klassennamen). Dadurch entsteht ein Objekt der Klasse Beispielbild.



Bei diesem Objekt kann man mit der rechten Maustaste weitere Methoden aufrufen. Zum Beispiel kann man mit load(String dateiname) ein anderes Bild laden (Achtung: Strings mit Anführungszeichen eingeben: z.B. "beispiel2.png", alle Bilder müssen im Unterordner images liegen). Unter "inherited from Picture" finden sich weitere Methoden der Klasse Picture, mit denen man z.B. auf dem Bild malen oder seine Größe herausfinden kann. Außerdem ist dort auch die Methode getPixelArray().

Mit "Get" kann man dieses zweidimensionale Array als Objekt bekommen. Inspiziert man dieses Objekt kann man gut den Aufbau eines zweidimensionalen Arrays als Array von Arrays erkennen.

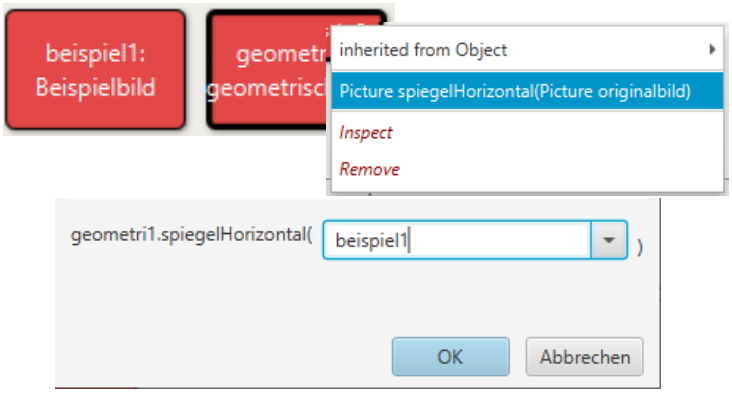


Man kann auch mit dem Codepad von BlueJ (Menü View -> Show Code Pad) gezielt einzelne Pixel verändern, wenn man zunächst mit dem Befehl import java.awt.Color; die Color-Klasse importiert hat:

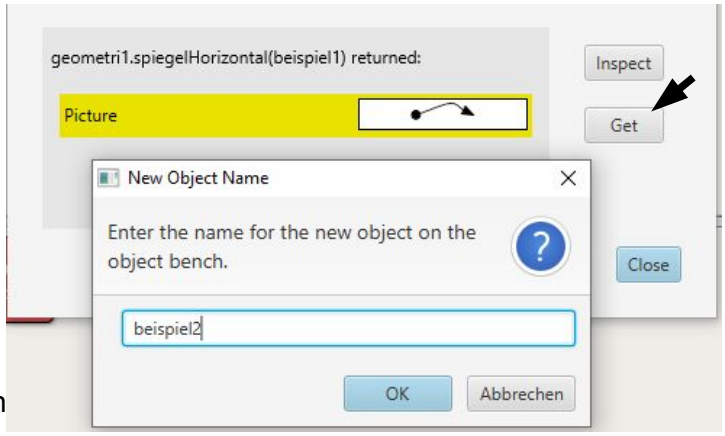
```
import java.awt.Color;
pixel[2][3] = new Color(231, 23,123);
```

Setzt man im Beispielbild die Pixel mit setPixelArray neu, dann sieht man die vorgenommenen Änderungen. Eventuell kann man dies nach den ersten Aufgaben zu den Arrays machen.

Normalerweise werden aber keine einzelnen Pixel verändert, sondern Algorithmen zur Bildbearbeitung implementiert, die sich auf das ganze Bild auswirken.



In der Klasse "geometrischeBildoperationen" ist ein erstes Beispiel (Spiegelung) implementiert. Erzeugt man ein Objekt der Klasse Beispielbild und ein Objekt der Klasse geometrische Bildoperationen, kann man die implementierte Operation auf das Bild anwenden (Rechtsklick auf geometri1).



Als Parameter gibt man den Namen des Bild-Objektes (beispiel1) an. Das Ergebnis wird sofort angezeigt. Möchte man mit diesem Bild weitere Operationen durchführen, gibt man ihm mit "Get" einem Namen (z.B. beispiel2). Das neue Objekt taucht in der Objektleiste auf.

Nachdem die SuS einige Überlegungen zu den Pixelpositionen angestellt haben, implementieren sie weitere geometrische Bildoperationen.

Am Ende schließt sich die Implementation einer Methode an, die die neuen Methoden nutzt, um eine weitere Fähigkeit (Drehen um 180°) zu realisieren. Hierzu müssen die in BlueJ von Hand durchgeführten Operationen in Quelltext umgesetzt werden.

