Qt5-Installationsanleitung

Wichtig: Qt ist kostenlos und man muss sich nicht registrieren!

Inhaltsverzeichnis

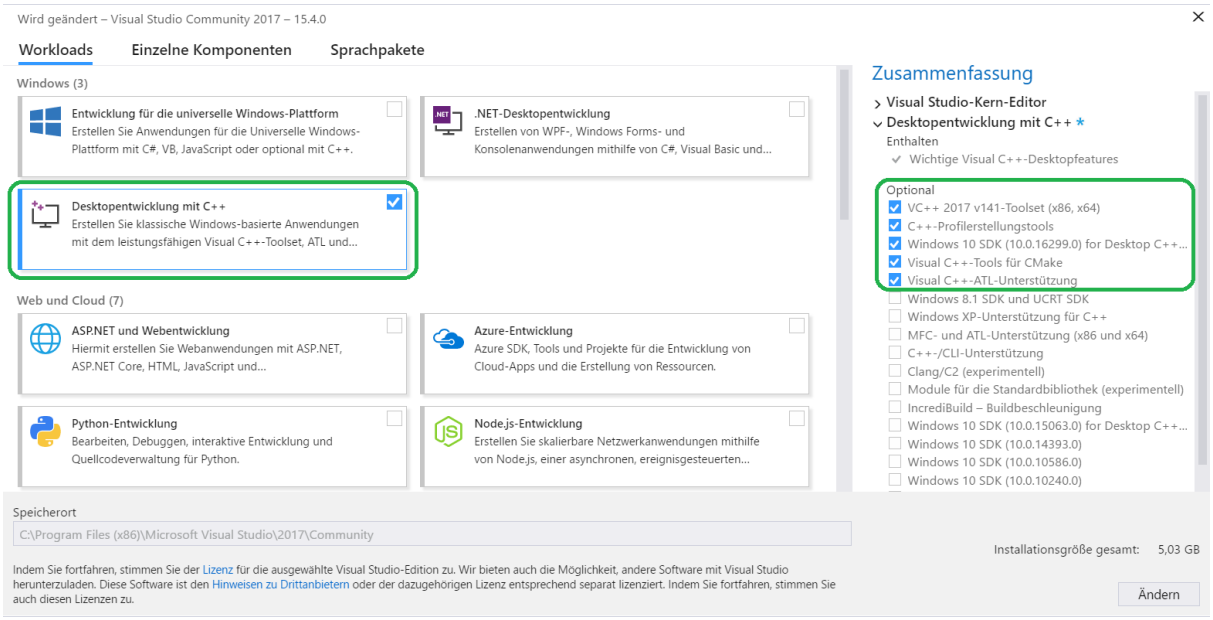
Windows Abhängigkeiten 2
Qt5 Offline Installer 4
QtCreator 7
Linux (apt) 8
Linux (Source) 9
macOS (Homebrew) 10
CMake Projektdateien generieren (Win/Linux/Mac) 11

Windows Abhängigkeiten

Vor der Qt5-Installation sollten CMake, ein Visual Studio mit C/C++-Compiler (nicht „Visual Studio Code“!) und ein aktueller Grafikkartentreiber installiert werden, damit Qt sie beim ersten Start gleich finden und konfigurieren kann. Für Visual Studio wird ein Microsoft-Account benötigt.

- CMake 3.x: https://cmake.org/download/#rc
- Visual Studio 2019
  - Community: https://visualstudio.microsoft.com/thank-you-downloading-visual-studio/?sku=Community&rel=16
  - Enterprise: https://azureforeducation.microsoft.com/devtools
- Visual Studio 2017
  - Community: https://visualstudio.microsoft.com/thank-you-downloading-visual-studio/?sku=Community&rel=15
- Grafikkartentreiber:
  - AMD: https://www.amd.com/de/support
  - Intel: https://downloadcenter.intel.com/De/product/80939/Grafiktreiber
  - Nvidia: http://www.nvidia.de/Download/index.aspx?lang=de

