LMU LUDWIG-MAXIMILIANS-UNIVERSITÄT MÜNCHEN

Prof. Dr. Andreas Butz | Prof. Dr. Ing. Axel Hoppe

Dipl.-Medieninf. Dominikus Baur
Dipl.-Medieninf. Sebastian Boring

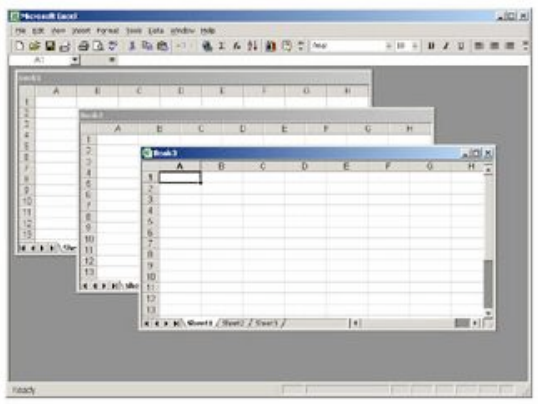
Übung: Computergrafik 1

MDIs in Qt
Farbmodelle

MDIs

Multiple Document Interfaces

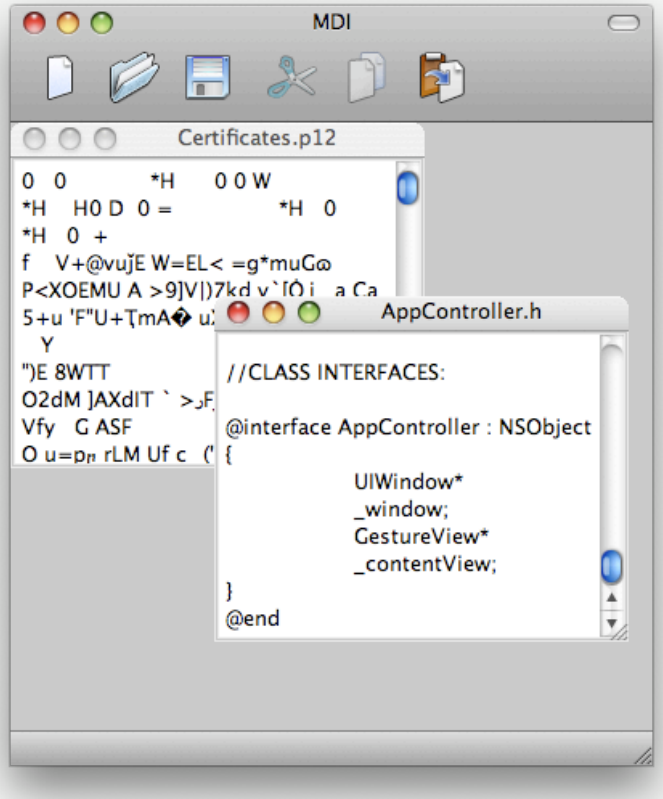
- Multiple Document Interfaces beschreiben Interfaces in denen mehrere Dokumente innerhalb eines Hauptfensters angezeigt werden (Gegenteil: Single Document Interfaces)
- Vorteile:
  - Alle geöffneten Dokumente können gleichzeitig maximiert/minimiert werden
  - Anordnungen wie “Tile” oder “Cascade” möglich
- Nachteile:
  - Möglichkeiten des Betriebssystems zur Taskübersicht können nicht genutzt werden
- Alternative: Tabbed Document Interfaces (Browser)



(Quelle: http://www.pixelcentric.net/article.php?art=docs)

MDIs in Qt

- Qt bietet direkte Unterstützung für MDIs mit der QMdiArea (ehemals QWorkspace) Klasse
- Im Folgenden:
  - Ein MDI für Textdateien in Qt
  - (Beispiel aus dem Qt SDK)
- Zu finden:
  - OSX: /Developer/Examples/Qt/mainwindows/mdi
  - Windows: C:\Qt\2009.02\qt\examples\mainwindows\mdi



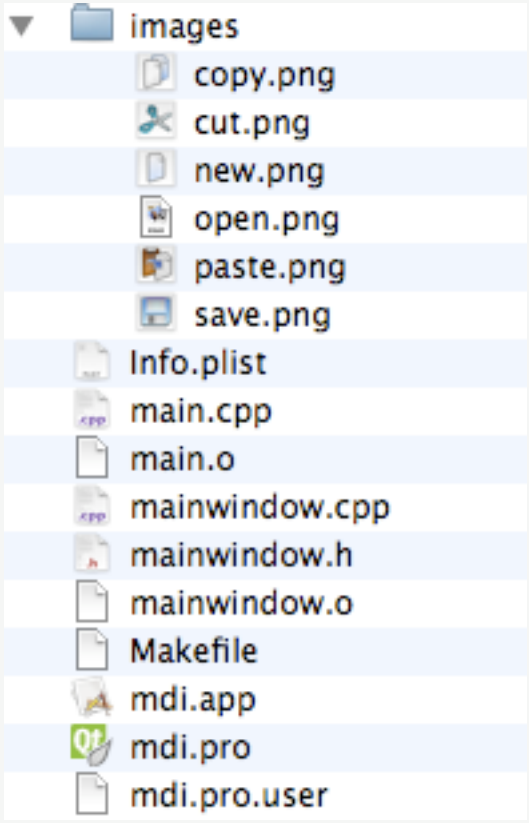
(Quelle: http://doc.trolltech.com/4.2/mainwindows-mdi.html)

MDIs in Qt

```
MainWindow::MainWindow()
{
    mdiArea = new QMdiArea;
    mdiArea->setHorizontalScrollBarPolicy(Qt::ScrollBarAsNeeded);
    mdiArea->setVerticalScrollBarPolicy(Qt::ScrollBarAsNeeded);
    setCentralWidget(mdiArea);
}
```

(Quelle: http://doc.trolltech.com/4.2/mainwindows-mdi.html)

Resources



- Nicht-Quellcodedateien können als Resourcen in Qt-Projekte eingebunden werden
- Vorteile:
  - Direkte Verfügbarkeit im Quellcode
  - Keine Probleme mit Pfadangaben

