

Dokumentieren mit *Doxygen*

Prof. Dr. Manfred Brill

Oktober 2003

Inhaltsverzeichnis

1	Doxygen	1
2	Dokumentation des Quelltexts	2
3	Konfiguration von <i>Doxygen</i>	3
4	<i>Doxygen</i> ausführen	5
	Literatur	5

1 Doxygen

Zur Software-Entwicklung gehört auch die Dokumentation dazu – das ist eigentlich eine Selbstverständlichkeit. Sie kennen das bestimmt, endlich sind alle Klassen implementiert, die Funktionalität getestet – jetzt soll noch eine Dokumentation erstellt werden. Wenn Sie mit *Java* programmiert haben kennen Sie eventuell das Werkzeug *JavaDoc*. Damit kann mit Hilfe von Kommentaren, die in den Quelltexten enthalten sind eine Dokumentation erstellt werden.

Ein mit *JavaDoc* vergleichbares Werkzeug ist die Freeware *Doxygen*. Sie erstellen Ihre Software und kommentieren dabei; je nach Ihren Angewohnheiten eventuell etwas mehr als zuvor. Sie werden bemerken, dass die Arbeit mit *Doxygen* schnell und einfach in Ihre Implementierungsarbeit integriert werden kann. *Doxygen* unterstützt neben *Java* die Programmiersprachen *C*, *C++* und *PHP*. Eine ganze Menge von großen Projekten, beispielsweise *VTK* oder *OpenSG* verwenden *Doxygen* zur Erstellung der Dokumentation.

Doxygen kann neben einer *HTML*-Ausgabe auch *LaTeX*, *RTF* oder *UNIX-man*-Seiten erzeugen. Wir werden uns in diesem Text auf die *HTML*-Ausgabe konzentrieren und nur eine knappe Einführung geben. Viele Tips und Einstellungsmöglichkeiten finden Sie in [vH02].

Drei Schritte trennen Sie von einer Dokumentation mit *Doxygen*:

1. Sie müssen in Ihren Quelltexten entsprechende Kommentare einbauen; mehr dazu finden Sie in Abschnitt 2.
2. Sie müssen *Doxygen* konfigurieren; mehr dazu finden Sie in Abschnitt 3.
3. Sie müssen *Doxygen* ausführen; mehr dazu finden Sie in Abschnitt 4.

2 Dokumentation des Quelltexts

Der Kern der Dokumentation mit *Doxygen* besteht darin, spezielle Kommentarblöcke in Ihr Programm einzubauen. Für jedes Element des Quellcodes gibt es für *Doxygen* zwei Beschreibungen: eine *Kurzbeschreibung* und eine *ausführliche Beschreibung*. Die Kurzbeschreibung besteht in der Regel aus einer Zeile.

Es gibt mehrere Möglichkeiten, die beiden Beschreibungen in den C++-Quelltext einzubauen. In diesem Text beschreibe ich die Variante, die ich in meinen Quelltexten verwende - und die Sie also in dieser Form im Praktikum erhalten. Es ist Ihnen überlassen, sich diesem Vorbild anzuschließen oder nach einem Blick in [vH02] einen eigenen Stil zu wählen.

Eine Kurzbeschreibung erhalten Sie durch eine Zeile der Form

```
//! Eine Kurzbeschreibung.
```

Sie können die Kurzbeschreibung bei der Deklaration oder der Definition stehen haben; eine gute Regel ist, die Kurzbeschreibung zur Deklaration zu schreiben. Hier ein Beispiel für eine Kurzbeschreibung einer Klasse:

```
//! Virtuelle Basisklasse für Standardgeometrien in OpenGL
class vlGeometry
{
    public:
        //! Defaultkonstruktor
        vlGeometry(void);
```

Das Beispiel enthält zwei Kurzbeschreibungen: einmal für die Klasse `vlGeometry`, einmal für den Defaultkonstruktor. Auf die gleiche Weise können Sie alle weiteren Methoden der Klasse und Datenelemente der Klasse dokumentieren. Die Beschreibungen müssen *vor* dem dokumentierten Quelltext stehen.

```
protected:
    //! Die Parameter für die Auflösung der Standardgeometrien
    int resU, resV;
```