Da Visual Studio auch für andere Programmiersprachen verwendet werden kann, muss der C/C++-Compiler explizit installiert werden. Beim ersten Start sollte daher „Desktopentwicklung mit C++“ ausgewählt werden. **Wollen Sie QtCreator anstatt Visual Studio als Entwicklungsumgebung verwenden, notieren/merken Sie sich Ihre „Windows 10 SDK“ Version (siehe rechts im Bild)!**

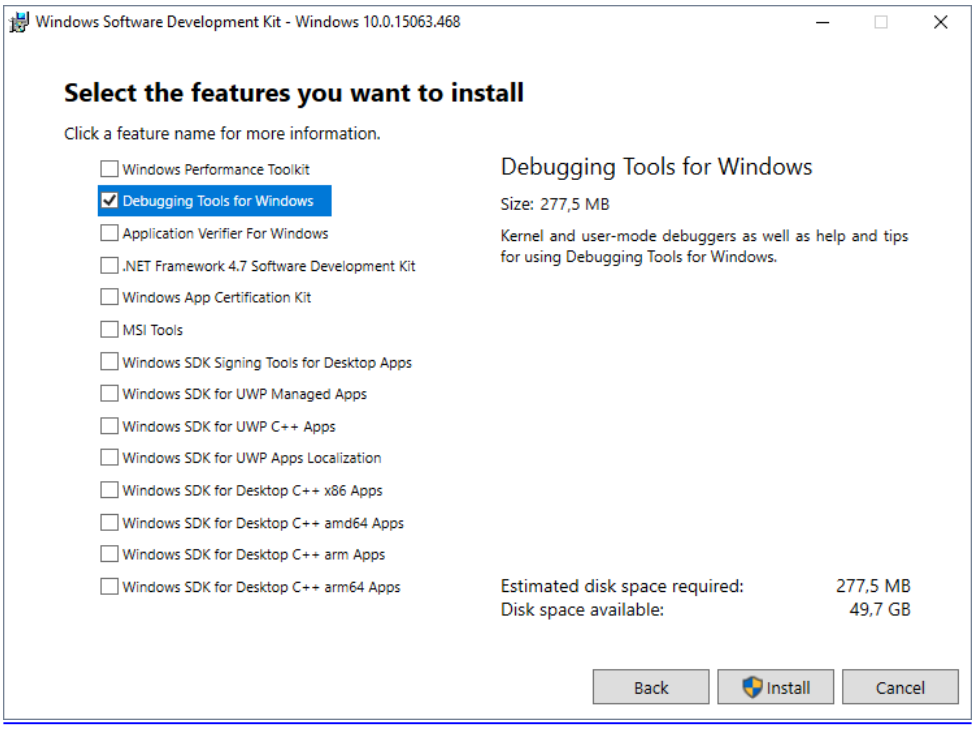


Solltet Sie keinen Zugriff mehr auf das Fenster haben, können Sie es über „Extras“ > „Tools und Features abrufen …“ erneut öffnen.

Sollten Sie Visual Studio mittels CMake verwenden wollen, starten Sie jetzt den PC neu, da CMake sonst keine Visual Studio Instanz findet. Fahren Sie danach mit dem nächsten Kapitel fort.

Wollen Sie den QtCreator verwenden, muss noch der Debugger CDB installiert werden. Für Visual Studio 2019 kann das passende WDK auf https://developer.microsoft.com/en-us/windows/hardware/

windows-driver-kit heruntergeladen werden. Achten Sie auf die gleiche Version wie beim „Windows 10 SDK“ (siehe vorheriges Bild). Für Visual Studio 2017 kann die letzte Version 10.0.17763 hier heruntergeladen werden: https://go.microsoft.com/fwlink/?linkid=2026156