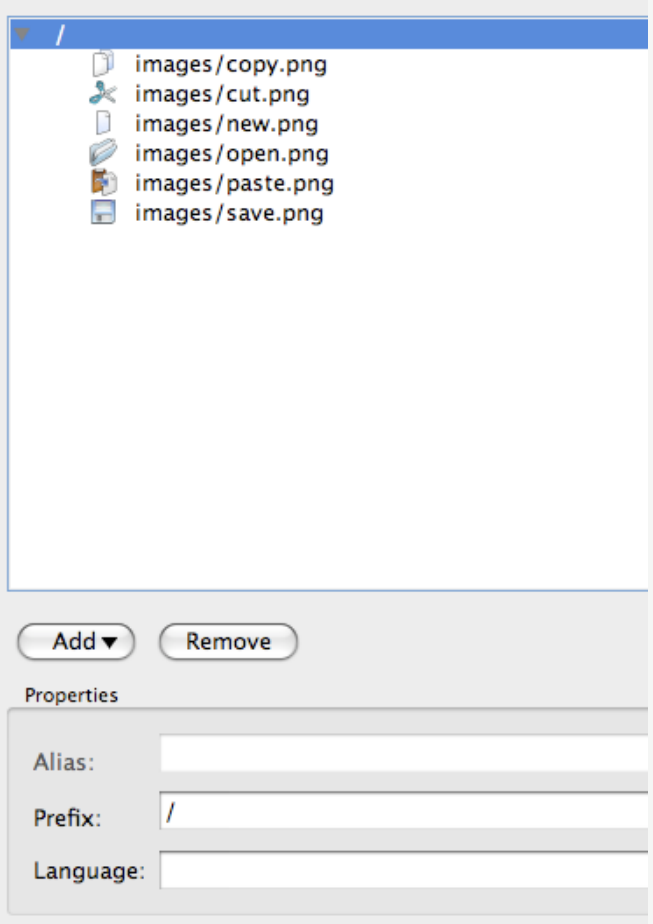
- Qt-Resourcen befinden sich in Resource Collection Files (.qrc)
- Resourcendateien müssen in der Projektbeschreibung (.pro) auftauchen:


```
HEADERS = mainwindow.h \
    mdichild.h
SOURCES = main.cpp \
    mainwindow.cpp \
    mdichild.cpp
RESOURCES = mdi.qrc
```

(Quelle: http://doc.trolltech.com/4.0/resources.html)

Resources

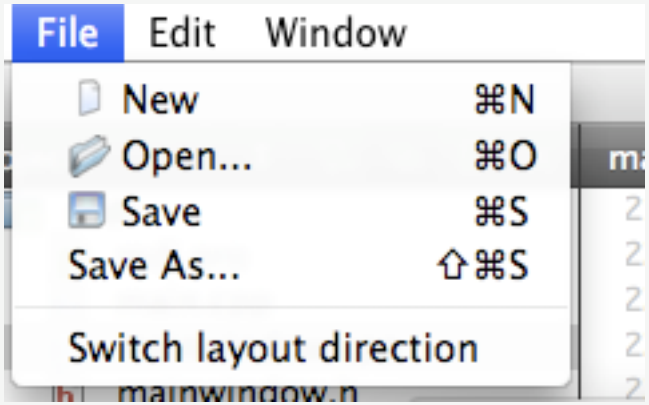
- Erstellen von Resource Collection Files in QtCreator:
  - Rechtsklick Projekt -> “Add New”
  - “Resource File”
- Ressourcen erhalten “Prefixes” um Problemen mit gleichen Dateinamen aus dem Weg zu gehen
- Zusätzlich erhält jede Ressource noch den jeweils relativen Pfad zum Projekt (hier: “images/”)
- Ressourcen werden im Quelltext mit “:Pfad/Dateiname” angesprochen
- (Beispiel: “images/copy.png” => “:/images/copy.png”)



(Quelle: http://doc.trolltech.com/4.0/resources.html)

MDIs in Qt

```
MainWindow::MainWindow()
{
    mdiArea = new QMdiArea;
    mdiArea->setHorizontalScrollBarPolicy(Qt::ScrollBarAsNeeded);
    mdiArea->setVerticalScrollBarPolicy(Qt::ScrollBarAsNeeded);
    setCentralWidget(mdiArea);

    createActions();
    createMenus();
}
```



(Quelle: http://doc.trolltech.com/4.2/mainwindows-mdi.html)

MDIs in Qt

```
void MainWindow::createActions()
{
    newAct = new QAction(QIcon(":/images/new.png"), tr("&New"), this);
    newAct->setShortcuts(QKeySequence::New);
    newAct->setStatusTip(tr("Create a new file"));
    connect(newAct, SIGNAL(triggered()), this, SLOT(newFile()));

    openAct = new QAction(QIcon(":/images/open.png"), tr("&Open..."), this);
    openAct->setShortcuts(QKeySequence::Open);
    openAct->setStatusTip(tr("Open an existing file"));
    connect(openAct, SIGNAL(triggered()), this, SLOT(open()));

    saveAct = new QAction(QIcon(":/images/save.png"), tr("&Save"), this);
    saveAct->setShortcuts(QKeySequence::Save);
    saveAct->setStatusTip(tr("Save the document to disk"));
    connect(saveAct, SIGNAL(triggered()), this, SLOT(save()));
```

● ● ●

(Quelle: http://doc.trolltech.com/4.2/mainwindows-mdi.html)

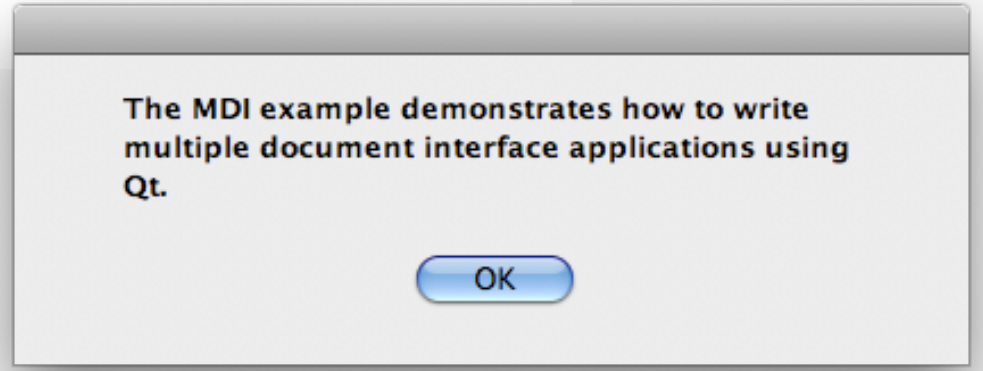
MDIs in Qt

```
void MainWindow::createMenus()
{
    fileMenu = menuBar()->addMenu(tr("&File"));
    fileMenu->addAction(newAct);
    fileMenu->addAction(openAct);
    fileMenu->addAction(saveAct);
    fileMenu->addAction(saveAsAct);
    fileMenu->addSeparator();
    QAction *action = fileMenu->addAction(tr("Switch layout direction"));
    connect(action, SIGNAL(triggered()), this, SLOT(switchLayoutDirection()));
    fileMenu->addAction(exitAct);

    editMenu = menuBar()->addMenu(tr("&Edit"));
    editMenu->addAction(cutAct);
    editMenu->addAction(copyAct);
    editMenu->addAction(pasteAct);

    windowMenu = menuBar()->addMenu(tr("&Window"));
    updateWindowMenu();
    connect(windowMenu, SIGNAL(aboutToShow()), this, SLOT(updateWindowMenu()));

    menuBar()->addSeparator();

    helpMenu = menuBar()->addMenu(tr("&Help"));
    helpMenu->addAction(aboutAct);
    helpMenu->addAction(aboutQtAct);
}
```

