

## Computergrafik 1

### Abgabetermin:

Die Lösung zu diesem Übungsblatt ist bis zum Freitag den **12. Juni 2009, 12:00 Uhr s.t.** per Email abzugeben.

### Inhalt:

Dieses Blatt dient zum Bekanntmachen mit *Multiple Document Interfaces (MDI)* in *Qt*. Hierzu erzeugen Sie ein Fenster, das die Grundfunktionalitäten (*Öffnen, Speichern, Schließen, etc.*) eines gewöhnlichen Bildbearbeitungswerkzeugs (z.B. *Photoshop*) besitzt. Zusätzlich erlernen Sie das Erzeugen einfacher Bildfilter (hier *Graustufen, Invertierung* und *HSL Filterung*). Pro Aufgabe können maximal zehn (Aufgaben 22 und 23 maximal 20) Punkte erreicht werden. Zum Bestehen des Übungsblatts müssen in **JEDER** Aufgabe mindestens fünf (Aufgabe 22 und 23 mindestens 10) Punkte erreicht werden.

Geben Sie insgesamt (d.h. ein Projekt für alle Aufgaben zusammen) alle benötigten Header-, Source-, und Projektdateien (von *Qt Creator* erzeugt) mit ab, d.h. \*.h, \*.cpp und \*.pro Dateien. Abgaben die nicht kompilieren werden mit 0 Punkten bewertet. Fassen Sie alle Aufgaben zu einer zip-Datei zusammen und senden Sie diese an: cg1\_ss09@medien.ifi.lmu.de.

### Aufgabe 22 *Multiple Document Interface* – Hauptfenster (20 Punkte)

In dieser Aufgabe erzeugen Sie den ersten Teil eines sogenannten *Multiple Document Interface (MDI)*. Analog zu den Aufgaben in der 3D-Grafik wird dieses Fenster in den folgenden Aufgaben (auch über dieses Übungsblatt hinaus) weiter verwendet.

- Erzeugen Sie ein neues Projekt vom Typ *Qt4 Gui Application* und nennen Sie es `CG1_2D`. Anders als in den 3D-Aufgaben fügen Sie auf der nächsten Seite kein weiteres Modul hinzu. Auf der folgenden Seite könnten Sie nun die Namen der Hauptklassen ändern. Lassen Sie jedoch alle Einstellungen wie sie sind, entfernen aber das Häkchen bei *Generate from*, damit keine Datei vom Typ \*.ui zusätzlich erstellt wird (2 Punkte).
- Geben Sie dem Hauptfenster den Titel `Computer Graphics 1 :: 2D` mit der Funktion `setWindowTitle` im Konstruktor des Hauptfensters (2 Punkte).
- Erstellen Sie in Ihrem Hauptfenster eine Instanzvariable vom Typ `QMdiArea` und setzen Sie es als zentrales Widget mit der Funktion `setCentralWidget`. Setzen Sie zusätzlich die Größe der `QMdiArea` auf ausreichende Werte, z.B. `800px · 600px`. Erlauben Sie ferner keine Scrollbars in der `QMdiArea`. **Hinweis:** Die Funktionen `setHorizontalScrollBarPolicy` bzw. `setVerticalScrollBarPolicy` sind hier von Vorteil (4 Punkte).
- Erstellen Sie in Ihrem Hauptfenster nun noch ein Menü, mit den nachfolgenden Einträgen:
  - File:*  
Dieses Menü beinhaltet zunächst nur vier Optionen: *Open, Save, Save As...* und *Exit*. Erstellen Sie die nötigen Instanzvariablen vom Typ `QAction`. Erstellen Sie zusätzlich

auch alle Funktionen, d.h. für jeden Menüeintrag bzw. für jede `QAction` eine Funktion in Ihrem Hauptfenster. Achten Sie darauf, dass zunächst die Einträge *Save* und *Save As...* ausgegraut sein müssen, da sich zu Beginn noch keine geöffneten Dokumente in der Ansicht befinden.