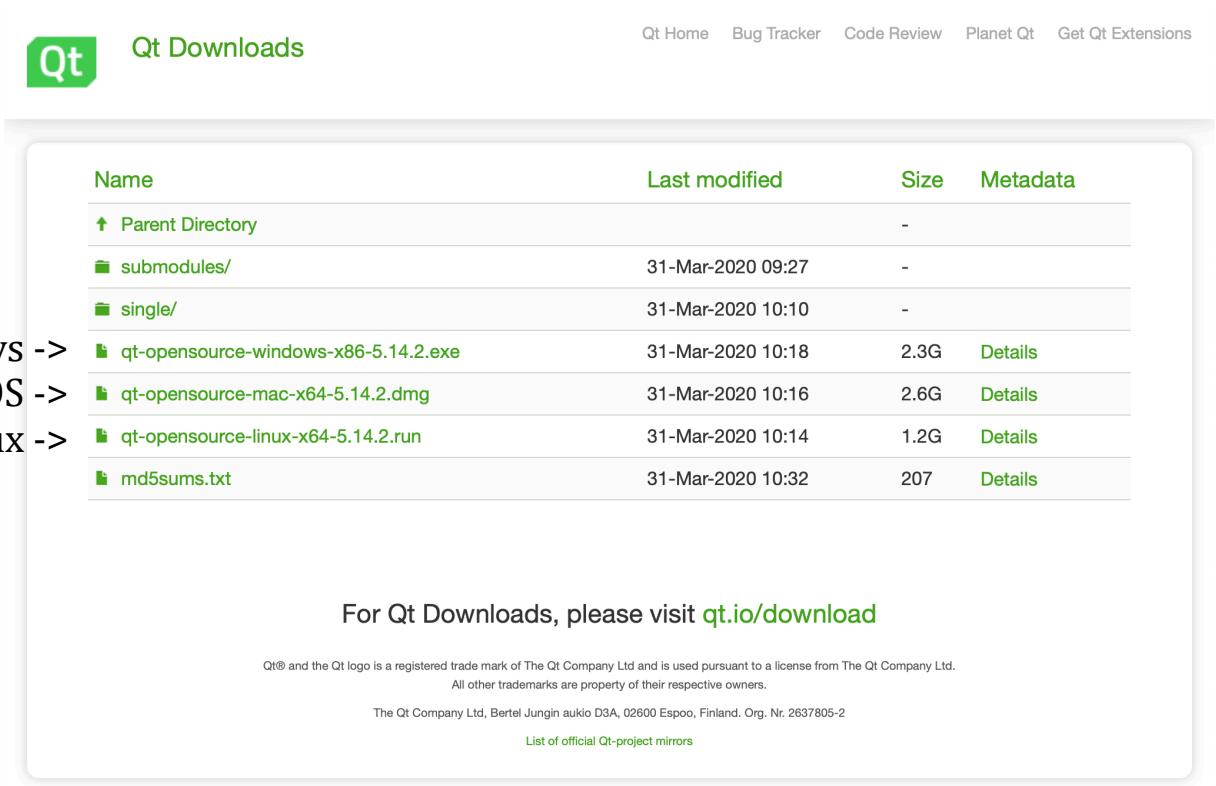


Fahren Sie mit dem nächsten Kapitel „Qt5 Offline Installer“ fort.