MDIs in Qt

```
void MainWindow::switchLayoutDirection()
{
    if (layoutDirection() == Qt::LeftToRight)
        qApp->setLayoutDirection(Qt::RightToLeft);
    else
        qApp->setLayoutDirection(Qt::LeftToRight);
}
```

```
void MainWindow::about()
{
   QMessageBox::about(this, tr("About MDI"),
            tr("The <b>MDI</b> example demonstrates how to write multiple "
               "document interface applications using Qt."));
}
```



MDIs in Qt

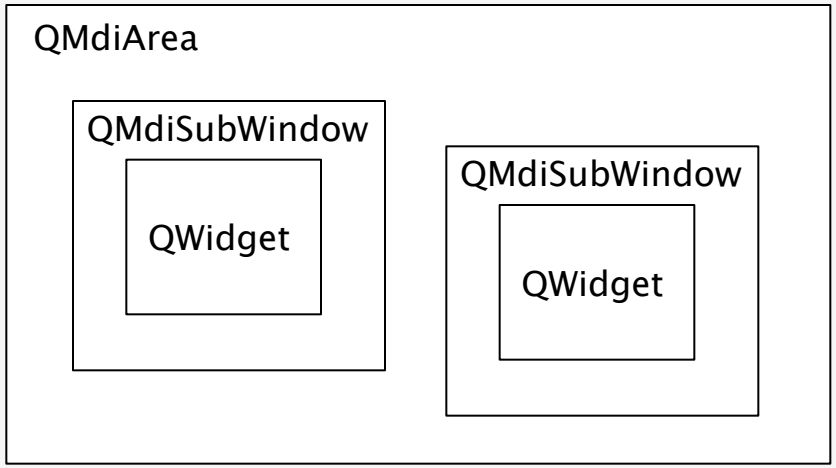
mainwindow.cpp

```
void MainWindow::newFile()
{
    MdiChild *child = createMdiChild();
    child->newFile();
    child->show();
}

void MainWindow::open()
{
    QString fileName = QFileDialog::getOpenFileName(this);
    if (!fileName.isEmpty()) {
        QMdiSubWindow *existing = findMdiChild(fileName);
        if (existing) {
            mdiArea->setActiveSubWindow(existing);
            return;
        }

        MdiChild *child = createMdiChild();
        if (child->loadFile(fileName)) {
            statusBar()->showMessage(tr("File loaded"), 2000);
            child->show();
        } else {
            child->close();
        }
    }
}
```

QMdiArea

- QMdiArea ist ein Window Manager der QMdiSubWindows als Objekte enthält
- Jedes dieser QMdiSubWindows enthält wiederum ein beliebiges QWidget
- Durch Aufruf von QMdiArea::addSubWindow(QWidget*) wird das übergebene Widget automatisch in ein QMdiSubWindow verpackt und in die QMdiArea eingefügt
- QMdiSubWindows können auch manuell erstellt, mit setWidget(QWidget*) mit einem Widget versehen und ebenfalls mit addSubWindow() hinzugefügt werden

- Wir arbeiten im Folgenden mit QTextEdit Widgets

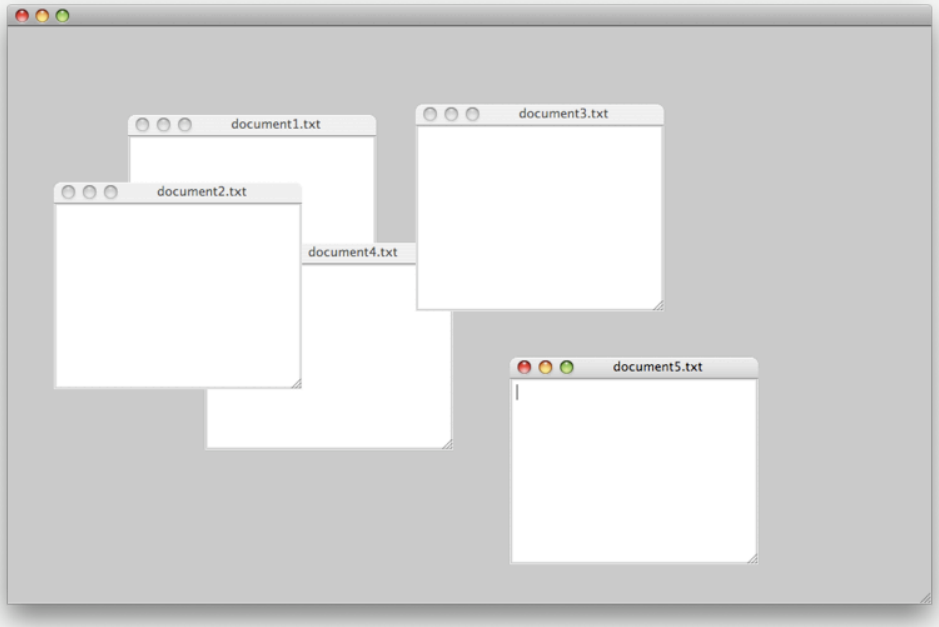


(Quelle: http://doc.trolltech.com/4.5/qmdiarea.html)

MDIs in Qt

mdichild.h

```
class MdiChild : public QTextEdit
{
    Q_OBJECT

public:
    MdiChild();

    void newFile();
    bool loadFile(const QString &fileName);
    bool save();
    bool saveAs();
    bool saveFile(const QString &fileName);
    QString userFriendlyCurrentFile();
    QString currentFile() { return curFile; }

protected:
    void closeEvent(QCloseEvent *event);

private slots:
    void documentWasModified();

private:
    bool maybeSave();
    void setCurrentFile(const QString &fileName);
    QString strippedName(const QString &fullFileName);

    QString curFile;
    bool isUntitled;
};
```



MDIs in Qt

mdichild.cpp

```
void MdiChild::newFile()
{
    static int sequenceNumber = 1;

    isUntitled = true;
    curFile = tr("document%1.txt").arg(sequenceNumber++);
    setWindowTitle(curFile + "[*]");

    connect(document(), SIGNAL(contentsChanged()),
            this, SLOT(documentWasModified()));
}
```

