



# Software-Entwicklung mit *OpenGL* und *C++*

Manfred Brill

Oktober 2005

# Inhaltsverzeichnis

<b>1</b>	<b>OpenGL-Entwicklung mit <i>Eclipse</i></b>	<b>1</b>
1.1	OpenGL-Komponenten installieren für Eclipse . . . . .	1
1.2	Projekt anlegen . . . . .	1
1.3	<i>Make Targets</i> . . . . .	3
<b>2</b>	<b>OpenGL-Entwicklung mit Microsoft Visual Version 6</b>	<b>6</b>
2.1	OpenGL-Komponenten installieren . . . . .	6
2.2	Projekte anlegen . . . . .	7
2.3	Projekteinstellungen vornehmen . . . . .	7
2.4	Quellcodedateien ins Projekt aufnehmen . . . . .	8
<b>3</b>	<b>OpenGL-Entwicklung mit Editor und Compiler</b>	<b>9</b>

### **Zusammenfassung**

Dieser Text ist eine Einführung in die Software-Entwicklung mit *OpenGL* und *C++*. Vorgestellt werden die Entwicklungsumgebungen *Eclipse*, *Microsoft Visual C++ Version 6* und die Entwicklung mit Hilfe eines Editors und Makefiles.

# Kapitel 1

## OpenGL-Entwicklung mit *Eclipse*

### 1.1 OpenGL-Komponenten installieren für Eclipse

Um mit *Eclipse* und *OpenGL* entwickeln zu können, müssen bestimmte Komponenten installiert sein. Sie benötigen das Plug-in *CDT* (C/C++ Development Tools), und eine Installation von *Cygwin* oder *MinGW32*.

**Tip:**

*Eclipse* ist nur eine oberfläche für die Software-Entwicklung und enthält keinen eigenen C++-Compiler. Diesen installieren Sie durch *Cygwin* oder *MinGW32* – Sie verwenden den *GNU*-Compiler.

Folgende Komponenten müssen bei der Installation von *Cygwin* ausgewählt werden:

- Unter der Kategorie *Devel*: *gcc*, *gdb*, *make*
- Unter der Kategorie *Libs*: *w32api* und *OpenGL*.
- Die *dll*-Datei *cygwin1.dll* aus dem *bin*-Verzeichnis des Installationsverzeichnis von *Cygwin* müssen Sie *manuell* in das Systemverzeichnis des Betriebssystems kopieren, also z.B.: `C:\WINDOWS\system32`.

**Tip:**

Achten Sie beim Download darauf, dass die ausgewählte *CDT*-Version zu der installierten *Eclipse*-Version kompatibel ist!

### 1.2 Projekt anlegen

*Eclipse* arbeitet mit Projekten. Um ein neues Projekt anzulegen gehen Sie wie folgt vor:

1. Erzeugen Sie einen Ordner mit dem Namen des Projekts.
2. Starten Sie *Eclipse*. In Abbildung 1.1 sehen Sie das User Interface von *Eclipse*.
3. Klicken Sie auf *File* → *New* → *Project...*
4. Unter *C++* wählen Sie *Standard Make C++ Project* aus. Abbildung 1.2 zeigt diesen Dialog.

