Übung Nr. 2:

Dem Programm `DIPintro` unter http://homepages.thm.de/christ/ sollen im folgenden weitere Funktionalitäten zugefügt werden (insg. ca. 20 Zeilen).

Eine Pixeloperation, die das Menü vorsieht, ist die Rückführung des Bildes auf seine ursprüngliche Größe durch Drücken von 'r', falls diese zuvor mit der Maus (durch Größenveränderung des Fensters) geändert wurde. Zuvor soll sichergestellt werden, daß sich die Maße des Bildes und des Fensters nur verzerrungsfrei verändert werden: Bei jedem Versuch einer Größenänderung am Fenster soll die Seite mit der prozentual kleineren Änderung angenommen, die andere unter Wahrung des Seitenverhältnisses im Original angepaßt werden.

Dafür ist die Callback-Funktion `Reshape()` vorbereitet. Dort soll der korrekte Wert für eine einheitliche Maßstabsänderung (globale Variable `zoom`) in x- und y-Richtung berechnet werden, um eine Verzerrung des jeweiligen Motivs zu vermeiden. Um dem Fenster exakt die neuen Maße des Bildes zu geben, müssen in `Reshape()` die von der Mausaktivität übermittelten Werte der aktuellen Fensterbreite und -höhe (lokale Variablen `w` und `h`) in Abhängigkeit von `zoom` geändert werden, bevor sie zur Festlegung der endgültigen Fenstermaße (über die Variablen `winWid` und `winHig`) verwendet werden.

Für die Bild-Darstellung im neuen Maßstab wird die OpenGL-Funktion `glPixelZoom()` eingesetzt. Wie ihr Name besagt (und mit der ausführbaren Musterlösung im `Demo`-Verzeichnis festzustellen ist), besteht ihre Wirkung vordergründig darin, daß Pixel größer dargestellt werden[1]; aber sie hat auch integriertes Anti-Aliasing, damit stark vergrößerte Bilder nicht „verpixelt“ aussehen.

Die entsprechende Stelle in `DIPintro.c` ist mit einer Präprozessor-Direktive (`#ifdef MORE_DIP`) gekennzeichnet.

1. Teilaufgabe: Die Funktion `Reshape()` ist um die Funktionalität einer verzerrungsfreien, fensterfüllenden Darstellung zu ergänzen.

Die übrigen Aufgaben zu dieser Übung beziehen sich auf Funktionen in den Dateien `DIPhisto.c`. und `DIPhisto.h`. und sind mit `MORE_HIST` markiert

Als eine Grundfunktion zur Bildverbesserung soll der Kontrast des aktuellen Bildes maximiert werden (Menüpunkt 'o'). Dazu werden zunächst die Helligkeitswerte ermittelt, die im Bild überhaupt vorkommen. Die hierfür zuständige Funktion `picStats()` soll nach evtl. notwendigen Initialisierungen die Farbkomponenten sämtlicher Pixel durchgehen und die über die Parameterliste angeforderten Größen ermitteln. Es handelt sich um den minimal (`min`) und den maximal (`Max`) im Bild vorkommenden Helligkeitswert, um die Auftrittshäufigkeit des am häufigsten im Bild vorkommenden Wertes (`maxNpix`) und das Histogramm der drei Farbkomponenten, die hier immer abgefragt werden können, weil alle Bilder im RGBA-Format gespeichert werden.

[1] Es sollte beachtet werden, daß die hier besprochene Größenänderung eines Bildes keine Änderung der Bild-Maße zur Folge hat: Die Anzahl der Pixel eines Bildes bleibt unverändert; sie wird lediglich anders (größer oder kleiner) dargestellt. Für eine veränderte Bildauflösung müßte man mit diesem Programm ein Bild auf die gewünschte Größe bringen und es dann so sichern (Menüpunkt 's').

Zur Codierung des Histogramms dient (in der Funktion `Hist()`) eine einfach indizierte Variable (`hist[]`) mit 768 (= 3 x 256) Einträgen, die nacheinander die absoluten Häufigkeiten der einzelnen Helligkeitswerte für Rot, Grün und Blau aufnehmen sollen; deren grafische Darstellung ist das Histogramm.

Zur Kontrastoptimierung werden die o.a. Häufigkeiten nicht benötigt. Die für sie sonst aufgewendeten Speicher und Rechenzeit sollen eingespart werden können, wenn an `picStats()` über die Parameterliste für `hist` ein `NULL`-Zeiger übergeben wird; dann sollen Wertzuweisungen und andere Zugriffe auf den (dann nicht reservierten) Speicher unterbleiben.

2. Teilaufgabe: In der Funktion `picStats()` sind zunächst die evtl. benötigten Maßnahmen (Deklaration zusätzlicher Variablen, Initialisierungen o.ä.) vorzunehmen; dann sollen in den vorbereiteten geschachtelten Schleifen die über die Parameterliste angeforderten Werte ermittelt werden. [2]

Mit den von `picStats()` gelieferten Werten kann der Kontrast des geladenen Bildes dahingehend optimiert werden, daß seine Helligkeitswerte die gesamte Spannbreite (0 bis 255) abdecken (Abb. 1).