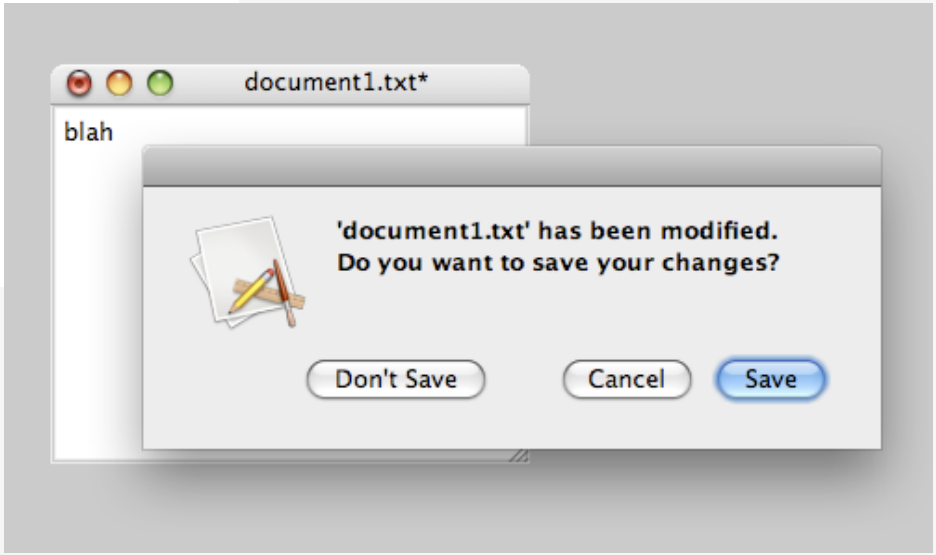
```
void MdiChild::documentWasModified()
{
    setWindowModified(document()->isModified());
}
```

```
void MdiChild::closeEvent(QCloseEvent *event)
{
    if (maybeSave()) {
        event->accept();
    } else {
        event->ignore();
    }
}
```

MDIs in Qt

mdichild.cpp

```
bool MdiChild::maybeSave()
{
    if (document()->isModified()) {
        QMessageBox::StandardButton ret;
        ret = QMessageBox::warning(this, tr("MDI"),
                     tr("'%1' has been modified.\n"
                        "Do you want to save your changes?")
                     .arg(userFriendlyCurrentFile()),
                     QMessageBox::Save | QMessageBox::Discard
                     | QMessageBox::Cancel);
        if (ret == QMessageBox::Save)
            return save();
        else if (ret == QMessageBox::Cancel)
            return false;
    }
    return true;
}
```



MDIs in Qt

mdichild.cpp

```
bool MdiChild::saveFile(const QString &fileName)
{
    QFile file(fileName);
    if (!file.open(QFile::WriteOnly | QFile::Text)) {
        QMessageBox::warning(this, tr("MDI"),
                             tr("Cannot write file %1:\n%2.")
                             .arg(fileName)
                             .arg(file.errorString()));
        return false;
    }

    QTextStream out(&file);
    QApplication::setOverrideCursor(Qt::WaitCursor);
    out << toPlainText();
    QApplication::restoreOverrideCursor();

    setCurrentFile(fileName);
    return true;
}
```

```
void MdiChild::setCurrentFile(const QString &fileName)
{
    curFile = QFileInfo(fileName).canonicalFilePath();
    isUntitled = false;
    document()->setModified(false);
    setWindowModified(false);
    setWindowTitle(userFriendlyCurrentFile() + "[*]");
}
```

MDIs in Qt

mdichild.cpp

```
bool MdiChild::loadFile(const QString &fileName)
{
    QFile file(fileName);
    if (!file.open(QFile::ReadOnly | QFile::Text)) {
        QMessageBox::warning(this, tr("MDI"),
                             tr("Cannot read file %1:\n%2.")
                             .arg(fileName)
                             .arg(file.errorString()));
        return false;
    }

    QTextStream in(&file);
    QApplication::setOverrideCursor(Qt::WaitCursor);
    setPlainText(in.readAll());
    QApplication::restoreOverrideCursor();

    setCurrentFile(fileName);

    connect(document(), SIGNAL(contentsChanged()),
            this, SLOT(documentWasModified()));

    return true;
}
```

QSignalMapper

- QSignalMapper ist eine Klasse, die Signale von identifizierbaren Klassen zusammenfügt und an eine andere Methode weitergibt
- Dabei lassen sich auch zusätzliche Parameter, die an den Zielslot weitergereicht werden, definieren (mit setMapping)
- QSignalMapper hat ein Signal mapped, das entweder int, QString, QWidget oder QObject übergibt

```
windowMapper = new QSignalMapper(this);
connect(windowMapper, SIGNAL(mapped(QWidget *)),
        this, SLOT(setActiveSubWindow(QWidget *)));
```

```
connect(action, SIGNAL(triggered()), windowMapper, SLOT(map()));
windowMapper->setMapping(action, windows.at(i));
```

MDIs in Qt

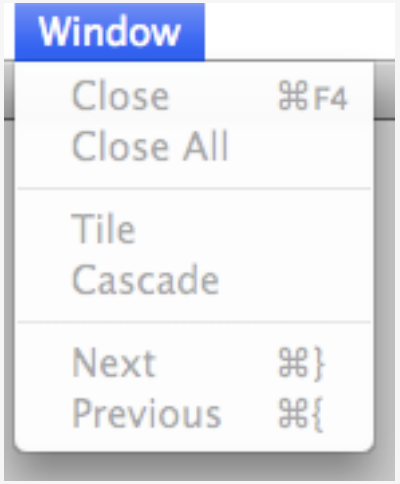
mainwindow.cpp

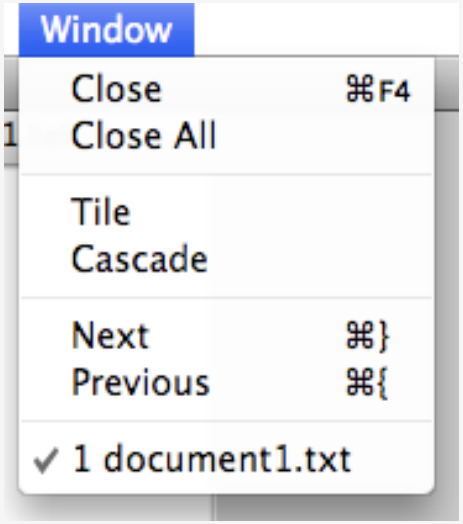
```
MainWindow::MainWindow()
{
    mdiArea = new QMdiArea;
    mdiArea->setHorizontalScrollBarPolicy(Qt::ScrollBarAsNeeded);
    mdiArea->setVerticalScrollBarPolicy(Qt::ScrollBarAsNeeded);
    setCentralWidget(mdiArea);

    createActions();
    createMenus();

    connect(mdiArea, SIGNAL(subWindowActivated(QMdiSubWindow *)),
            this, SLOT(updateMenus()));

    windowMapper = new QSignalMapper(this);
    connect(windowMapper, SIGNAL(mapped(QWidget *)),
            this, SLOT(setActiveSubWindow(QWidget *)));
}
```