- *Edit*:  
Dieses Menü beinhaltet noch keine Einträge. Die Funktionen in diesem Menü (z.B. *Undo*, *Copy*, *Paste*) werden in einem späteren Übungsblatt realisiert. Bedenken Sie diese jedoch bereits bei Ihrer Implementierung.
- *Filter*:  
Dieses Menü wird in einer späteren Aufgabe auf diesem Blatt erweitert. Auch dieses Menü sollte ausgegraut dargestellt sein, da noch kein Dokument geöffnet wurde. Fügen Sie jedoch bereits ein Untermenü *Color Filters* hinzu.
- *Window*:  
Dieses Menü enthält standardmäßig zwei Einträge: *Tile Windows* und *Cascade Windows*. Erstellen Sie auch hier die nötigen Instanzvariablen vom Typ `QAction` und zusätzlich alle benötigten Funktionen. Beachten Sie, dass zu Beginn beide Funktionen ausgegraut sein müssen. Alternativ können Sie auch das ganze Menü inaktiv setzen.

(12 Punkte)

### Aufgabe 23 *Multiple Document Interface* – Dokumentfenster

(20 Punkte)

In dieser Aufgabe erzeugen Sie nun die „Kindfenster“ des *Multiple Document Interface (MDI)*. Jedes Kindfenster ist später die Ansicht eines geöffneten (und eventuell gefilterten) Bildes. Zusätzlich erlernen Sie hier das Window-Management eines *MDI*.

- a) Erzeugen Sie sich eine Klasse namens *MdiChild*, d.h. den Header `mdichild.h` und die Source-Datei `mdichild.cpp`. Diese Klasse soll von `QScrollArea` erben. Geben Sie der Klasse nun drei Instanzvariablen: Ein Objekt vom Typ `QImage*`, das das aktuelle Bild beinhaltet, ein Objekt vom Typ `QString`, das den aktuellen Pfad speichern soll, sowie ein Objekt vom Typ `QLabel*`, das später für das Anzeigen des Bildes verantwortlich sein wird. Geben sie dieser Klasse ferner folgende Funktionen (Deklaration im Header):
- `QSize` `minimumSizeHint()`; **const** sowie `QSize` `sizeHint()`; **const**, die beide als `protected` Funktionen. Diese unterstützen später *Qt* beim Setzen der richtigen Größe für jedes *MdiChild*.
  - Getter- und Setter-Funktionen für die Instanzvariable `currentFile`. Diese Variable vom Typ `QString` beinhaltet den Pfad des aktuell geladenen Bildes.
  - Getter- und Setter-Funktionen für die Instanzvariable `image`. Diese Variable vom Typ `QImage*` beinhaltet das aktuell geladene Bild. Erstellen Sie zudem eine Funktion namens `QImage*` `getImageCopy()`, die eine **tiefe** Kopie des Bildes zurück gibt.
  - `QSize` `imageSize()`; **const**: Diese Funktion gibt die Größe des aktuellen Bildes zurück. Falls gerade kein Bild geladen ist (d.h. der Pointer ein `NULL`-Pointer ist), geben Sie einen geeigneten Wert zurück, sodass das *MdiChild* nicht zu klein angezeigt wird.

(4 Punkte)

- b) Implementieren Sie diese Funktionen nun in der Klasse *MdiChild*, d.h. in der Source-Datei `mdichild.cpp`. Im Konstruktor fügen Sie die Instanzvariable vom Typ `QLabel*` noch dem *MdiChild* hinzu, d.h., es wird später das *Widget* auf der *ScrollArea*. Setzen Sie außerdem

das *Label* horizontal sowie vertikal zentriert (**Hinweis:** Die Funktion `setAlignment()` ist hier von großem Nutzen. Die Parameter können mit einem logischen „oder“ verknüpft werden). Zusätzlich geben Sie dem Konstruktor noch ein Parameter vom Typ `QString`, der den Pfad enthält. Laden Sie nun im Konstruktor das Bild und weisen Sie es als Objekt vom Typ `QPixmap` dem zuvor erstellten `QLabel*` zu (**6 Punkte**).

c) Erstellen Sie nun das Window-Management. Bedienen Sie sich hierbei dem Tutorial zum Thema *MDI* auf der *Qt*-Homepage. Dabei gilt Folgendes:

- Wird ein Fenster geöffnet, ist es automatisch das aktive Kindfenster. Es wird ferner dem Menü *Window* hinzugefügt. Wird ein Fenster geschlossen, so wird es aus dem Menü *Window* wieder entfernt.
- Wird ein anderes Fenster im Menü *Window* ausgewählt, so wird dieses aktiv. Das aktive Fenster erhält immer ein Häkchen im Menü *Window*. Alle Operationen, d.h. *speichern* oder Bildfilter werden immer nur auf dem aktiven Fenster ausgeführt.