Für eine ausführliche Beschreibung, die aus mehreren Zeilen bestehen kann, kann ein Kommentar der Form

```
/*!
    Eine ausführliche Beschreibung,
    die aus mehreren Zeilen besteht. Die führenden Leerzeichen
    in den Zeilen werden beim Erzeugen der Dokumentation ignoriert!
*/
```

verwendet werden. Sie sehen, eigentlich müssen Sie Ihre Kommentare nur durch das Ausrufezeichen ergänzen – das ist doch wirklich nicht viel, oder?

Eine gute Regel ist, die ausführliche Beschreibung zur Definition einer Klasse oder Methode zu schreiben. Sie sollten die Deklaration in eine *Header-Datei* schreiben. Enthält diese Datei auch die ausführlichen Beschreibungen wird dies für die Benutzer Ihrer Programme leicht unübersichtlich – dafür erzeugen Sie ja die Dokumentation.

Das folgende Beispiel zeigt die ausführlichen Beschreibungen für die Klasse aus dem bereits gezeigten Beispiel:

```
#include "vlGeometry.h"
/*!
Die Basisklasse enthält zwei ganze Zahlen, mit deren Hilfe die
abgeleiteten Klassen jeweils die Auflösung festlegen können.
Für das Setzen und Abfragen dieser Parameter werden entsprechende
inline-Funktionen definiert.
```

Darüberhinaus wird die rein virtuelle Funktion `::draw()` deklariert, die von den instanzierbaren Klassen implementiert werden muss.

```
*/
/*!
Die Auflösungen werden jeweils mit 1 initialisiert.
*/
vlGeometry::vlGeometry(void)
{
    resU = 1;
    resV = 1;
}
```

Doxygen kennt eine Menge von *Tags*, die für die jeweiligen Text-Systeme in Links oder Index-Einträge umgewandelt werden. Im folgenden Beispiel sehen Sie den Tag `\param`, mit dem Sie sich speziell auf Parameter einer dokumentierten Methode beziehen können. In *HTML* wird dadurch ein Hyperlink auf die Datei erzeugt, in der diese Parameter dokumentiert sind. Mit `\sa` können Sie einen Verweis erzeugen; die Abkürzung steht für „see also“.

```
/*!
    \param u,v sind die beiden als ganze Zahlen
    deklarierten Auflösungsparameter, die in den von vlGeometry
    abgeleiteten Klassen entsprechend verwendet werden können.

    \sa resU \sa resV
*/
vlGeometry::vlGeometry(int u, int v)
{
    resU = u;
    resV = v;
}
```

Das Ergebnis dieser Dokumentation ist eine Menge von *HTML*-Dateien; die Einstiegsseite ist die Datei `index.html`. Was auf dieser Einstiegsseite enthalten ist können Sie ebenfalls durch einen entsprechenden Kommentar festlegen. Dazu verwenden Sie den Tag `\mainpage` und Text, den Sie entsprechend verfassen. Hilfreich sind auch die Tags `\author`, `\date` und `\image`:

```
/*!
\mainpage

\author Manfred Brill
\date Oktober 2003

\image html vislab.jpg

Ergänzen Sie diesen Text und dokumentieren Sie Ihre Klassen mit
Hilfe von Doxygen!

Für die Erstellung der HTML-Seiten verwenden Sie doxywizard oder das
Kommando doxygen in der Cygwin-Shell. Die Konfiguration finden Sie in
der Datei doxy. In der Shell entspricht dies dann dem Kommando
<<doxygen doxy>>.
*/
```

3 Konfiguration von Doxygen

Den wichtigsten Schritt haben Sie schon hinter sich – Sie haben die entsprechenden Kommentare in Ihren Quelltext eingebaut. Bevor die Dokumentation erstellt wird müssen Sie jetzt festlegen, in

welchem Format die Dokumentation zur Verfügung stehen soll; wo Sie abgespeichert wird und welche Bestandteile die Dokumentation enthalten sind.