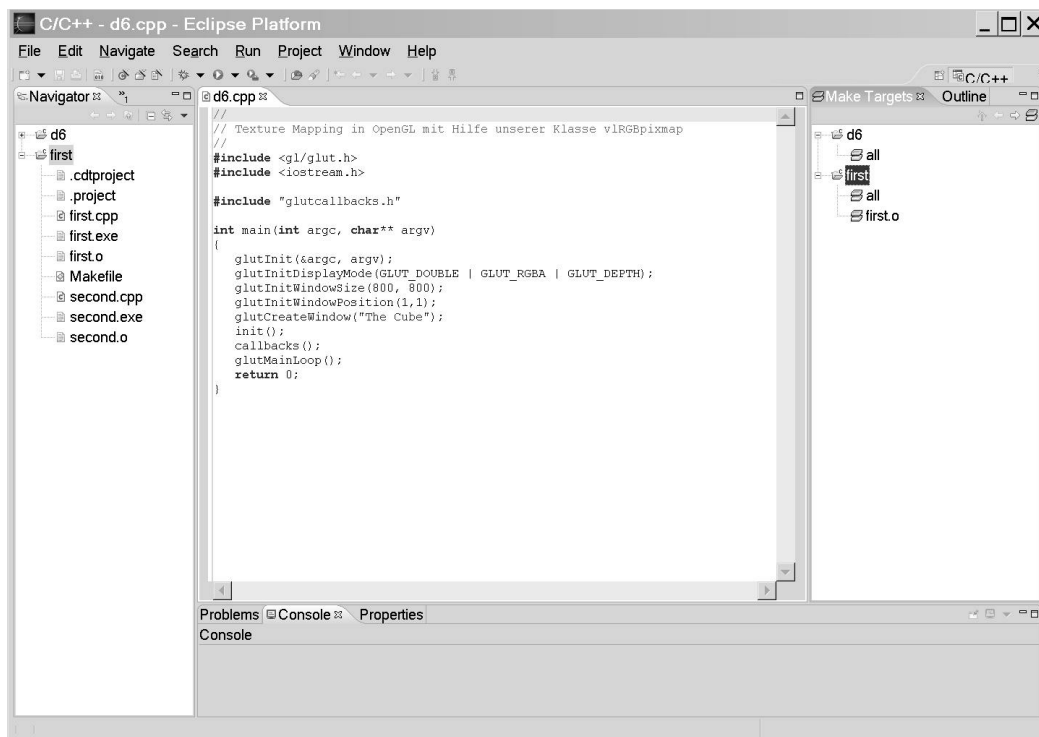


Abbildung 1.1: Das User Interface von Eclipse

5. Geben Sie den Projektnamen an. Deaktivieren Sie *Use default Location* und geben Sie den Pfad zu ihrem Projektordner an, wie in Abbildung 1.3 zu sehen. *Alle* bereits im Ordner enthaltenen Dateien werden in das Projekt aufgenommen.
6. Bestätigen Sie anschließend mit *Finish*, das Projekt wird erzeugt und erscheint im *Navigator-Fenster* von *Eclipse*.