MDIs in Qt

```
void MainWindow::updateMenus()
{
    bool hasMdiChild = (activeMdiChild() != 0);
    saveAct->setEnabled(hasMdiChild);
    saveAsAct->setEnabled(hasMdiChild);
    pasteAct->setEnabled(hasMdiChild);
    closeAct->setEnabled(hasMdiChild);
    closeAllAct->setEnabled(hasMdiChild);
    tileAct->setEnabled(hasMdiChild);
    cascadeAct->setEnabled(hasMdiChild);
    nextAct->setEnabled(hasMdiChild);
    previousAct->setEnabled(hasMdiChild);
    separatorAct->setVisible(hasMdiChild);

    bool hasSelection = (activeMdiChild() &&
                         activeMdiChild()->textCursor().hasSelection());
    cutAct->setEnabled(hasSelection);
    copyAct->setEnabled(hasSelection);
}
```

```
MdiChild *MainWindow::activeMdiChild()
{
    if (QMdiSubWindow *activeSubWindow = mdiArea->activeSubWindow())
        return qobject_cast<MdiChild *>(activeSubWindow->widget());
    return 0;
}
```





MDIs in Qt

mainwindow.cpp

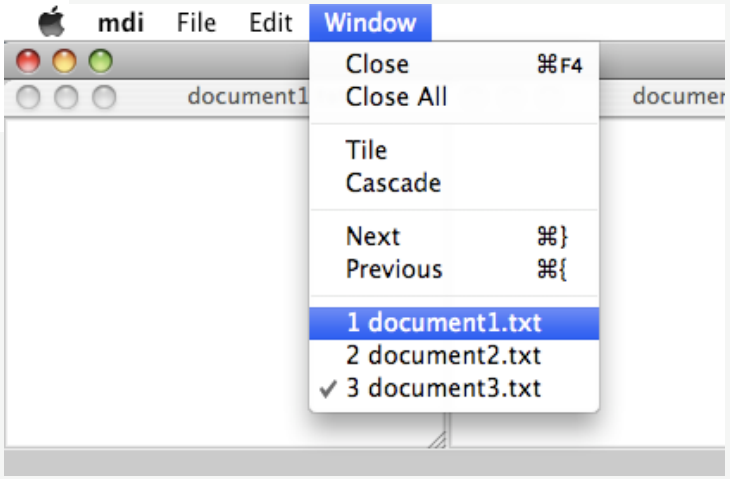
```
void MainWindow::updateWindowMenu()
{
    windowMenu->clear();
    windowMenu->addAction(closeAct);
    windowMenu->addAction(closeAllAct);
    windowMenu->addSeparator();
    windowMenu->addAction(tileAct);
    windowMenu->addAction(cascadeAct);
    windowMenu->addSeparator();
    windowMenu->addAction(nextAct);
    windowMenu->addAction(previousAct);
    windowMenu->addAction(separatorAct);

    QList<QMdiSubWindow *> windows = mdiArea->subWindowList();
    separatorAct->setVisible(!windows.isEmpty());
```

● ● ●

MDIs in Qt

mainwindow.cpp

```
    for (int i = 0; i < windows.size(); ++i) {
        MdiChild *child = qobject_cast<MdiChild *>(windows.at(i)->widget());

        QString text;
        if (i < 9) {
            text = tr("&%1 %2").arg(i + 1)
                               .arg(child->userFriendlyCurrentFile());
        } else {
            text = tr("%1 %2").arg(i + 1)
                              .arg(child->userFriendlyCurrentFile());
        }
        QAction *action  = windowMenu->addAction(text);
        action->setCheckable(true);
        action ->setChecked(child == activeMdiChild());
        connect(action, SIGNAL(triggered()), windowMapper, SLOT(map()));
        windowMapper->setMapping(action, windows.at(i));
    }
}
```

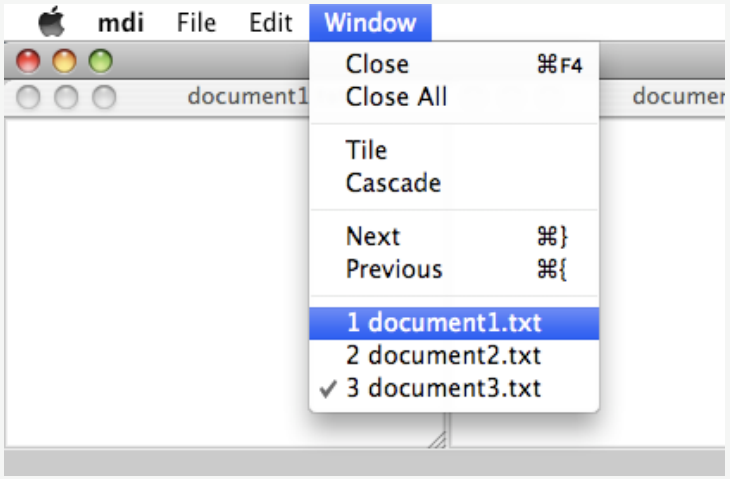



```
QString MdiChild::userFriendlyCurrentFile()
{
    return QFileInfo(fullFileName).fileName();
}
```

mdichild.cpp

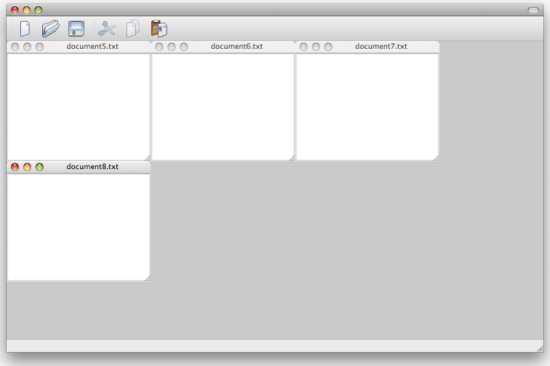
QMdiArea Subwindows

- QMdiArea bietet verschiedene Möglichkeiten an, Subwindows automatisch anordnen zu lassen und zwischen ihnen zu navigieren
- Ein Subwindow lässt sich mit setActiveSubWindow(QMdiSubWindow*) aktivieren (Fokus!)
- closeActiveSubWindow() schliesst das aktive Fenster, closeAllSubWindows() schliesst alle Fenster
- activateNextSubWindow() und activatePreviousSubWindow() gehen zum nächsten bzw. vorherigen Fenster
- subWindowList() gibt eine QList<QMdiSubWindow*> zurück die alle Fenster enthält