Dabei können Sie die meisten Grundeinstellungen so belassen, wie sie von *Doxygen* als Default vorgeschlagen werden. Sie erhalten auf diese Weise beispielsweise eine komplette *Cross-Referenz* Ihres Projekts. Die Einstellung für *Doxygen* stehen in einer Konfigurationsdatei. Das Format entspricht einem *Makefile*. In einer Shell, sei es *cmd.exe* in *MS Windows* oder eine *UNIX-Shell* können Sie mit

```
doxygen -g <config-file>
```

eine entsprechende Datei erzeugen. Geben Sie keinen Namen an und nur die Option *-g*, dann wird eine Datei mit dem Namen *Doxy-file* erzeugt.

Wollen Sie anschließend die Einstellungen überprüfen oder verändern, dann können Sie dies mit Hilfe eines ASCII-Editors durchführen. Eine Alternative dazu ist das Werkzeug *doxywizard*, das ein kleines Frontend zum Erzeugen, Verändern und auch Speichern von Konfigurationsdateien unter *Microsoft Windows* anbietet. In *Abbildung 1* sehen Sie eine Darstellung des Wizards.

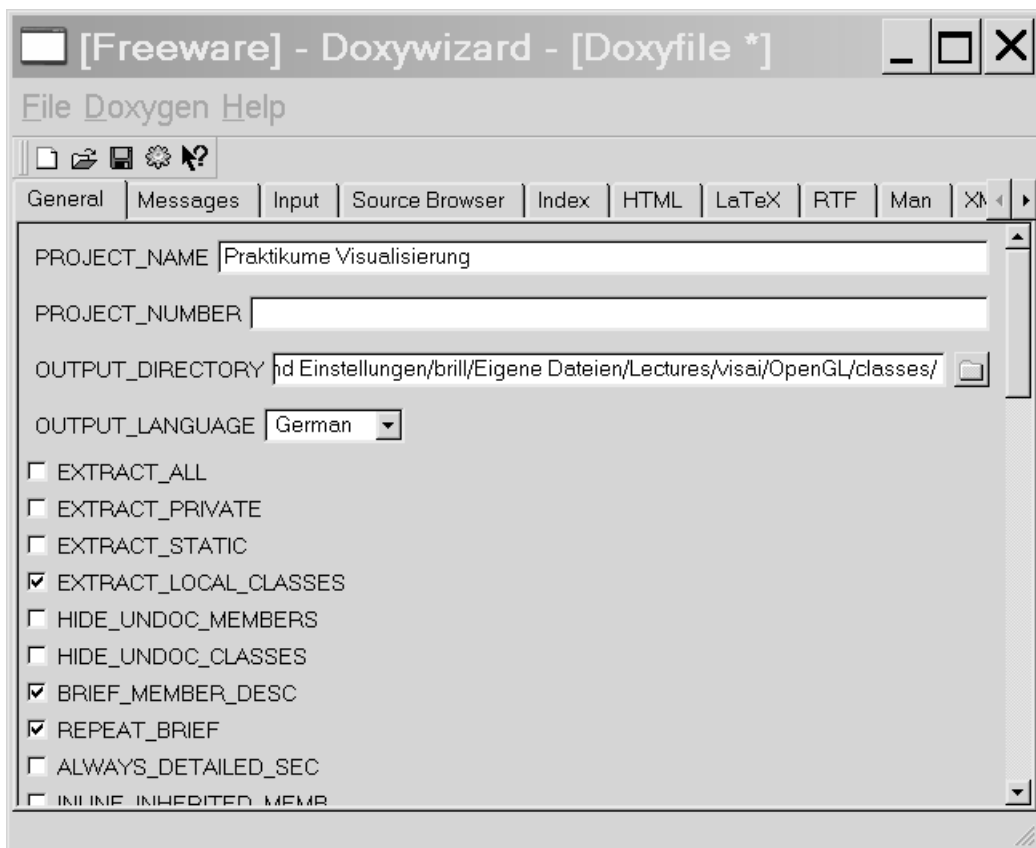


Abbildung 1: Der *doxywizard* in Aktion

Zwei Einträge sind wichtig. Einmal müssen Sie das Verzeichnis angeben, in dem die Ausgabe erfolgen soll. *Doxygen* legt *HTML*-Ausgaben in einem Unterverzeichnis mit dem Namen *html*, *LaTeX*-Ausgaben im Unterverzeichnis *latex* ab. Und Sie müssen natürlich festlegen, wo die Eingaben, also die zu dokumentierenden Quelltexte liegen. Dies können Sie unter dem Reiter *INPUT* durchführen. Für kleine Projekte reicht es, die Konfigurationsdatei in das Verzeichnis zu speichern, in dem sich die zu bearbeitenden Quellen befinden. Bei großen Projekten, die vielleicht sogar einen ganzen Baum von Unterverzeichnissen mit Quellen enthalten, geben Sie als *INPUT* die Wurzel des Baums an. Sie können auch Muster wie **.cpp* und **.h* angeben, um sicher zu stel-

len, dass die richtigen Quelldateien bearbeitet werden. Und Sie müssen den *RECURSIVE*-Tag auf *YES* stellen!

