

# Computergrafik Praktikum

## Hilfe zum Rahmenprogramm

6. November 2017

### Inhaltsverzeichnis

<b>1</b>	<b>CMake</b>	<b>2</b>
<b>2</b>	<b>Rahmenprogramm entpacken und mit CMake erstellen</b>	<b>2</b>
<b>3</b>	<b>Startprojekt ändern</b>	<b>4</b>
<b>4</b>	<b>Programm für die Abgabe erstellen</b>	<b>5</b>
<b>5</b>	<b>Neue Quelltextdateien hinzufügen</b>	<b>6</b>
<b>6</b>	<b>Ein neues Projekt hinzufügen</b>	<b>7</b>











Wenn Sie darüber hinaus Fragen haben oder Hilfe benötigen, können Sie gerne eine E-Mail an [markus.miller@hm.edu](mailto:markus.miller@hm.edu) schreiben.

# 1 CMake

Für die Erstellung der Projekte wird CMake (<http://www.cmake.org>) benötigt. CMake ist ein universelles Build-System, dass Visual-Studio Projektmappen oder Makefiles generiert. CMake gibt es für nahezu jede Plattform, allerdings wurden die Praktikumsaufgaben nur auf Windows 7 und Arch Linux getestet. Das Kernelement von CMake sind die „CMakeLists.txt“ Dateien, die die Build-Konfiguration enthalten. Wenn etwas an der Build-Konfiguration geändert werden muss, dann müssen die CMakeLists.txt angepasst werden, z.B. für das Hinzufügen von neuen Quelltextdateien oder von neuen Projekten. Eine entsprechende Anleitung finden Sie in diesem Dokument. Für Linux werden zusätzlich die „X11“ Bibliotheken benötigt. Diese sind über den jeweiligen Paketmanager der Distribution installieren.

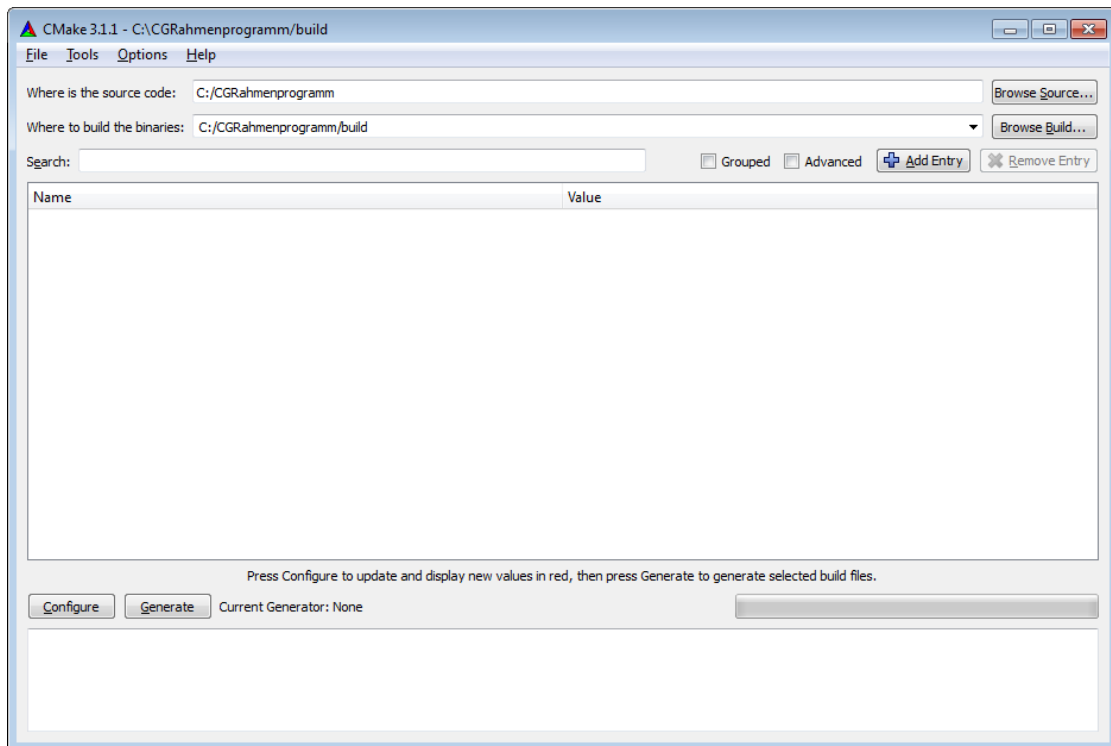
## 2 Rahmenprogramm entpacken und mit CMake erstellen

Nach dem Entpacken der ZIP-Datei sollte nun folgende Ordner-Struktur vorhanden sein:

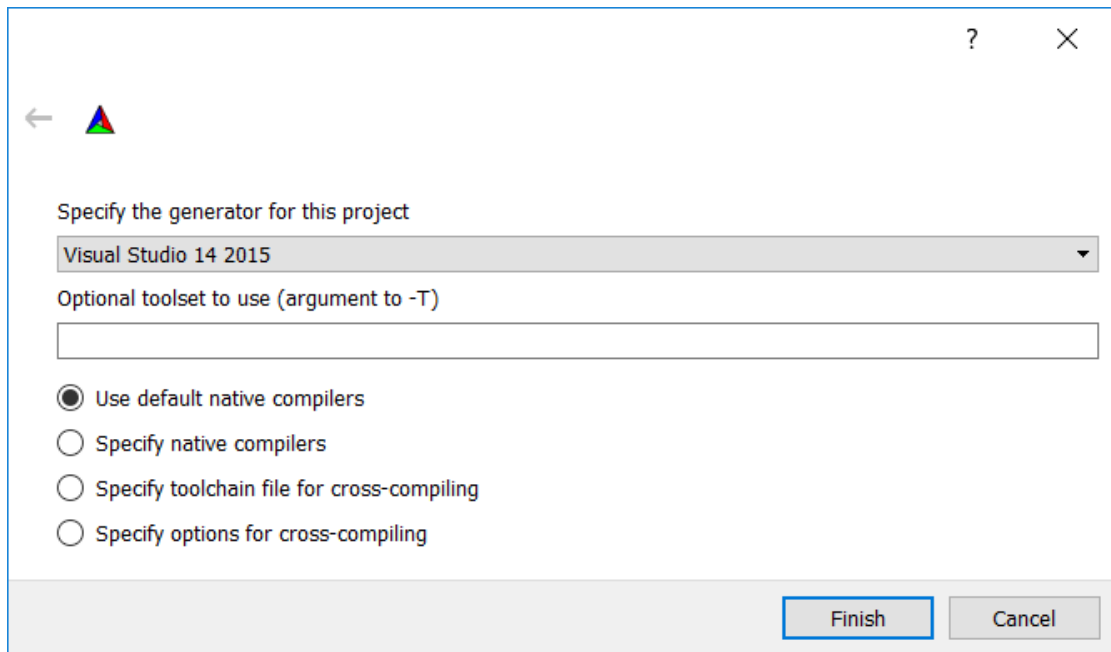
	A1_Bildverarbeitung	21.11.2016 10:42	Dateiordner
	A1_Versuch1a	30.06.2017 14:16	Dateiordner
	A1_Versuch1b	06.11.2017 20:11	Dateiordner
	A3_Normalenvektoren	30.06.2017 14:18	Dateiordner
	cmake	06.11.2017 20:11	Dateiordner
	Dependencies	06.11.2017 20:11	Dateiordner
	ImageLoader	26.07.2016 16:46	Dateiordner
	Templates	26.07.2016 16:46	Dateiordner
	Texturen	26.07.2016 16:46	Dateiordner
	CMakeLists.txt	28.01.2016 11:42	TXT-Datei

**Achtung:** Liegt der Inhalt der ZIP-Datei an irgendeinem Ort im Netzlaufwerk, darunter fallen Ordner im eigenen Verzeichnis auf dem Netzlaufwerk, aber auch der Desktop, der über das Netzlaufwerk synchronisiert wird, *können*, von Laufzeitproblemen beim Kompilieren des Rahmenprogramms ganz abgesehen, Pfadprobleme mit CMake, bzw. in den entstandenen Projektdaten für die gewünschte Entwicklungsumgebung auftreten.

Im Folgenden wird nun gezeigt, wie man eine Visual Studio Projektmappe mit CMake erstellt. Für andere Plattformen sind die Schritte nahezu analog. Als erstes muss CMake-GUI aus dem Startmenü gestartet werden (Befehl für Linux: cmake-gui). Es erscheint folgende Oberfläche:



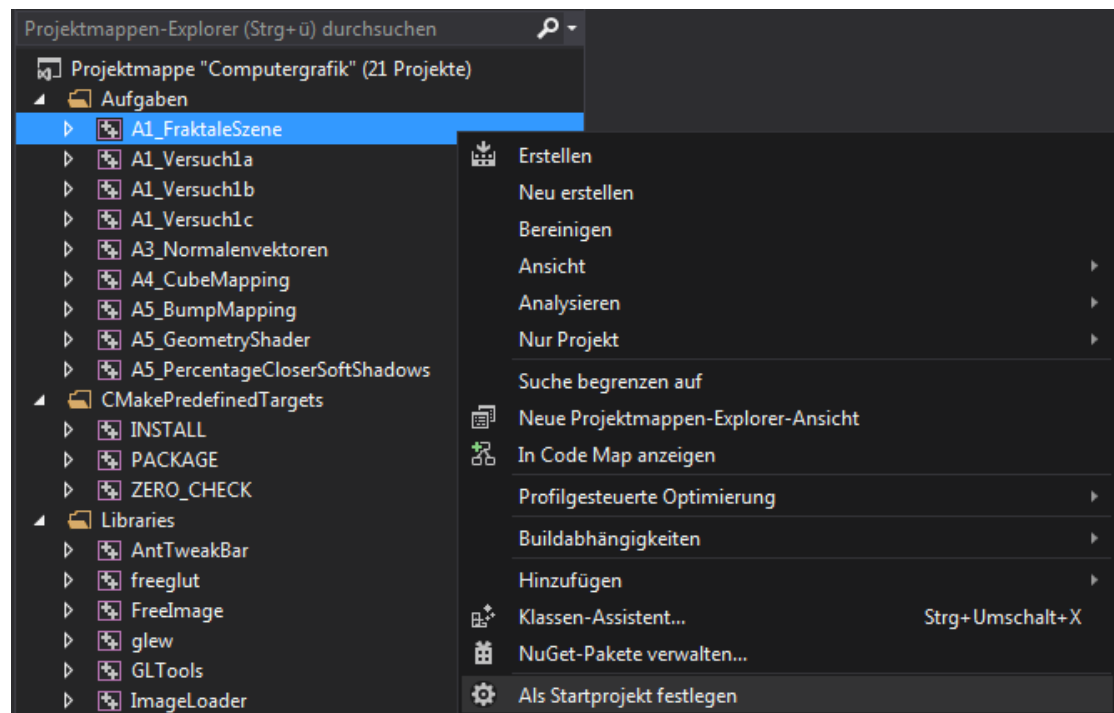
Nun muss das Quellcode-Verzeichnis mit „Browse Source...“ ausgewählt werden (= das Verzeichnis mit der oben genannten Ordnerstruktur). Im zweiten Schritt muss ein Build-Ordner mit „Browse Build...“ ausgewählt werden. CMake erlaubt „out of source“ Builds, bei dem die Binärdateien in separaten Verzeichnissen erstellt werden. Wenn der Quellcode der Praktikumsaufgaben kopiert werden muss, z.B. für die Heimarbeit, kann der Build- Ordner gelöscht werden, wodurch die Dateigröße enorm schrumpft. Daheim können Sie die Projektmappen von CMake neu generieren lassen. Für „out of source“ Builds wird ein extra „Build“ Ordner angegeben, z.B. „build“ wie im Screenshot oben. Nachdem die beiden Pfade gesetzt sind, erfolgt die Projekt-Konfiguration mit „Configure“. Im darauffolgenden Dialog muss die entsprechende Entwicklungsumgebung angegeben werden, z.B. Visual Studio 14 - 2015 für die Laborrechner. Für Linux können hier entweder Makefiles, KDevelop- Projekte oder Eclipse-Projektdateien erstellt werden. Bei den Optionen noch „Use Native Compiler“ auswählen.



Nun konfiguriert CMake sämtliche benötigte Bibliothek- und Include-Pfade. Wenn eine Bibliothek nicht gefunden wurde, erscheint eine Fehlermeldung. Die Pfade müssen dann im entsprechenden Eintrag manuell hinzugefügt werden bzw. die Bibliothek installiert und „Configure“ erneut gestartet werden. Das sollte allerdings nicht passieren, da alle Bibliotheken als Quellcode mitgeliefert und erstellt werden. Wenn alles geklappt hat, kann die Projektmappe bzw. das Makefile mit „Generate“ erstellt werden. Im „Build“ Ordner befinden sich die Visual Studio Projektmappe bzw. das Makefile. Danach kann die Projektmappe mit Visual Studio geöffnet und das Rahmenprogramm erstellt werden. Jede weitere Änderung an den CMakeLists.txt Dateien wird bei Visual Studio automatisch über Makros aktualisiert.

### 3 Startprojekt ändern

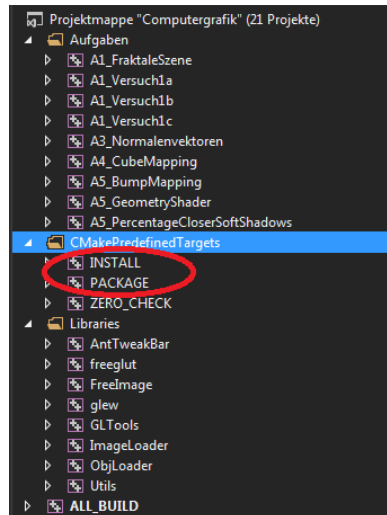
Standardmäßig wird das Projekt „ALL\_BUILD“ als Startprojekt festgelegt. Jedes Mal wenn man „F5“ bzw. der Debugger gestartet wird, wird das Startprojekt aufgerufen. Das Startprojekt wird mit fester Schrift dargestellt. Das Startprojekt kann geändert werden, indem in der Projektmappe mit der rechten Maustaste auf das gewünschte Projekt geklickt und „Als Startprojekt festlegen“ ausgewählt wird (Abb).



