

# 1 Anleitung zur Installation der Qt-Beispiele für Computergraphik bei Prof. Zachmann

Die Anleitung bezieht sich auf die Installation von Qt und der Einbindung der Beispiele unter Visual Studio .NET 2003.

## 2 Voraussetzungen

Visual Studio .NET 2003 und Qt Version (4.3.2) von Trolltech. Es sollte zuerst Visual Studio installiert werden. Für die Installation / Kompilieren von Qt werden ca. 2 GB freier Platz auf C benötigt.

## 3 Qt

### 3.1 Download

Das Qt Framework kann von <http://www.trolltech.com/developer/downloads/qt/windows> als Quelldaten, die benötigt werden, geladen werden. Es muss die reine Quellcode Version verwendet werden **ohne** MinGW. Es empfiehlt sich die Quellen nach `C:\Programme\Qt\«Version»\` zu entpacken<sup>1</sup>

### 3.2 Patch von QtWin

Der inoffizielle Patch des QtWin Projektes darf ab der Qt Version 4.3.2 **nicht** mehr verwendet werden.

### 3.3 System anpassen

Qt benötigt noch verschiedene Umgebungsvariablen<sup>2</sup>, die auch als Benutzervariablen angelegt werden können. Diese müssen entweder neu angelegt oder passend ergänzt werden:

PATH	C:\Programme\Qt\«Version»\bin
INCLUDE	C:\Programme\Qt\«Version»\include
LIB	C:\Programme\Qt\«Version»\lib
QMAKESPEC	win32-msvc.net <sup>3</sup> <i>oder</i> win32-msvc2005 <sup>4</sup>
QTDIR	C:\Programme\Qt\«Version»

Um die korrekten Angaben zu prüfen, kann man mit Hilfe des *Visual Studio Command Prompt* und dem Befehl `echo %LIB%` (analog die anderen Variablen) die Angaben zu den Libraries sich ausgeben lassen. Sollten hier Probleme auftreten, wird auch das Kompilieren im nächsten Schritt fehlschlagen.

### 3.4 Kompilieren und installieren

Im *Visual Studio Command Prompt* wechselt man nach `C:\Programme\Qt\«Version»\`.

- Dort führt man den Befehl `configure` aus. Dieser Vorgang kann je nach Rechner schon einige Zeit in Anspruch nehmen und muss ohne Fehler durchlaufen werden.
- Nach dem Durchlauf kompiliert man nun Qt mit dem Befehl `nmake`. Dies kann ebenfalls einige Stunden dauern.
- Um nun erstellte temporäre Dateien zu entfernen, verschiebt man das Verzeichnis `C:\Programme\Qt\«Version»\lib` aus dem Qt-Ordner, ruft dann im *Command Prompt* den Befehl `nmake clean` auf und verschiebt das `lib`-Verzeichnis wieder zurück.

<sup>1</sup>Der Qt Pfad darf keine Leerzeichen enthalten, da dies zu Problemen führen kann

<sup>2</sup>Die Umgebungsvariablen werden unten den Eigenschaften des Arbeitsplatzes, Reiterkarte *Erweitert* und dem Button *Umgebungsvariablen* gesetzt

<sup>3</sup>für Visual Studio .NET 2003

<sup>4</sup>für Visual Studio .NET 2005

Danach ist Qt für das System passend kompiliert und installiert

### 3.5 Erzeugen der Qt und Visual Studio Projektdateien

Man erstellt ein Verzeichnis mit allen notwendigen Dateien (\*.cpp, \*.h, \*.ui o.ä.). Man wechselt im *Visual Studio Command Prompt* in dieses Verzeichnis und führt dort folgende zwei Befehle aus:

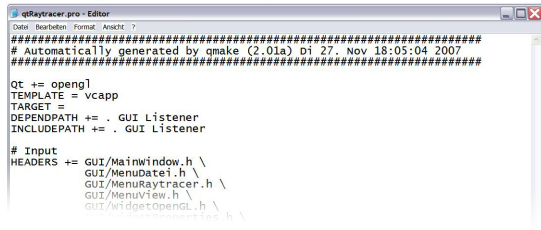


```
Visual Studio .NET 2003 Command Prompt
D:\qtRaytracer>qmake -project -t vcapp "Qt += opengl"
D:\qtRaytracer>qmake -tp vc
D:\qtRaytracer>_
```