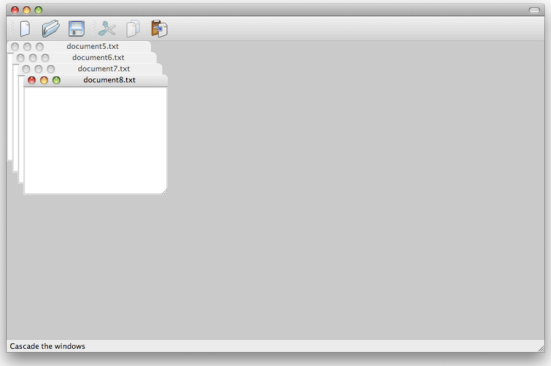


QMdiArea Subwindows Layout

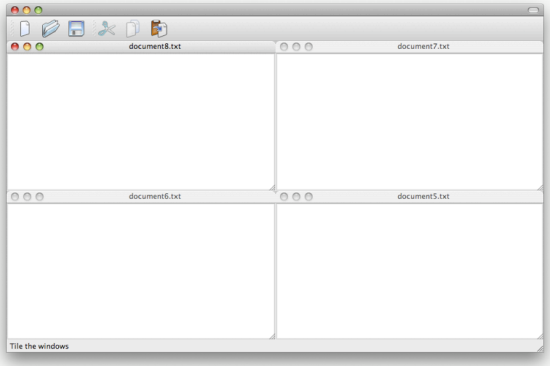
- QMdiArea enthält zwei vorgegebene Layouttypen für die enthaltenen Fenster:
  - Cascade: cascadeSubWindows() ordnet die Fenster kaskadenförmig an
  - Tile: tileSubWindows() versucht den vorhandenen Platz im Hauptfenster möglichst gut auszunutzen



Normal



Cascade

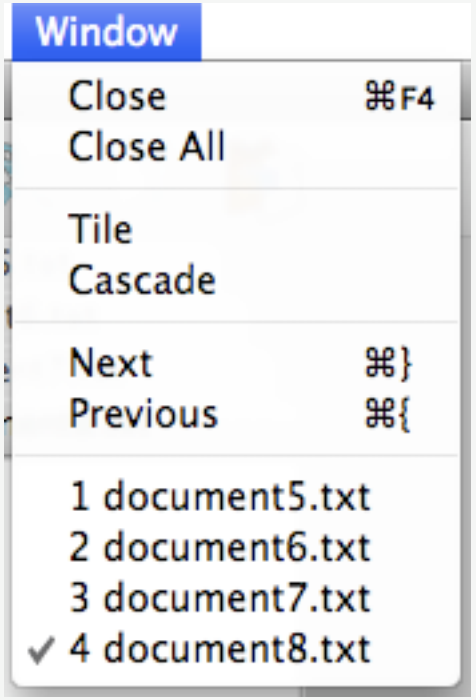


Tile

MDIs in Qt

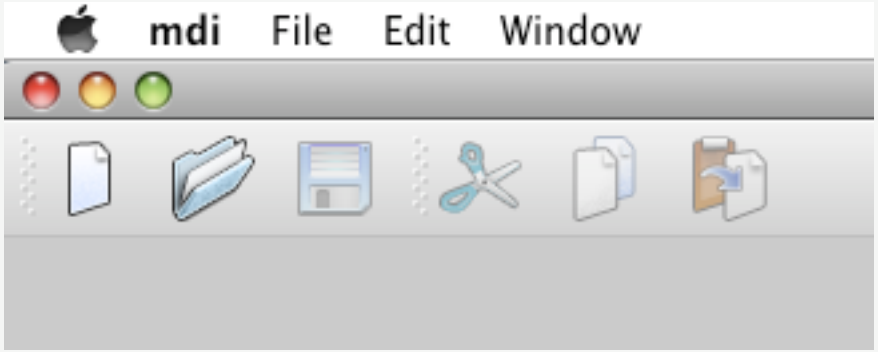
```
closeAllAct = new QAction(tr("Close &All"), this);
closeAllAct->setStatusTip(tr("Close all the windows"));
connect(closeAllAct, SIGNAL(triggered()),
        mdiArea, SLOT(closeAllSubWindows()));

tileAct = new QAction(tr("&Tile"), this);
tileAct->setStatusTip(tr("Tile the windows"));
connect(tileAct, SIGNAL(triggered()), mdiArea, SLOT(tileSubWindows()));

cascadeAct = new QAction(tr("&Cascade"), this);
cascadeAct->setStatusTip(tr("Cascade the windows"));
connect(cascadeAct, SIGNAL(triggered()), mdiArea, SLOT(cascadeSubWindows()));

nextAct = new QAction(tr("Ne&xt"), this);
nextAct->setShortcuts(QKeySequence::NextChild);
nextAct->setStatusTip(tr("Move the focus to the next window"));
connect(nextAct, SIGNAL(triggered()),
        mdiArea, SLOT(activateNextSubWindow()));

previousAct = new QAction(tr("Pre&vious"), this);
previousAct->setShortcuts(QKeySequence::PreviousChild);
previousAct->setStatusTip(tr("Move the focus to the previous "
                             "window"));
connect(previousAct, SIGNAL(triggered()),
        mdiArea, SLOT(activatePreviousSubWindow()));
```

mainwindow.cpp
createActions()



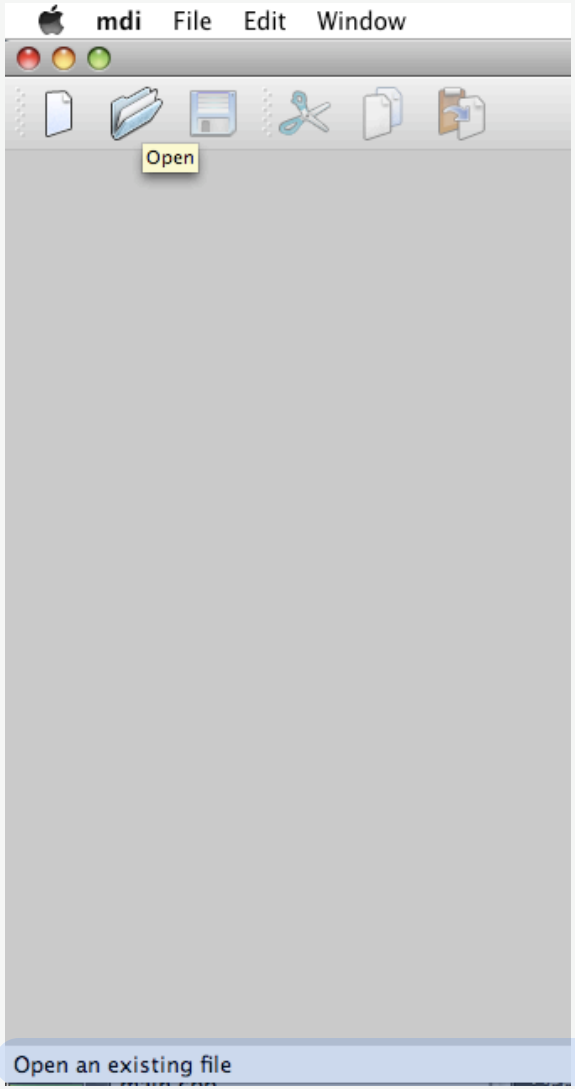
MDIs in Qt

```
void MainWindow::createToolBars()
{
    fileToolBar = addToolBar(tr("File"));
    fileToolBar->addAction(newAct);
    fileToolBar->addAction(openAct);
    fileToolBar->addAction(saveAct);

    editToolBar = addToolBar(tr("Edit"));
    editToolBar->addAction(cutAct);
    editToolBar->addAction(copyAct);
    editToolBar->addAction(pasteAct);
}
```