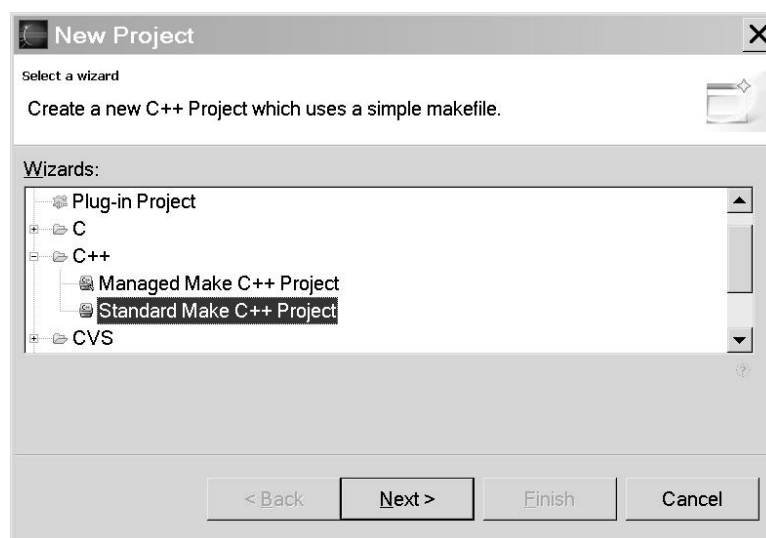


Abbildung 1.2: Anlegen eines neuen C++-Projekts in Eclipse

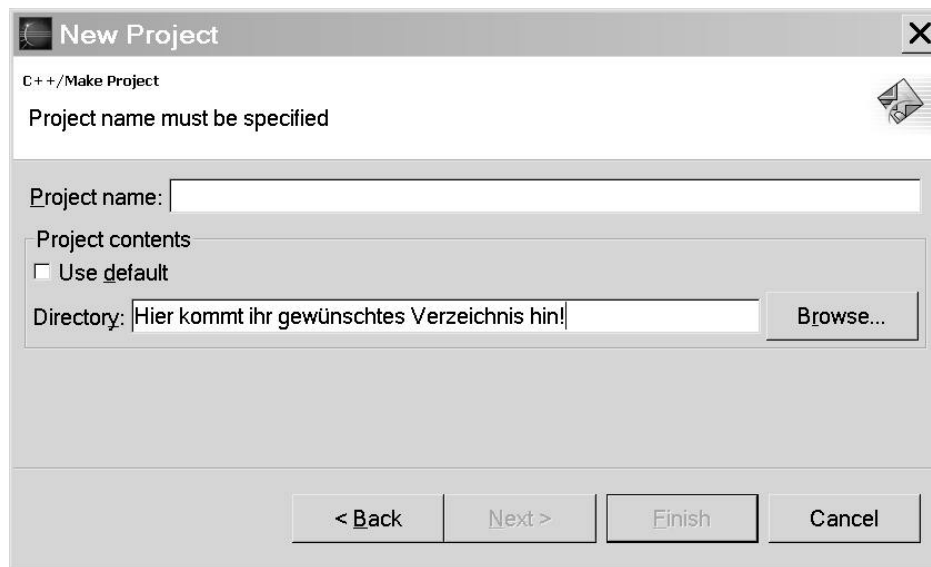


Abbildung 1.3: Angabe des Verzeichnisses und des Projektnamens in Eclipse

## Quelldateien ins Projekt aufnehmen

Sie werden in der Übung teilweise mit bestehenden Quelldateien arbeiten. Um eine Quelldatei in ein neues Projekt zu übernehmen gehen Sie wie folgt vor:

1. Wählen Sie im *Navigator*-Fenster von *Eclipse* das gewünschte Projekt aus.
2. Klicken Sie auf *File* → *Import*
3. Wählen Sie *File System* aus und klicken Sie auf *Next*.
4. Geben Sie die Dateien an, die importiert werden sollen und klicken Sie auf *Finish*.

## 1.3 Make Targets

Neben den Quellcodes benötigen Sie für ein Eclipse-Projekt einen Makefile. Achten Sie darauf, dass wie in Abbildung 1.1 ein Fenster mit dem Namen *Make Targets* geöffnet ist. Ist dies nicht der Fall, können sie dies mit *Windows* → *Show View* → *Make Targets* einblenden.

Klicken Sie mit der rechten Maustaste in diesem Fenster auf den Projektnamen und wählen *Add Make Target* aus. Im ersten Schritt sollten Sie im dann auftretenden Fenster wie in Abbildung 1.4 im Eingabefeld *Target Name* den Namen des *Make Targets* eintragen. Im Eingabefeld *Make Target* müssen Sie den Begriff des Targets eintragen, so wie er im Makefile auftaucht.

In einem ersten Schritt tragen Sie das Target *all* ein. Ist dies geschehen, können Sie das komplette Projekt durch einen Doppelklick auf *all* aktualisieren. Ein weiteres Target, das in einem Makefile, der für Eclipse verwendet werden soll auf jeden Fall auftreten *muss*, ist *clean*. Dieses Target dient dazu, alle Zwischendateien im Projekt zu löschen.

### Tip:

Ein Blick in den Makefile lohnt sich immer. Insbesondere sollten Sie die Make Targets für die einzelnen Object-Dateien eintragen. Haben Sie nur eine Datei verändert, sollten Sie in einem ersten Schritt *immer* überprüfen, ob diese Datei auch übersetzt wird!