2. Doppelstunde: RGB-Farbmodell

Ziele

Java: Umgang mit der Klasse Color

Bildbearbeitung: RBG-Farbmodell

Didaktische Überlegungen:

Auch die zweite Doppelstunde startet mit theoretischen Überlegungen. Anhand der Präsentation RGB_Farbmodell.odp gibt der Lehrer eine Einführung in das RGB-Farbmodell. Das HSB-Farbmodell ist eine mögliche Ergänzung.

Die SuS lernen mit der Java Klasse Color umzugehen. Wichtig ist dabei, dass die Farbanteile eines einmal erzeugten Farbobjektes nie mehr geändert werden können. Möchte man die Farbe verändern, muss man ein neues Farbobjekt erzeugen.

Die Implementation von Methoden zur Farbmanipulation ist analog zu der von Methoden aus der ersten Doppelstunde. Daher können die SuS sie selbstständig erstellen. Die verschiedenen Algorithmen der Graustufenberechnung, zum Invertieren und Farbtauschen können als Differenzierung eingesetzt werden. Später können ggf. mehrere SuS ihre Methoden in einer gemeinsamen GUI kombinieren. Die Methode farbaenderung sollten alle SuS kennenlernen, da sie später noch benötigt wird, um in der GUI einen Eingabedialog einzuführen.

Sehr schnelle SuS können außerdem an dieser Stelle auch schon die Methoden zur Bildkombination erstellen.

3. Doppelstunde: Faltungen

Ziele

Java: Vertiefung des Umgangs mit Schleifen und zweidimensionalen Arrays

Bildbearbeitung: Berechnung von Faltungen, Untersuchung von vorgegebenen Faltungsfiltern

Didaktische Überlegungen:

Die implementierten Algorithmen werden nun komplexer, da sie nicht nur ein Pixel berücksichtigen, sondern die Umgebung eines Pixels. Daher sollen die SuS zunächst die Berechnung der Faltung durchschauen, bevor die Implementation in Angriff genommen wird. Dazu ist die Faltung mit einer 3x3 Matrix auf einigen Beispielbildern in einem Tabellenkalkulationsblatt als Formel hinterlegt. Damit kann sehr schnell die Wirkung eines Filters untersucht werden. Dabei bleibt der Rand zunächst unberücksichtigt.

Die SuS sollten verstehen, dass die Summe der Filtergewichte 1 sein muss, wenn die Gesamthelligkeit nicht verändert werden soll. Die Wirkung von Mittelwertbildung bzw. Kantenverstärkung durch negative Gewichte muss deutlich werden.

Implementation:

Da die Anwendung eines Filters für sehr viele Zwecke eingesetzt werden kann, wird zunächst eine allgemeine Methode erstellt. Diese erfordert die Verschachtelung von vier For-Schleifen. Das ist nicht einfach. Die SuS erhalten daher ein Puzzle für den Quelltext des Algorithmus. Um

die SuS zu animieren, sich über die einzelnen Abschnitte des gegebenen Quelltextes Gedanken zu machen, müssen sie die Codeabschnitte kommentieren.

Danach kann das Programm in BlueJ übertragen werden und dann mit verschiedenen Faltungsfiltern aufgerufen werden.

Als Differenzierungsmöglichkeiten für besonders schnelle SuS stehen Arbeitsblätter für die Verwendung größerer Faltungsfilter inklusive verschiedener Algorithmen zur Randbehandlung und die Berechnung eines Gauß-Filters bereit.

4. Doppelstunde: Graphische Benutzeroberflächen

Ziele

Java: Erstellung einer Oberfläche mit dem Scene Builder, Erstellen einer Controller-Klasse,
Umgang mit Klassen und Objekten

Konzeptwissen: Trennung in Model-View und Controller (MVC-Modell),
Layout-Konzept mit automatisch wachsenden Elementen.

Didaktische Überlegungen:

In den wenigen zur Verfügung stehenden Stunden ist es unmöglich, die Erstellung von graphischen Benutzeroberflächen (GUI) in Java erschöpfend zu behandeln. Daher wird hier eine kurze Einführung in die GUI-Erstellung und Implementation eines Controllers anhand eines einzigen Beispiels - des Bildbearbeitungsprogramms gegeben. Dies kann als Grundlage für eine weitere Beschäftigung der SuS mit der GUI-Programmierung außerhalb des Unterrichts dienen. Der Informatikunterricht in der Schule darf die GUI-Programmierung nicht in den Vordergrund stellen, da hier viele Spezialitäten der Programmiersprache statt grundlegenden Konzepten erlernt werden.

Da es sehr schwer ist, ausreichend präzise das Vorgehen im Scene Builder zu beschreiben, erhalten die SuS Videos, die die entscheidenden Schritte vorführen. Anhand dieser Videos sollten die SuS selbstständig die GUI erstellen können. Dies kann auch als vorbereitende Hausaufgabe erfolgen, um für die Programmierung mehr Unterrichtszeit zur Verfügung zu haben.