MDIs in Qt

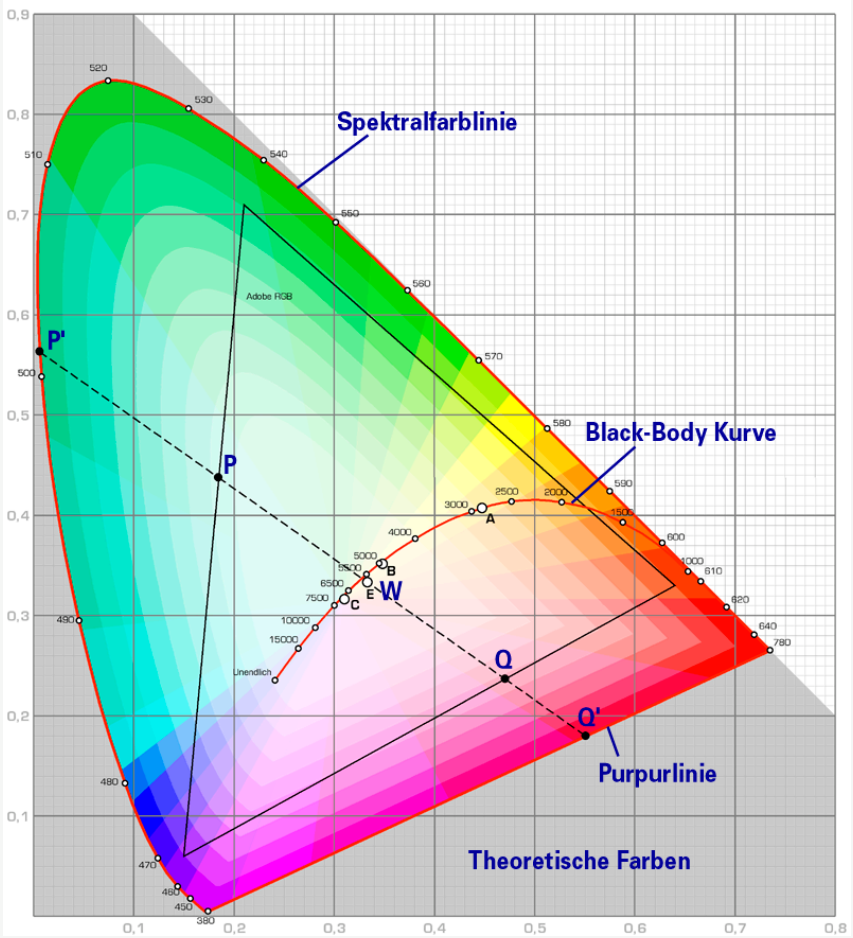
```
void MainWindow::createStatusBar()
{
    statusBar()->showMessage(tr("Ready"));
}
```



Farbmodelle

Farbspezifikation und Farbräume

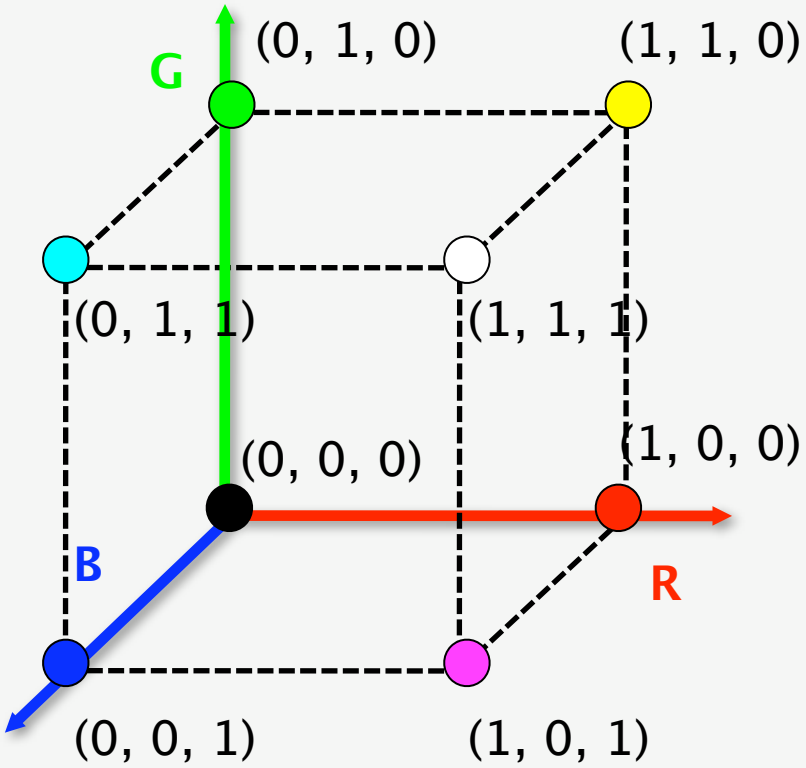
- Farbmodell: Spezifikation eines 3D-Koordinaten-systems und einer Untermenge davon, in der alle sichtbaren Farben eines bestimmten Farbbereiches (Gammut) liegen.
- CIE-Diagramm und Bildschirmgammut



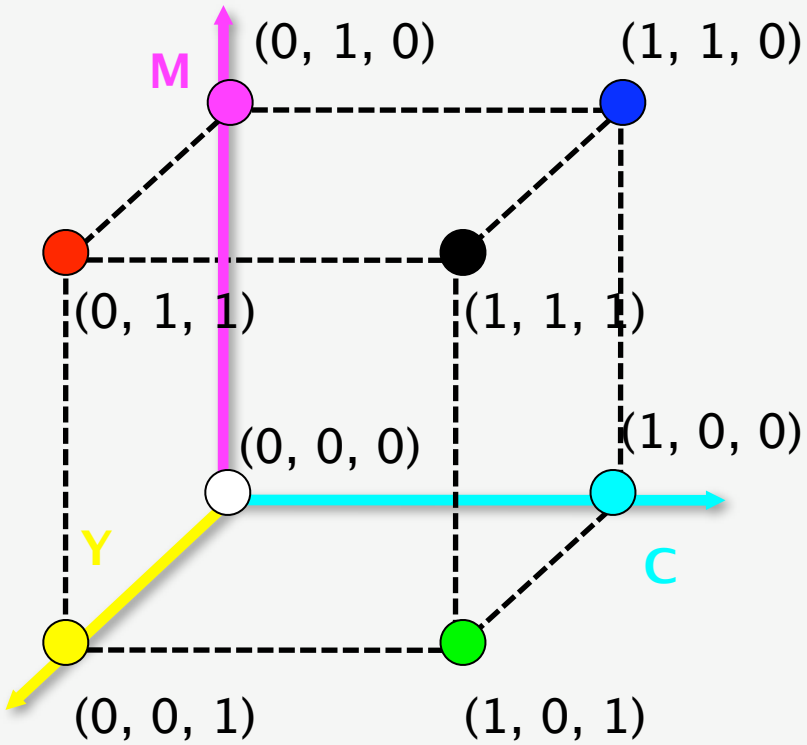
Farbspezifikation und Farbräume

- Hardwareorientierte Farbmodelle:
  - Motiviert durch die Charakteristika von Ausgabegeräten.
  - Beispiele: RGB- und CMY-Modell
- Perzeptionsorientierte Farbmodelle:
  - Gleiche Abstände im Farbraum korrespondieren zu (etwa) gleichen Abständen in der Farbwahrnehmung.
  - Nutzung von physiologischen Größen: Farbton, -sättigung, -helligkeit
  - Beispiele: HLS- und HSV-Modell
- Hardwareorientierte Modelle sind unerlässlich; perzeptionsorientierte für die Farbeingabe wünschenswert. → Transformation notwendig.