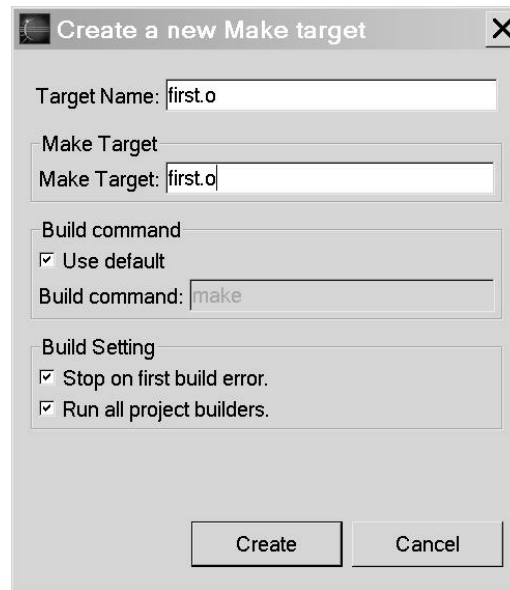


Abbildung 1.4: Make Targets in Eclipse

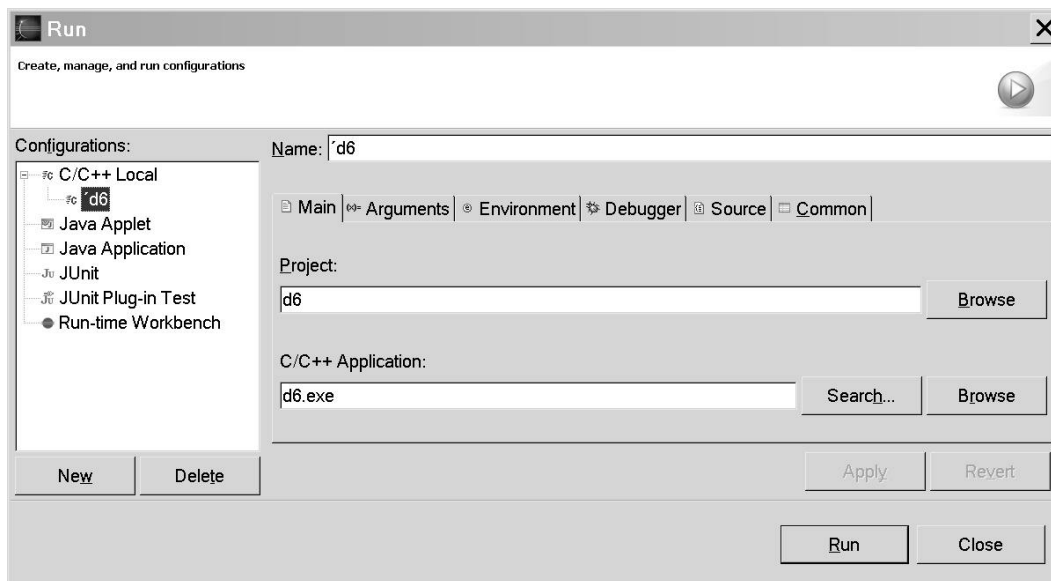


Abbildung 1.5: Eine Run-Umgebung in Eclipse

## Ausführen von Programmen

Benötigt Ihr Programm keine Eingabedateien, dann können sie die ausführbare Datei mit einem Doppelklick im *Navigator* starten.

Es kann vorkommen, dass Ihr Programm externe Dateien öffnen und bearbeiten soll. Bei solchen Programmen reicht es nicht aus, im Navigator die ausführbare Datei mit einem Doppelklick zu starten. Sie müssen eine *Run-Anweisung* erzeugen, um das Programm in *Eclipse* ausführen zu können. Wählen Sie dazu *Run* → *Run...* aus. Es erscheint das Fenster wie in Abbildung 1.5.

Wählen Sie *New* aus, tragen Sie den Namen der neuen Anweisung, des Projekts und das auszuführende Programm ein. Anschließend müssen Sie den Reiter *Arguments* auswählen. Abbildung 1.6 zeigt das entsprechende Fenster. Aktivieren Sie die Option *Local Directory* und achten

Sie darauf, dass die Dateien, die gelesen oder geschrieben werden sollen, im gleichen Verzeichnis wie ihr ausführbares Programm liegen. Ist dies nicht der Fall, können Sie ein entsprechendes Verzeichnis eintragen.

Sind alle Einstellungen gemacht, erkennen Sie das daran, dass die Befehle *Apply* und *Run* auswählbar sind. Testen Sie Ihre Run-Anweisung durch *Run*.

**Tip:**

Die hier dargestellten Einstellungen und Bedienhinweise gelten auch für die *Linux*-Version von *Eclipse*!