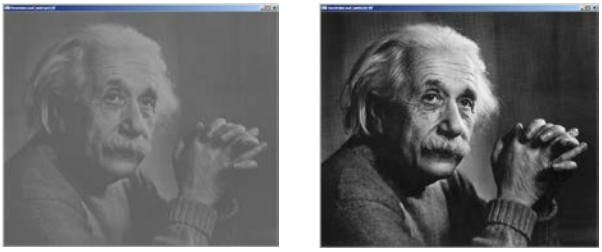

Abb. 1: Bild vor und nach der Kontrastoptimierung

Das soll in der Funktion `optiCont()` geschehen. Darin wird zunächst eine Zuordnungstabelle (Look-Up Table, LUT) aufgebaut. Sie dient dazu, anhand einer Tabelle jedem Pixel einen neuen Wert zuzuweisen, ohne jedes Mal die dazugehörige mathematische Transformation anwenden zu müssen.

Im ausgelieferten Zustand ist die Funktion `optiCont()` nur als schemenhaftes Gerüst vorhanden, wozu der eigentliche Code (weniger als 10 Zeilen) zu implementieren ist.

3. Teilaufgabe: Ergänzung der Funktion `optiCont()` durch die Erstellung und Anwendung der Zuordnungstabelle zwischen alten (Original-) und neuen (transformierten) Farb- und Helligkeitswerten

Durch Drücken von 'h' soll schließlich das Histogramm des aktuellen Bildes grafisch dargestellt werden, also die Anteile der verwendeten unter den 256 Intensitätswerte, in getrennten Hilfsfenstern für Rot, Grün und Blau; dabei wird der Bildausschnitt (AOI), aus dem das Histogramm gebildet wurde, durch ein gelbes Rechteck markiert. (Das ist in der aktuellen Fassung das gesamte Bild.)

[2] Während bei Schwarzweißbildern die Parameter von `picStats()` eindeutig zu verstehen sind, sollte bei der Behandlung dieser Teilaufgabe in Bezug auf Farbbilder zunächst geklärt und entschieden werden, welche Parameter `picStats()` liefern sollte.

Die dafür vorgesehene Funktion `Hist()` erhält von `picStats()` die benötigten Werte und paßt den Skalierungsfaktor `scale` so an, daß die höchste darzustellende Häufigkeit von Grauwerten mit der Länge `HIST_DIM` im quadratischen Fenster mit Kantenlänge (`HIST_DIM+2*border`) gezeichnet wird. In `Hist()` ist zwar die Markierung der AOI korrekt eingesetzt; das Diagramm selbst muß aber noch an der dafür vorbereiteten Stelle codiert werden.

4. Teilaufgabe: Ergänzung der Funktion `Hist()` durch die korrekte Wertzuweisung für `scale` und die Codierung der Balken, die das Histogramm visualisieren

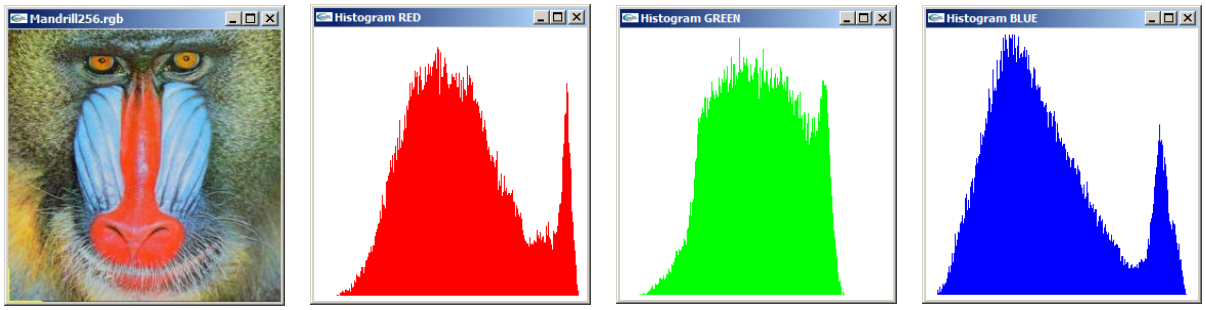

Abb. 2: Bild und Histogramm seiner Farbkomponenten

`Hist()` (und damit `DIPintro.exe`) zeigt nun das Verhalten, das man von ihm erwartet (vgl. Abb. 2). Bei genauem Hinsehen erkennt man jedoch, daß beim Compilieren Warnungen für die Funktion `InitAuxWins()` entstehen, die jedoch selbsterklärend sind.

5. Teilaufgabe: Die Parameterliste von `InitAuxWins()` ist (bei ihrer Deklaration und/oder bei ihrer Definition) so abzuändern, daß die Compiler-Warnungen entfallen.

Da diese letzte Teilaufgabe zwar typisch für Bild- und Grafik-Programmierung, aber nicht integraler Bestandteil der Bildverarbeitung ist, sollte bei Bedarf spätestens hier die Hilfe des Autors in Anspruch genommen werden.