RGB-Farbmodell

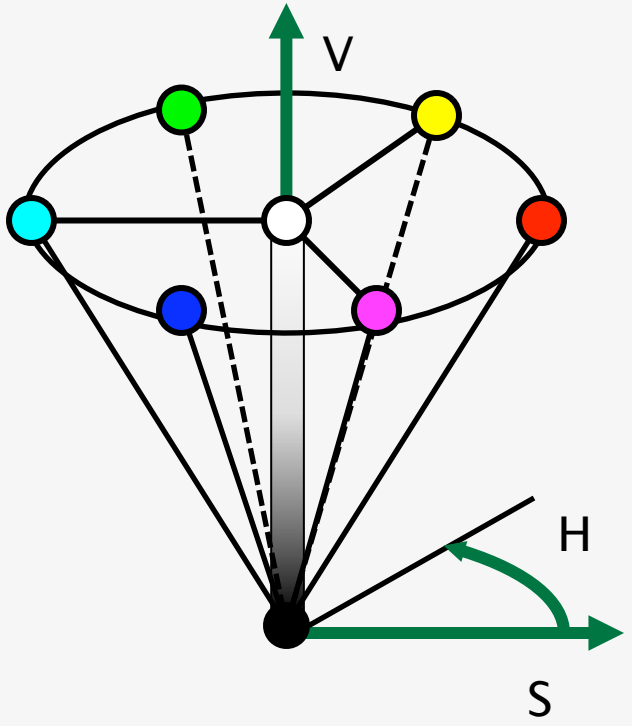


CMY-Farbmodell



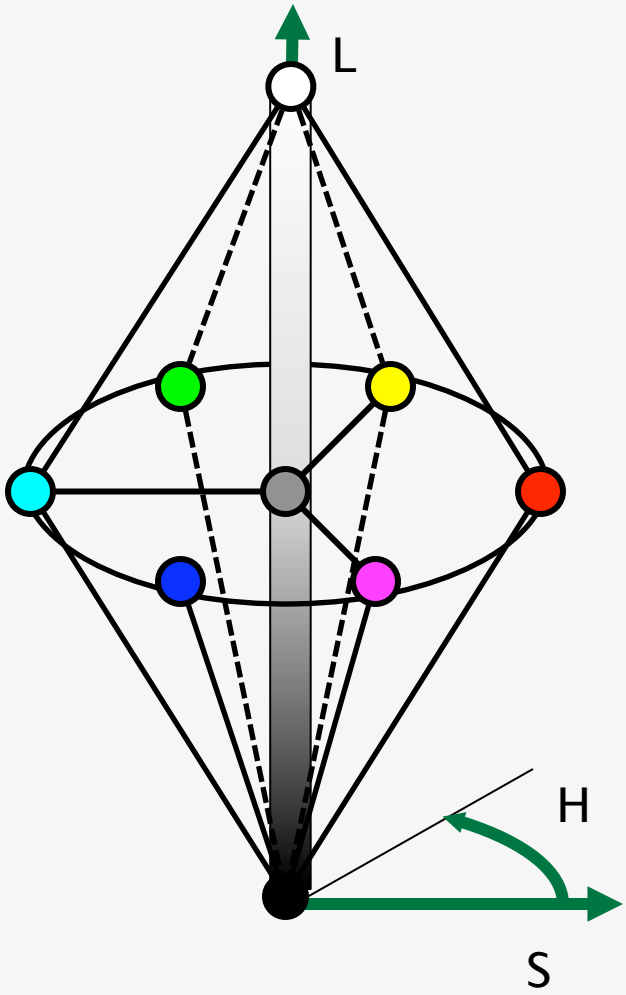
Perzeptionsorientierte Modelle: HSV

- Hue, Saturation, Value
- Hue – Winkel um vertikale Achse, 0° entspricht rot
- Änderung der Sättigung
  - 0 … 1
- Helligkeit entspricht dem Schwarzanteil
- Beispiel: gesättigtes dunkelblau:
  - H = 225°,
  - S = 100 %,
  - V = 100 %.



Perzeptionsorientierte Modelle: HSL

- Hue, Saturation, Lightness
- Strategie von Malern: nimm reines Pigment (H), Weiß dazu (S), Schwarz dazu (1 – L)
- Komponenten nicht unabhängig voneinander
- Graustufen: S = 0
- Voll gesättigte Farben:
- L = 0,5, S = 1



Konvertierung HSL in RGB

$$M_1 = S\sin(H)$$

$$M_2 = S\cos(H)$$

$$I = \frac{L}{\sqrt{3}}$$

$$(R,G,B) = (M_1, M_2, I) \cdot \begin{pmatrix} \frac{2}{\sqrt{6}} & -\frac{1}{\sqrt{6}} & \frac{1}{\sqrt{6}} \\ 0 & \frac{1}{\sqrt{2}} & -\frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{3}} & \frac{1}{\sqrt{3}} & \frac{1}{\sqrt{3}} \end{pmatrix}$$

Konvertierung RGB in HSL

$$(M_1, M_2, I) = (R, G, B) \cdot \begin{pmatrix} \frac{2}{\sqrt{6}} & 0 & \frac{1}{\sqrt{6}} \\ -\frac{1}{\sqrt{6}} & \frac{1}{\sqrt{6}} & \frac{1}{\sqrt{6}} \\ -\frac{1}{\sqrt{6}} & -\frac{1}{\sqrt{6}} & \frac{1}{\sqrt{6}} \end{pmatrix}$$

$$H = \arctan\left(\frac{M_1}{M_2}\right)$$

$$S = \sqrt{M_1^2 + M_1^2}$$

$$L = I\sqrt{3}$$

Weiterführende Literatur:

- Jasmin Blanchette, Mark Summerfield: “C++ GUI Programming with Qt 4”, ISBN-13: 978-0132354165
  Erste Edition kostenlos online: http://www.qtrac.eu/C++-GUI-Programming-with-Qt-4-1st-ed.zip

- http://doc.trolltech.com/4.5/
- http://www.qtsoftware.com/products/