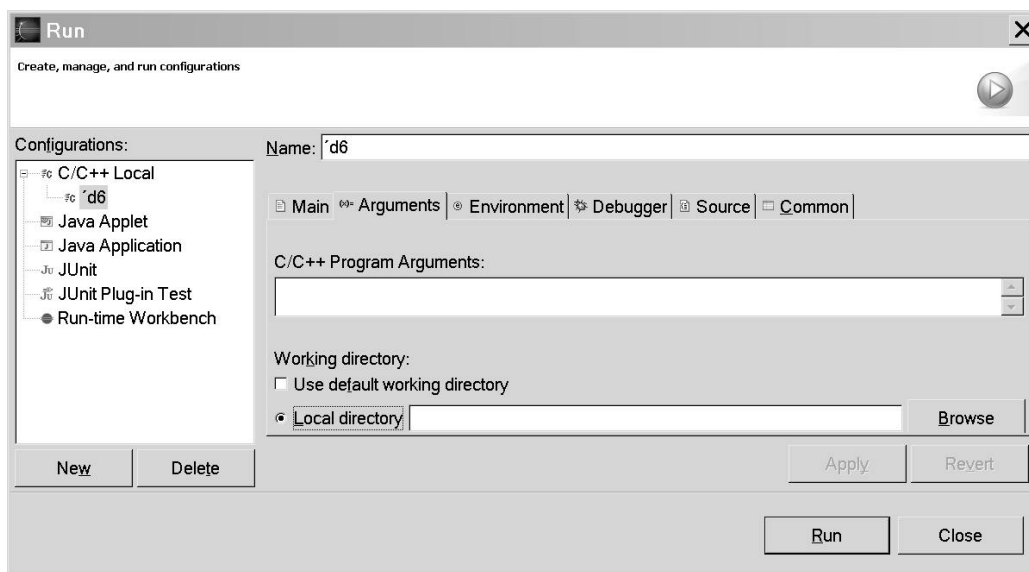


Abbildung 1.6: Das Arbeitsverzeichnis in Eclipse

## Kapitel 2

# OpenGL-Entwicklung mit Microsoft Visual Version 6

### 2.1 OpenGL-Komponenten installieren

Um mit *Microsoft Visual C++* und *OpenGL* zu entwickeln, müssen bestimmte Komponenten installiert sein. Alle hier genannten Komponenten sind im VisLab bereits installiert. Sie benötigen folgende Komponenten:

- Zu *OpenGL* gehören folgenden Komponenten, die in der Regel standardmäßig von *Microsoft Visual C++* installiert werden:
  - Den Header Dateien `gl.h` und `glu.h`, die in einem Ordner namens `gl` im include-Pfad des Compilers stehen muss, also z.B.:  
`C:\Programme\Microsoft Visual C++\VC98\Include\GL.`
  - Die Bibliotheken `opengl32.lib` und `glu32.lib`, die im lib-Pfad des Compilers stehen müssen, also z.B.:  
`C:\Programme\Microsoft Visual C++\VC98\Lib.`

Sollten Ihnen diese Dateien fehlen, erhalten Sie diese unter:

<http://www.microsoft.com>

<http://www.opengl.org>

<http://www.vislab.de/cgbuch/>

Haben Sie eine moderne *OpenGL*-Karte, dann empfiehlt es sich, das entsprechende *SDK* von der Website des Herstellers zu beziehen; auch dort finden Sie entsprechende Header- und Bibliotheksdateien.

- Wenn Sie *GLUT* nutzen wollen, benötigen Sie folgenden Dateien:
  - Die Header Datei `glut.h`, die im Ordner `gl` im include-Pfad des Compilers stehen muss, also z.B.:  
`C:\Programme\Microsoft Visual C++\VC98\Include\GL.`
  - Die Bibliothek `glut32.lib`, die im lib-Pfad des Compilers stehen müssen, also z.B.:  
`C:\Programme\Microsoft Visual C++\VC98\Lib.`
  - Die *dll*-Datei `glut32.dll`, die im Systemverzeichnis des Betriebssystems stehen muss, also z.B.:  
`C:\WINNT\system32.`

Sollten Ihnen diese Dateien fehlen, erhalten Sie diese unter:

<http://www.opengl.org>

<http://www.vislab.de/cgbuch/>

- Wollen Sie auch noch die Funktionen der *GLUI* nutzen, benötigen Sie neben den Komponenten des *GLUT* noch folgendes:

- Die *Header*-Datei `glui.h`, die im Ordner `gl` im `include`-Pfad des Compilers stehen muss, also z.B.:

`C:\Programme\Microsoft Visual C++\VC98\Include\GL.`

- Die Bibliothek `glui32.lib`, die im `lib`-Pfad des Compilers stehen muss, also z.B.:

`C:\Programme\Microsoft Visual C++\VC98\Lib.`

## 2.2 Projekte anlegen

*Visual C++* arbeitet mit Projekten. Zu einem Projekt gehört unter anderem ein Ordner, der bestimmte Dateien und Unterverzeichnisordner des Projekts enthält. Um ein neues Projekt für diese Übung anzulegen gehen Sie wie folgt vor:

1. Starten Sie *Visual C++*. Erscheint das Dialogfenster *Select Source Code Control System* der Software *alienbrain* klicken Sie auf *Cancel* um das Fenster zu schliessen (dies gilt nur für die Rechner im VisLab).
2. Klicken Sie auf *Datei* → *Neu...*
3. Wählen Sie *Win32 Konsolenanwendung*, geben Sie einen Projektnamen und einen Pfad an.
4. Bestätigen Sie anschließend mit *OK* und wählen Sie im folgenden Dialogfenster *Ein leeres Projekt* aus.