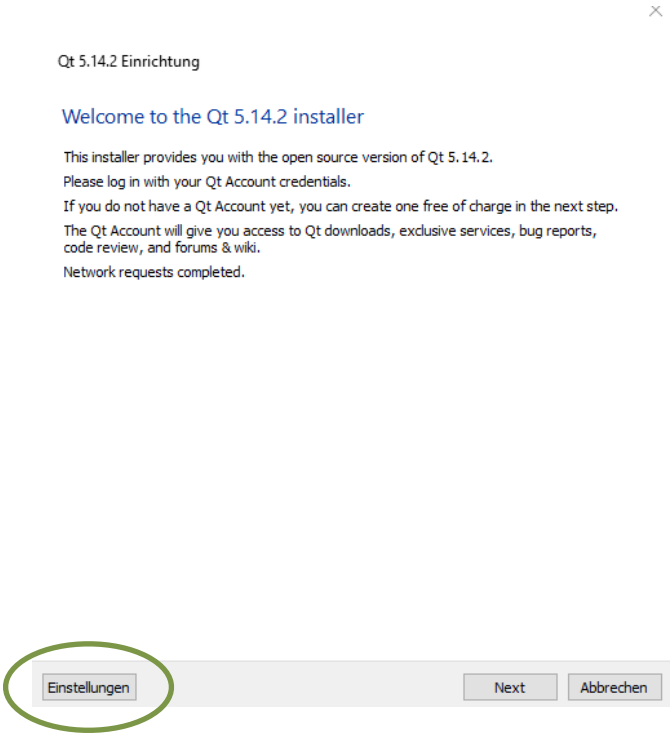
Qt5 Offline Installer

In diesem Kapitel wird die Installation des Qt5 Offline Installer beschrieben, da hier eine Qt-Anmeldung ausgelassen werden kann. Für Linux/macOS-User sind in späteren Kapiteln gängige Alternativen beschrieben. Installieren Sie vorher CMake, einen Compiler und einen aktuellen Grafikkartentreiber.

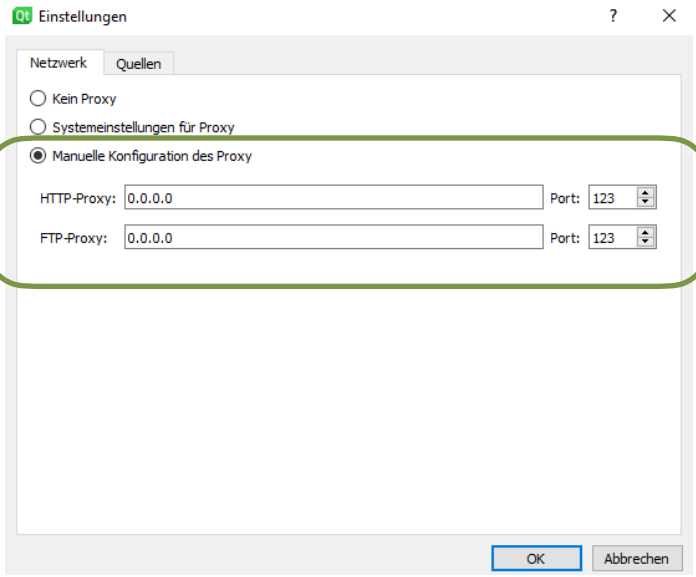
Laden Sie den Qt5 Offline-Installer für Ihr Betriebssystem herunter (http://download.qt.io/official_releases/qt/5.14/5.14.2/).



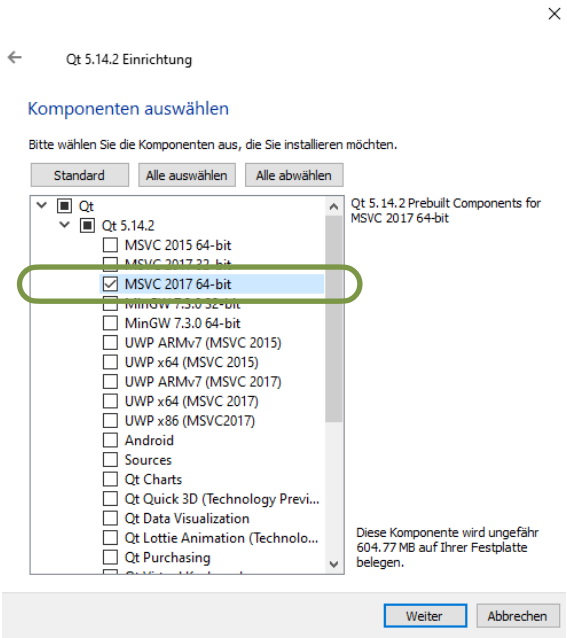
Windows ->
macOS ->
Linux ->



Stellen sie in den Einstellungen einen ungültigen Proxy ein, z.B. „0.0.0.0“ und Port „123“. So kann die Anmeldung übersprungen werden.