Sollen die Quellen in der Dokumentation enthalten sein, dann stellen Sie den Wert für *INLINE_SOURCES* auf *YES*; setzen Sie *SOURCE_BROWSER* auf *YES*, falls Sie eine *Cross-Referenz* Ihrer Quellen erzeugen möchten.

4 Doxygen ausführen

Dies wird jetzt der einfachste Schritt. In einer Shell geben Sie

```
doxygen <config-file>
```

ein. Damit wird, je nach Konfigurationseinstellung, die gewünschte Ausgabe erzeugt. Haben Sie kein Ausgabeverzeichnis mit dem Tag *OUTPUT_DIRECTORY* festgelegt dann wird die Ausgabe im aktuellen Verzeichnis abgelegt.

Tip:

Sie können die Erzeugung der Dokumentation in Ihren Makefile integrieren. Damit wird es möglich, *Doxygen* in *Eclipse* anzustoßen. Alles was Sie dazu benötigen ist ein entsprechendes *make-Target*.

Angenommen, die Konfigurationsdatei hat den Namen `doxy`, dann reicht ein Target `doc` und die Regel `doxygen doxy`. Mehr zur Arbeit mit Makefiles finden Sie in [Bri03].

Für die Ansicht der *HTML*-Ausgabe müssen Sie einen Browser verwenden, der *CSS* unterstützt. Die *LaTeX*-Ausgabe muss anschließend natürlich bearbeitet werden. Dies können Sie wie gewohnt durchführen; die Hauptdatei trägt als Default den Namen `refman.tex`. Oder, falls Sie *make* installiert haben, mit der Eingabe von *make* in dem Verzeichnis, in dem *Doxygen* die *LaTeX*-Ausgabe abgelegt hat. Dadurch wird eine Datei `refman.dvi` erzeugt. Wollen Sie eine Postscript-Datei, dann geben Sie *make ps* ein; vorausgesetzt Sie haben das Programm *dvips* installiert. Sie können mit dem Makefile auch *PDF* erzeugen, vorausgesetzt Sie haben *ghostscript* installiert. Geben Sie einfach *make pdf* ein. Ist in der Konfiguration der Tag *PDF_HYPERLINKS* auf *YES* gesetzt, dann werden in der *PDF*-Datei Links erzeugt!

Literatur

[Bri03] BRILL, MANFRED: *Programmieren mit Cygwin und UNIX*. Fachhochschule Kaiserslautern, 2003.

[vH02] HEESCH, DIMITRI VAN: *User Manual for Doxygen 1.2.17*, 2002.