## 2.3 Projekteinstellungen vornehmen

Um die Quellcodedateien der Übungsaufgaben fehlerfrei übersetzen zu können müssen am Projekt Änderungen vorgenommen werden (die Reihenfolge ist wichtig!):

1. Klicken Sie auf *Erstellen* → *Aktive Konfiguration festlegen...* und wählen Sie *Alle Konfigurationen* aus.
2. Klicken Sie auf *Projekt* → *Einstellungen...*
3. Wechseln sie zum Tab *C/C++*.
4. Stellen Sie unter *Warnstufe*: *Stufe 1* ein.
5. Wechseln Sie im Pulldown-Menü *Kategorie*: von *Allgemein* auf *Vorkompilierte Header* und stellen Sie dort *Vorkompilierte Header nicht verwenden* ein.
6. Wechseln sie zum Tab *Linker*.
7. Ergänzen Sie die *Objekt- Bibliothek-Module* um folgende Zeile:

`glut32.lib glu32.lib Opengl32.lib`

8. Schliessen Sie das Fenster *Projekteinstellungen* durch *OK*.

## 2.4 Quellcodedateien ins Projekt aufnehmen

Sie werden im Praktikum teilweise mit bestehenden Quellcodedateien arbeiten. Um eine Quellcodedatei in ein neues Projekt zu übernehmen gehen Sie wie folgt vor:

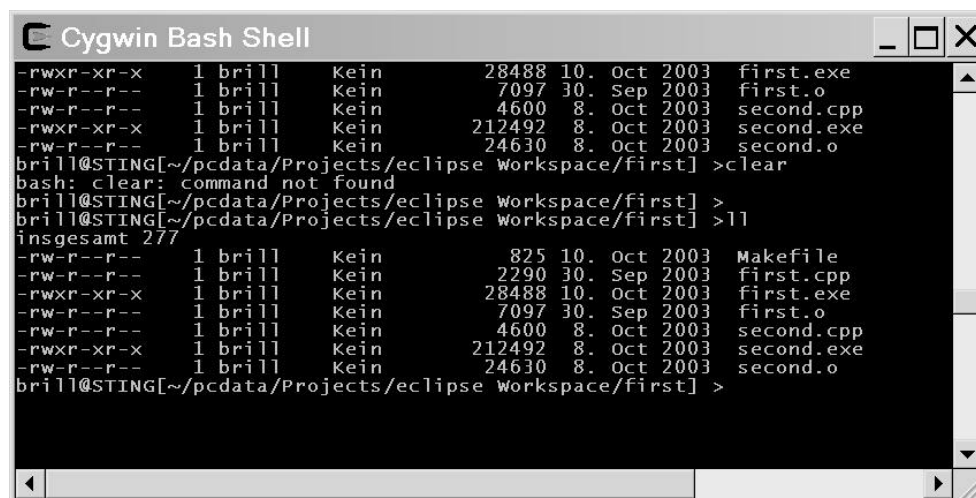
1. Durch das anlegen eines neuen Projekts wird auch ein Ordner mit dem Namen des Projekts angelegt. Kopieren Sie die gewünschte Quellcodedatei (Dateiendung `.cpp`) in diesen Ordner.
2. Im *Visual C++* sehen Sie am linken Rand den sog. *Arbeitsbereich*. Dieser ist standardmäßig auf den Tab *Klassen* gestellt. Wechseln Sie hier auf *Dateien*.
3. Die Ordnerstruktur des Projekts wird dargestellt. Öffnen Sie das Ordner Projektordner. Es enthält unter anderem das Ordner *Quellcodedateien*.
4. Klicken Sie mit der rechten Maustaste auf das Ordner *Quellcodedateien* und wählen Sie *Dateien zu Ordner hinzufügen*.
5. Wählen Sie die gewünschte *Quellcodedatei* aus. Die Datei sollte in der Ordnerstruktur des Projekts erscheinen. Mit einem Doppelklick öffnet sich die Datei im Editorfenster von *Visual C++*.
6. Die Datei ist nun Teil des Projekts und kann dort editiert und übersetzt werden.