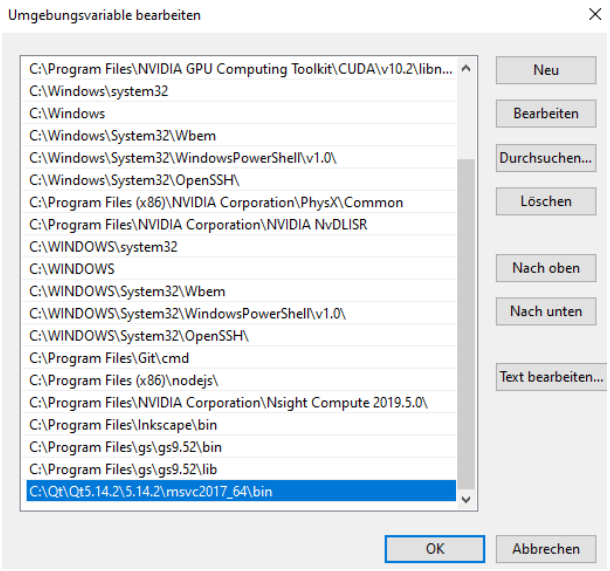


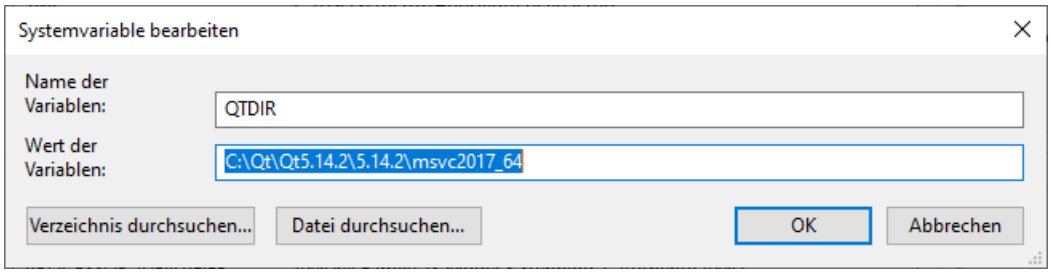
Wählen Sie unter Windows nun „MSVC 2017 64-bit“ aus, stimmen der Lizenz zu und fahren die Installation fort. Unter Linux und macOS muss der entsprechende Compiler (gcc/clang/…) ausgewählt werden. Die anderen Komponenten können deaktivieren bleiben, da sie in diesem Kurs nicht benötigt werden und so Speicherplatz gespart werden kann.



Nach der CMake- und Qt5-Installation müssen unter Windows noch zwei Umgebungsvariablen angepasst werden. Öffnen Sie „System“ durch Drücken von „Windows Taste“ + „PAUSE“ oder über „Systemsteuerung“ > „System und Sicherheit“ > „System“. Wählen Sie links „Erweitere Systemeinstellungen“ und dann unten den Knopf „Umgebungsvariablen“.

Unter „Benutzervariablen für …“ stehen Variablen für den aktuellen Windows-Account, unter „Systemvariablen“ für alle Windows-Accounts auf dem Gerät. Tragen Sie in einem Bereich Ihrer Wahl die Variable „QTDIR“ mit Qt-Installationspfad bis zum msvc-Ordner ein, z.B. „C:\Qt\Qt5.14.2\5.14.2\msvc2017_64“. Ergänzen Sie die existierende PATH-Variable um den Qt-Installationspfad bis hin zum bin-Order, z.B. „C:\Qt\Qt5.14.2\5.14.2\msvc2017_64\bin“. Die DLLs werden beim Starten der Übungsaufgaben sonst nicht gefunden. **Schließen Sie alle Entwicklungsrelevanten Programme (CMake, Visual Studio etc.) oder starten am Besten den PC neu, da die geänderten Umgebungsvariablen sonst keine Wirkung zeigen.**