Implementieren Sie nun auch die beiden Menü-Funktionen *Tile Windows* und *Cascade Windows* mit Hilfe der Funktionen `tileSubWindows()` bzw. `cascadeSubWindows()`, die in der Klasse `QMdiArea` definiert sind. (**6 Punkte**).

d) Implementieren Sie nun noch das Laden und Speichern eines Bildes. Dazu müssen die **drei** Menü-Einträge *Open*, *Save* und *Save As...* umgesetzt werden. Verwenden Sie dazu die statischen Funktionen der Klasse `QFileDialog`. Erlauben Sie dem Benutzer nur Bilder vom Typ `*.bmp`, `*.png`, `*.jpg` und `*.jpeg` zu öffnen. Selbiges gilt auch für das Speichern von Bildern. Beim Öffnen sollen zudem Mehrfachselektionen erlaubt sein.

- Wird ein Bild erfolgreich geöffnet, so wird ein neues *MdiChild* erzeugt und der `QMdiArea` hinzugefügt. Werden mehrere Bilder selektiert, so werden dementsprechend viele neue Fenster erzeugt.
- Wird der Menüpunkt *Save* ausgewählt, so wird das aktive Bild unter dem Pfad gespeichert, mit dem es geöffnet wurde.
- Wird der Menüpunkt *Save As...* ausgewählt, so wird das aktive Bild unter dem Pfad gespeichert, der vom Benutzer wiederum mit einem `QFileDialog` angegeben wird.

Achten Sie darauf, dass sich die Menüs korrekt verhalten, d.h. bestimmte Funktionen müssen ausgegraut sein, wenn kein aktives Fenster existiert (**4 Punkte**).

## Aufgabe 24 Einfache Bildfilter

(10 Punkte)

In dieser Aufgabe erlernen Sie den Umgang mit einfachen (dialoglosen) Bildfiltern. Dazu implementieren Sie zwei Filter (hier hier *Graustufen* und *Invertierung*).

a) Laden Sie sich die Klasse *ImageFilter* von der Vorlesungshomepage herunter. Diese Klasse dient als Basisklasse für alle weiteren Aufgabe im Bereich der 2D-Computergrafik. Binden Sie diese Klasse in Ihr Projekt ein, und erstellen Sie zusätzlich zwei Klassen namens *GrayscaleFilter* und *InvertFilter*, die beide von *ImageFilter* erben (**2 Punkte**).

b) Implementieren Sie nun die beiden abstrakten Funktionen von *ImageFilter*:

- **QImage** `apply(QImage image)`: Diese Funktion ist die Hauptfunktion jedes Bildfilters. In dieser Funktion erzeugen Sie sich zunächst eine **tiefe** Kopie (**Hinweis:** Ein `QImage` stellt dafür die Funktion `copy` bereit). Auf dieser Kopie bearbeitet ein Filter dann das Bild (siehe nächste Teilaufgabe) und gibt diese anschließend zurück.

- **QDialog\*** `getDialog()`: Diese Funktion gibt einen Dialog für den jeweiligen Filter zurück. In dieser Aufgabe kann bei **beiden** Filtern 0 zurück gegeben werden, da weder der *GrayscaleFilter* noch der *InvertFilter* einen Dialog benötigen.

**(3 Punkte)**

c) Setzen Sie nun die eigentlichen Filterfunktionen folgendermaßen um (Beschreibung gilt für 24-bit RGB Bilder):

- *GrayscaleFilter*: Ersetzen Sie für **jeden** Farbkanal (d.h. *Rot*, *Grün* und *Blau* **jedes** Pixels mit dem dazugehörigen GRaustufenwert. Das Bild wird danach immernoch ein 24-bit Bild sein, d.h. das alle drei Farbkanäle den gleichen Wert für ein Pixel besitzen. Die Formel für die Umrechnung nach BT 709 Standard lautet:

$$P_{Value} = 0.2126 \cdot P_{Red} + 0.7152 \cdot P_{Green} + 0.0722 \cdot P_{Blue}$$