## Kapitel 3

# OpenGL-Entwicklung mit Editor und Compiler

Natürlich können Sie auch ohne eine Entwicklungsumgebung arbeiten. Dazu benötigen Sie eine *Shell* wie die *bash*-Shell in *Cygin*, das *make*-Kommando und einen Editor.

Die Shell starten Sie mit Hilfe des Startmenüs und dem Befehl *Cygin*. Danach sehen Sie ein Fenster ähnlich dem in Abbildung 3.1.

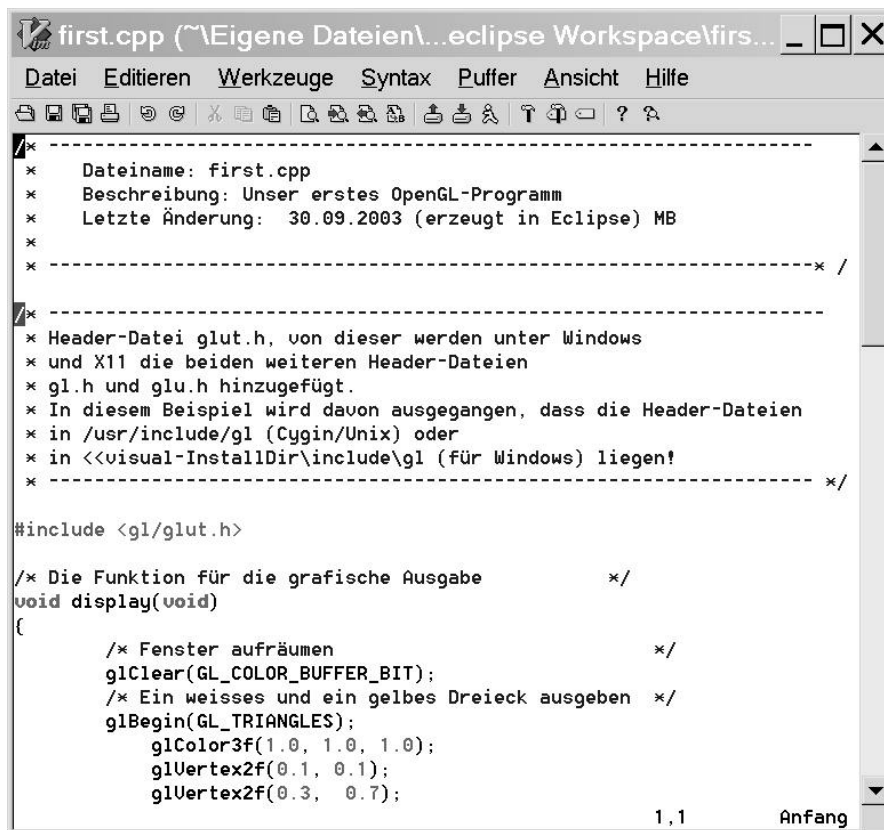


```
Cywin Bash Shell
-rwxr-xr-x  1 brill  Kein      28488 10. Oct 2003  first.exe
-rw-r--r--  1 brill  Kein      7097 30. Sep 2003  first.o
-rw-r--r--  1 brill  Kein      4600  8. Oct 2003  second.cpp
-rwxr-xr-x  1 brill  Kein     212492 8. Oct 2003  second.exe
-rw-r--r--  1 brill  Kein      24630  8. Oct 2003  second.o
brill@STING[~/pcdata/Projects/eclipse Workspace/first] >clear
bash: clear: command not found
brill@STING[~/pcdata/Projects/eclipse Workspace/first] >
brill@STING[~/pcdata/Projects/eclipse Workspace/first] >ll
insgesamt 277
-rw-r--r--  1 brill  Kein      825 10. Oct 2003  Makefile
-rw-r--r--  1 brill  Kein      2290 30. Sep 2003  first.cpp
-rwxr-xr-x  1 brill  Kein      28488 10. Oct 2003  first.exe
-rw-r--r--  1 brill  Kein      7097 30. Sep 2003  first.o
-rw-r--r--  1 brill  Kein      4600  8. Oct 2003  second.cpp
-rwxr-xr-x  1 brill  Kein     212492 8. Oct 2003  second.exe
-rw-r--r--  1 brill  Kein      24630  8. Oct 2003  second.o
brill@STING[~/pcdata/Projects/eclipse Workspace/first] >
```