Die Erstellung des Controllers findet dann im Unterricht statt. Auch hier steht ein Video als Anleitung zur Verfügung. Sobald die erste Action-Methode implementiert wurde, können weitere nach dem gleichen Schema erstellt werden. Wichtig ist immer das gleiche Vorgehen: Auslesen der notwendigen GUI-Elemente, Berechnungen durchführen und dann die Ergebnisse in der GUI anzuzeigen[1].

Da die erstellten Bildbearbeitungsalgorithmen in eigenen Klassen implementiert sind, ist es sehr einfach, von verschiedenen SuS erstellete Klassen in einer einzigen GUI zu verwenden. Um ein recht komplexes Bildbearbeitungsprogramm mit vielen Operationen zu erstellen, ist es daher möglich, die SuS kollaborativ an einem Programm arbeiten zu lassen.

Als Differenzierung bietet es sich hier an, dass schnellere SuS weitere Bildbearbeitungsmethoden implementieren und in ihre GUI einbinden.

1 Das MVC-Modell sieht eigentlich vor, dass die GUI-Elemente als "Beobachter" von Datenobjekten eingetragen werden. Sobald sich ein Datenobjekt ändert, informiert es alle eingetragenen Beobachter darüber, dass eine Änderung stattgefunden hat. Das View-Element (GUI-Element) sollte daraufhin die benötigten Daten vom Datenobjekt abfragen. In der Schule muss dieses (aufwändige) Verfahren aber nicht umgesetzt werden.

5. Doppelstunde: Graphische Benutzeroberflächen

Ziele

Java: Auslesen von Werten aus GUI-Elementen

Didaktische Überlegungen:

In der 4. Doppelstunde wurden vom Benutzer Eingaben nur in Form der Auswahl eines Menüpunktes oder Drücken eines Buttons verlangt. Üblicherweise muss er aber auch Werte in Eingabefelder eintragen. Der Controller muss diese Werte dann auslesen und verarbeiten.

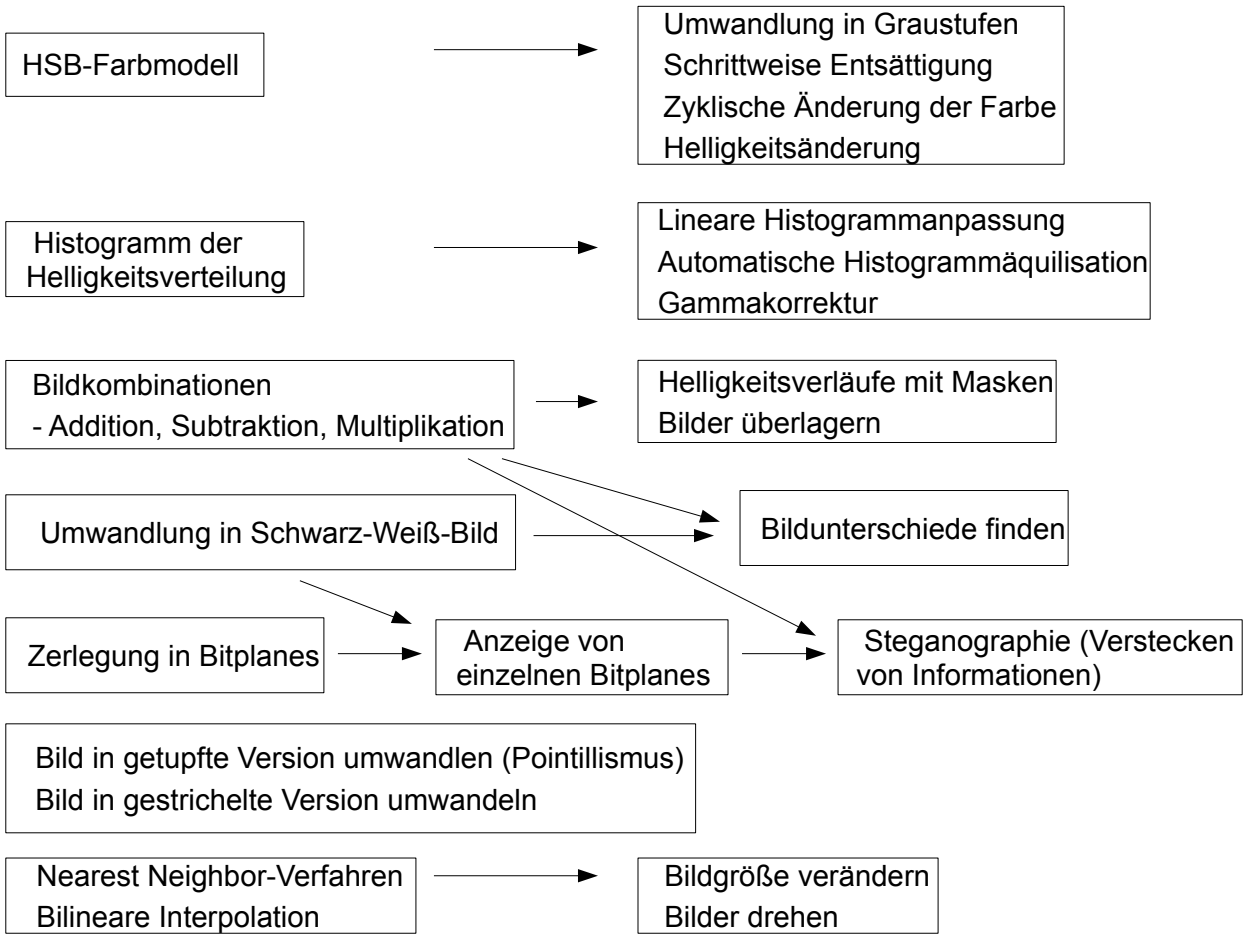
Dies wird in dieser Stunde mit der Methode farbaenderung(...) genutzt, um dem Benutzer die Möglichkeit zu geben, eine Bildverbesserung durch eine Anpassung der drei Farbkomponenten durchzuführen.