- *InvertFilter*: Drehen Sie **jeden** Farbkanal **jedes** Pixels um, d.h. bilden Sie die inversen Farbkomponenten. Dies geschieht nach der Formel:

$$P_{Invert} = 255 - P_{Original}$$

**(3 Punkte)**

d) Fügen Sie nun zwei Menüpunkte *Grayscale* und *Invert* zu dem zuvor erzeugten Untermenü *Color Filters* hinzu. Bei der Auswahl eines Menüpunktes soll dann der Filter auf das aktuell aktive Bild angewendet werden (**2 Punkte**).

## Aufgabe 25 Filterung im HSL-Farbraum

**(10 Punkte)**

In dieser Aufgabe schreiben Sie einen etwas komplexeren Bildfilter. Dieser enthält zusätzlich einen Dialog, durch den der Benutzer die Einstellungen für das Filtern vornehmen kann.

- a) Erstellen sie eine neue Klasse namens *HSLFilter*, die wiederum von *ImageFilter* erbt. Fügen Sie ferner einen neuen Menüpunkt *HSL Filter* in das *Color Filters* Menü ein (**2 Punkte**).
- b) Implementieren Sie nun die Funktion `getDialog`. Für diesen Filter muss nun ein Dialog geschaffen und zurück gegeben werden, der folgendes erfüllt:
  - Für den Wert *Hue* muss jeweils ein Textfeld für das Minimum und das Maximum erstellt werden. Die Werte sollen dabei zwischen minimal 0 und maximal 360 liegen.
  - Für die Werte *Saturation* und *Luminance* müssen ebenfalls jeweils ein Textfeld für das Minimum und das Maximum erstellt werden (d.h. insgesamt vier Textfelder). Die Werte sollen dabei zwischen minimal 0 und maximal 100 liegen.

Um einen solchen Dialog zu realisieren, müssen Sie sich eine eigene Klasse schreiben, die von `QDialog` erbt. Dieser Dialog wird dann automatisch aufgerufen, wenn ein Filter aktiviert wird. Achten Sie bei der Implementierung darauf, dass der Benutzer korrekte Werte eingibt (**4 Punkte**).

- c) Implementieren Sie nun noch die Funktion `apply`. Erzeugen Sie sich auch hier wieder eine **tiefe** Kopie des Bildes (das später auch zurückgegeben wird). Da Sie im HSL-Farbraum filtern, müssen Sie jeden Pixel aus dem RGB- in den HSL-Farbraum konvertieren. Gehen Sie dazu folgendermaßen vor:

1. Normieren Sie die R-, G- und B-Werte auf den Bereich 0 bis 1
2. Finden Sie das Minimum und das Maximum der R-, G- und B-Werte.
3.  $L$  (*Luminance*) ist der Mittelwert aus Maximum und Minimum.
4. Sind Minimum und Maximum gleich, gilt:  $S = 0$  (Saturation).
5. Falls das Minimum und Maximum ungleich sind:  
 $L < 0.5 \Rightarrow S = \frac{\max - \min}{\max + \min}$   $L \geq 0.5 \Rightarrow S = \frac{\max - \min}{2 - (\max + \min)}$
6. Nun muss unterschieden werden, welche Farbe maximal war:  
 $R = \max \Rightarrow H = \frac{G - B}{\max - \min}$   $G = \max \Rightarrow H = 2 + \frac{B - R}{\max - \min}$   $B = \max \Rightarrow H = 4 + \frac{R - G}{\max - \min}$
7.  $S$  und  $L$  sind nun im Bereich von 0 bis 1.  $H$  hingegen ist im Bereich von 0 bis 6. Beachten Sie das für die Eingaben aus dem Dialog, da diese von 0 bis 100 (bzw. 0 bis 360) reichen.

Die Filterung findet nun derart statt, dass alle Pixel, die innerhalb aller drei Kriterien (d.h. *Hue*, *Saturation* und *Luminance*) liegen unverändert gezeichnet werden, wohingegen alle Pixel, die nicht in diesen Bereichen liegen schwarz gezeichnet werden. Daher ist eine Konvertierung von HSL nach RGB nicht nötig (**4 Punkte**).