**qmake -project -t vcapp „Qt += opengl“** hiermit legt man die Qt spezifischen Projektdateien an

**qmake -tp vc** erzeugt die Visual Studio spezifischen Projektdateien

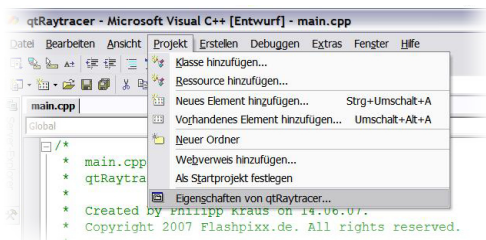
Enthält das Quellcodeverzeichnis Unterverzeichnisse, öffnet man die pro-Datei mit einem Texteditor und trägt unter *INCLUDEPATH* die Unterverzeichnisse ein



```
qtRaytracer.pro - Editor
#####
# Automatically generated by qmake (2.01a) Di 27. Nov 18:05:04 2007
#####
Qt += opengl
TEMPLATE = vcapp
TARGET =
DEPENDPATH += . GUI Listener
INCLUDEPATH += . GUI Listener

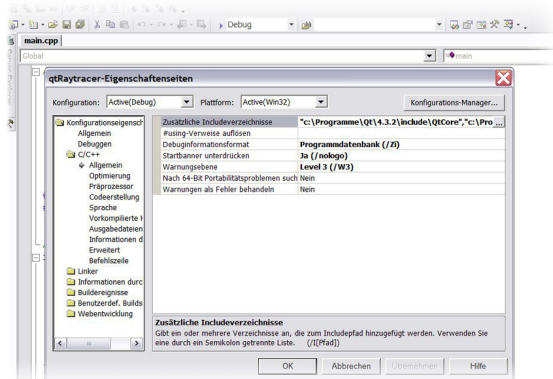
# Input
HEADERS += GUI/Mainwindow.h \
           GUI/MenuDatei.h \
           GUI/MenuRaytracer.h \
           GUI/MenuView.h \
           GUI/WidgetOpenGL.h \
```

### 3.6 Projekt kompilieren und linken

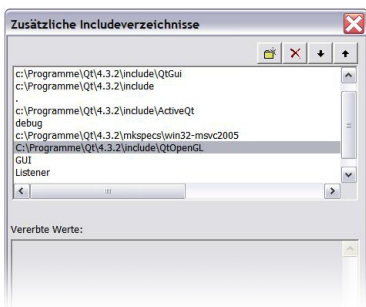


Nachdem man nun die Visual Studio Datei geöffnet hat, kann man die Quellcodes editieren. Die Ausführung von *qmake* importiert in das Visual Studio Projekt nur die Standard Headerdateien und Libraries. Für OpenGL-Projekte müssen noch verschiedene Pfade und Libraries ergänzt werden.

#### 3.6.1 Einbinden Headerdateien in das Projekt



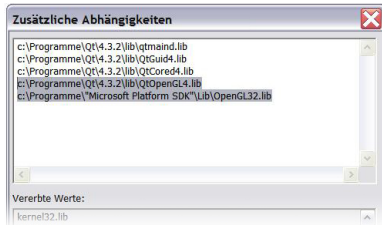
Für die Bearbeitung von OpenGL Programmen müssen zusätzlich die QtOpenGL-Headerdateien mit übernommen werden, da sonst Kompilierfehlermeldungen entstehen. Man öffnet dazu die Projekteigenschaften und trägt die fehlenden Include-Verzeichnisse nach, im QtOpenGL-Fall: `C:\Programme\Qt\«Version»\include\QtOpenGL`. Man kann über die *Konfiguration* für alle oder einzeln für den *Debug* und *Release* die Include-Pfade setzen. Das Beispiel bezieht sich nur auf den *Debug* Stand, ist aber für die Anwendung völlig ausreichend.



Über den erweiterten Dialog kann man die Verzeichnisse auswählen oder manuell eingeben. Natürlich müssen alle Include-Verzeichnisse angegeben werden. Es gehören auch die Include-Verzeichnisse dazu, die vom Projekt selbst verwendet werden, d.h. es müssen hier auch die Verzeichnisse angegeben werden, die man zuvor in die Qt-Projektdatei eingegeben hat. Absolute oder relative Pfadangaben können verwendet werden .

Damit können nun die Quellcodes kompiliert werden.

### 3.6.2 Einbinden der Libraries in das Projekt



Damit nun keine Linkerfehler auftreten muss das Projekt noch gegen die passenden Windowslibraries gelinkt werden. Für die OpenGL Anwendung einmal gegen die die QtOpenGL Library und nachfolgend gegen die windowseigene OpenGL32 Library. Im Dialog der Projekteigenschaft wird dazu dann unter dem Menüpunkt *Linker*, dort der Menüpunkt *Eingabe* und das Feld *Zusätzliche Abhängigkeiten* die passenden Dateien hinzugefügt. Für OpenGL sind folgende Dateien zu wählen:

- C:\Programme\Qt\<<Version>>\lib\QtOpenGLd4.lib
- C:\Programme\“Microsoft Platform SDK“\Lib\OpenGL32.lib

Bei den Qt-Libraries ist auf das *d* im Dateinamen zu achten, entweder müssen alle Dateinamen ohne *d* oder mit *d* angegeben werden. Unterschiedliche Angaben führen dazu, dass zwar ein Executable erzeugt wird, dieses aber nicht startet. Der *qmake* Prozess fügt automatisch Dateien mit *d* hinzu. Natürlich können auch hier wieder die verschiedenen *Konfigurationen* verwendet werden. Danach ist die Konfiguration abgeschlossen und man kann das Projekt nun vollständig kompilieren.