Neue Konzepte kommen kaum hinzu. Die Umsetzung erfordert allerdings wieder einige Java-spezifischen Befehle, die von den SuSn nur übernommen werden können. Das Konzept ist so angelegt, dass weitere ähnliche Dialoge analog eingefügt werden können und das Bildbearbeitungsprogramm daher sehr gut erweitert werden kann.

Die SuS sollten als Abschluss einen weiteren Bildbearbeitungsalgorithmus implementieren, ein Optionsmenü erstellen und den Algorithmus in der GUI nutzen. Dabei können die SuS durchaus sehr unterschiedliche Algorithmen umsetzen. Eine Auswahl steht als Arbeitsblätter zur Verfügung. Da sie unterschiedlich schwer sind, ist auch hier wieder eine Differenzierung möglich.

Erweiterung: Weitere Bildbearbeitungsalgorithmen

Es gibt eine Vielzahl von weiteren Algorithmen zur Bildbearbeitung. Hier wird nur ein kleiner Teil vorgestellt. Relativ einfach, aber trotzdem mit optisch ansprechenden Ergebnissen kann das Bild in eine gestrichelte oder gepunktete Variante umgewandelt werden. Auch die Algorithmen mit dem HSB-Farbmodell können leicht implementiert werden, da sie den Algorithmen mit dem RGB-Farbmodell recht ähnlich sind. Man erkennt dabei aber, dass sich das HSB-Farbmodell für manche Anwendungen deutlich besser eignet. Auch Bildkombinationen von zwei Bildern lassen sich recht leicht implementieren und als Voraussetzung für andere Anwendungen verwenden.



Histogramme sind die Grundlage für viele Strategien zur Bildverbesserung. Eine ausgewogene Verteilung der Helligkeiten sorgt für bessere optische Eindrücke. Für die Anpassung der Verteilung stehen verschiedene Algorithmen mit unterschiedlichem Schwierigkeitsgrad bereit.

Steganographie ist sehr faszinierend. Es ist möglich, Texte in einem Bild zu verstecken, ohne dass einem Betrachter überhaupt auffällt, dass ein Text im Bild enthalten ist. Damit lassen sich z.B. Wasserzeichen realisieren. Allerdings sind dafür einige Voraussetzungen notwendig.

Das Ändern der Bildgröße erscheint auf den ersten Blick nicht schwierig. Es ändert sich dabei aber die Anzahl der Pixel. Daher stellt sich die Frage, wie sich die Pixelanzahl reduzieren oder vergrößern lässt. Dies ist gar nicht so einfach mit guter Qualität zu bewerkstelligen. Nearest-Neighbor (recht einfach, aber schlechte Qualität) und Bilineare Interpolation (bessere Qualität, aber komplexer) lassen sich in der Schule noch behandeln. Noch bessere Verfahren (z.B. Bikubische Interpolation) sind nicht mehr zugänglich.

Alle Erweiterungen sind als Ergänzung zum normalen Unterricht zu verstehen und nicht Voraussetzung, um den Bildungsplan zu erfüllen. Als Lehrer sollten Sie je nach zur Verfügung stehender Zeit oder des Einsatzgebietes auswählen, ob und wenn ja welche weiteren Algorithmen Sie behandeln wollen. Für jede Aufgabe stehen Arbeitsblätter zur Verfügung. Die Algorithmen sollten zunächst in der Umgebung implementiert werden, in der die Klasse Beispielbild zur Verfügung steht. Danach kopiert man sie in das GUI-Projekt. Die inhaltlichen Abhängigkeiten sind in der Übersicht oben dargestellt.