Abbildung 3.1: Die Bash-Shell

Für das Erstellen des Quellcodes können Sie jeden *ASCII*-Editor verwenden. Im VisLab sind neben dem *notepad* (von dem wir Ihnen aber dringend abraten) auch *proton* und die Windows-Version *gvim* des *vi* installiert. Beide Editoren besitzen *Syntax-Highlighting*, was Ihnen die Arbeit deutlich erleichtert. In Abbildung 3.2 sehen Sie einen C++-Quellcode in *gvim*.

Wollen Sie eine Datei übersetzen, oder sogar das ganze Projekt neu erstellen, dann können Sie *make* durch eine Eingabe in der Shell starten. Das *Make-Target all* starten Sie mit der Eingabe *make all*. Existiert ein *Make Target* wie *first.o*, dann geben Sie *make first.o* ein. Ein solches Target gehört im Allgemeinen zu einer Datei *first.cpp* und dient dazu, diese Datei zu compilieren. Fehlermeldungen und Ausgaben von *make* oder den aufgerufenen Programmen erscheinen in der Konsole.



```

first.cpp (~\Eigene Dateien\...eclipse Workspace\first...
Datei Editieren Werkzeuge Syntax Puffer Ansicht Hilfe
-----
/*
 *   Dateiname: first.cpp
 *   Beschreibung: Unser erstes OpenGL-Programm
 *   Letzte Änderung: 30.09.2003 (erzeugt in Eclipse) MB
 *
 * -----*/
/*
 * Header-Datei glut.h, von dieser werden unter Windows
 * und X11 die beiden weiteren Header-Dateien
 * gl.h und glu.h hinzugefügt.
 * In diesem Beispiel wird davon ausgegangen, dass die Header-Dateien
 * in /usr/include/gl (Cygwin/Unix) oder
 * in <<visual-InstallDir\include\gl (für Windows) liegen!
 * ----- */
#include <gl/glut.h>

/* Die Funktion für die grafische Ausgabe */
void display(void)
{
    /* Fenster aufräumen */
    glClear(GL_COLOR_BUFFER_BIT);
    /* Ein weisses und ein gelbes Dreieck ausgeben */
    glBegin(GL_TRIANGLES);
        glColor3f(1.0, 1.0, 1.0);
        glVertex2f(0.1, 0.1);
        glVertex2f(0.3, 0.7);
}
1,1 Anfang

```

Abbildung 3.2: C++-Quellcode im *vim***Tip:**

Achten Sie darauf, alle im Editor geöffneten Dateien abzuspeichern, bevor Sie *make* anstossen!

War *make all* erfolgreich, können Sie das Programm ausführen. Dazu geben Sie einfach den Namen des Programms in der Konsole ein. Ist dieser Name *first.exe*, dann geben Sie *first* ein! Das Programm wird ausgeführt. Möglich ist auch ein Doppelklick im *File Explorer* von *MS Windows*.

**Tip:**

Das Kommando für eine Auslistung aller Dateien im aktuellen Ordner lautet *ls* oder *ls -l*. Mit der Option *-l* erhalten Sie neben den Namen auch Angaben über Zugriffsrechte und Änderungsdaten. Ausführbare Programme erkennen Sie an der Angabe *x* wie „execute“.

Die Arbeit mit der Konsole ist zu Beginn sicher gewöhnungsbedürftig, aber häufig deutlich schneller als der Einsatz einer Entwicklungsumgebung. Bis *Eclipse* oder *Microsoft Visual* gestartet sind, haben Sie längst *make all* und das Programm ausgeführt!

Die beschriebenen Schritte können Sie auch auf *Linux* übertragen. Dort öffnen sie eine Shell, beispielsweise eine *Konsole* oder ein *X-Term* statt *Cygwin*. Alles andere verläuft analog; welchen Editor Sie verwenden ist wieder Ihnen überlassen.