**Hinweis falls das Rahmenprogramm nicht compiliert:** bei manchen Entwicklungsumgebungen findet der Compiler die Shaderfiles bzw. Texturen nur, wenn sie sich im build-Ordner befinden, oder der Pfad zu den Shaderfiles vom build-Ordner zum Source-Ordner neu gesetzt wird (für Shaderfiles muss z.B. in der Methode `gltLoadShaderPairWithAttributes("../A3_Normalenvektoren/VertexShader.glsl", "../A3_Normalenvektoren/FragmentShader.glsl", ...)` der Pfad um `"../A3_Normalenvektoren/` ergänzt werden und für Texturen z.B. `"../Texturen/cubemap/positive_x.jpg"`).

## 4 Programm für die Abgabe erstellen

Im Rahmenprogramm ist eine Installationsroutine enthalten, die die Praktikumsabgabe erleichtern soll. Es kopiert die ausführbaren Dateien und die Quelltextdateien in den „Abgabe“ Ordner im Hauptverzeichnis. Um die Aufgaben zur Abgabe vorzubereiten muss das Projekt „INSTALL“ erstellt werden. Dazu mit der rechten Maustaste auf das Projekt klicken und „Erstellen“ auswählen. Nun werden die Dateien in den „Abgabe“ Ordner kopiert. Für Linux muss entsprechend „make install“ aufgerufen werden. Zusätzlich gibt es die Möglichkeit, den „Abgabe“ Ordner gleich mit ZIP zu komprimieren. Dazu das Projekt „PACKAGE“ erstellen. Die Zip-Datei befindet sich im Build- Ordner.



Für die Abgabe bitte immer die „Release“ Konfiguration auswählen

## 5 Neue Quelltextdateien hinzufügen

Um neue Quelltextdateien hinzuzufügen, muss die „SOURCES“ Variable der CMakeLists.txt des Projekts angepasst werden, z.B. im Ordner A3\_Normalenvektoren. Hier wird eine neue C++ Datei „MeineNeueCPPDatei.cpp“ hinzugefügt.

```
SET(PROJECT_NAME A3_Normalenvektoren)

set(SOURCES Aufgabe3.cpp
           MeineNeueCPPDatei.cpp
           VertexShader.glsl
           FragmentShader.glsl
)
set(LIBRARIES GLTools AntTweakBar freeglut ImageLoader)
```

Anschließend muss CMake erneut aufgerufen werden (Visual Studio: Projektmappe erstellen, den Rest machen die Makros. Linux: „Generate“ in der CMake-GUI erneut aufrufen).

**Beachten Sie, dass die Quelltextdatei auch wirklich vorhanden ist, sonst liefert CMake einen Fehler.**

Wenn das bei Visual Studio mal passiert ist, finden Sie eine leere Projektmappe vor. Um das zu beheben muss CMake-GUI erneut aufgerufen und „Generate“ ausgeführt werden.

## 6 Ein neues Projekt hinzufügen

Um ein neues Projekt hinzuzufügen, muss zuerst die CMakeLists.txt im Hauptverzeichnis angepasst werden. Am Ende der CMakeLists.txt werden die einzelnen Projekte hinzugefügt. Dort ist ein neuer Eintrag mit „add\_subdirectory(MeinNeuesProjekt)“ hinzuzufügen. Der Projektname ist natürlich beliebig.

```
# Die Projekte hinzufügen
add_subdirectory(A1_Versuch1a)
add_subdirectory(A1_Versuch1b)
add_subdirectory(A1_Bildverarbeitung)
add_subdirectory(A3_Normalenvektoren)

# neues Projekt hinzufügen
add_subdirectory(MeinNeuesProjekt)|
```

Im zweiten Schritt muss ein neuer Ordner mit **gleichen** Namen im Hauptverzeichnis angelegt werden. Nun muss im Ordner eine CMakeLists.txt erstellt werden. Als Basis kann die CMakeLists.txt aus dem „Template“ Ordner kopiert werden. Anschließend sind die mit „TODO“ gekennzeichneten Variablen in der CMakeLists.txt anzupassen.

```
### CMakeLists.txt Template für neue Projekte

#TODO Projektnamen ändern
SET(PROJECT_NAME MeinNeuesProjekt)
#TODO Quelldateien hinzufügen, auch GLSL Shader.
#mit Leerzeichen getrennt
set(SOURCES Foo.cpp Bar.cpp)
#TODO Bibliotheken anpassen
set(LIBRARIES GLTools AntTweakBar freeglut)
```

Anschließend muss CMake erneut aufgerufen werden (bei Visual Studio erledigen das die Makros).