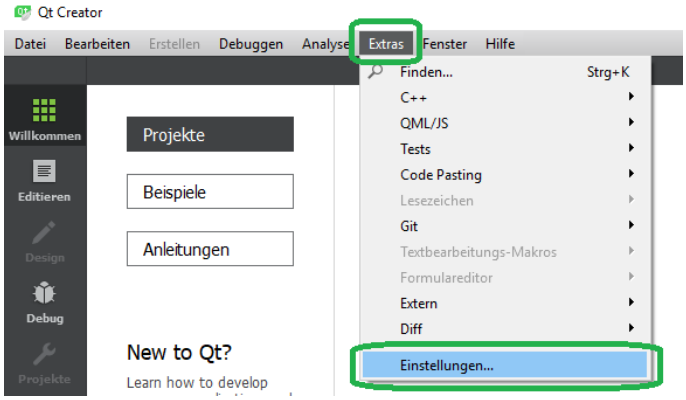


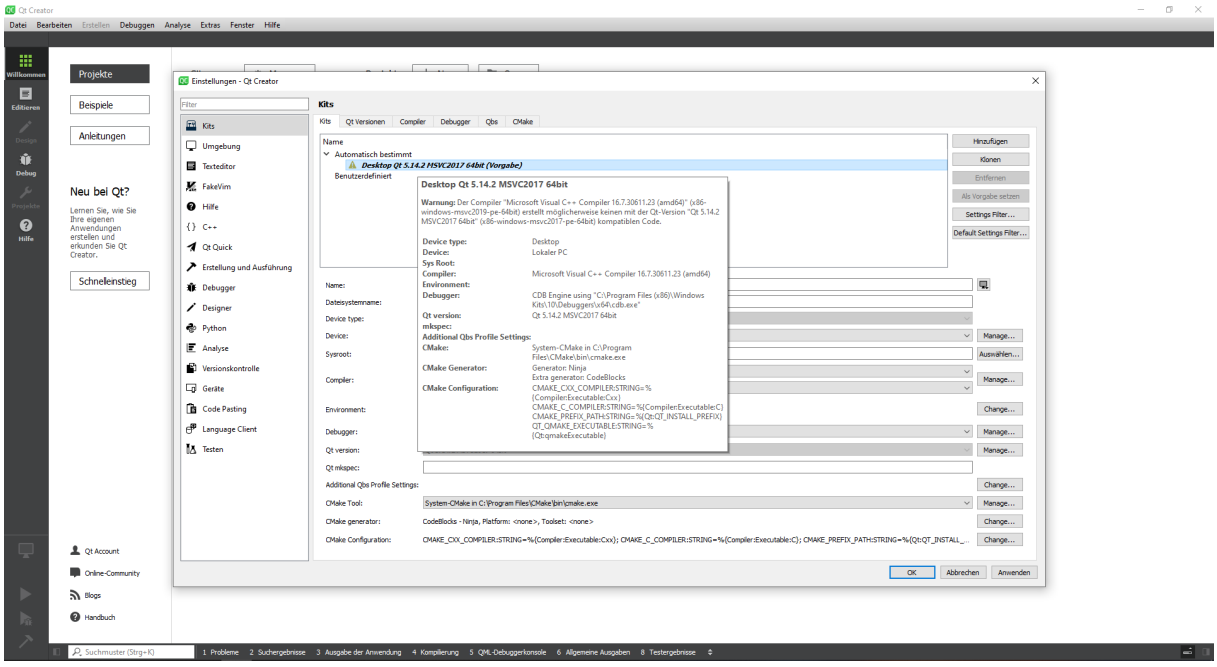
Sie können nun die Beispiel-Frameworks und Übungsaufgaben bearbeiten/starten, indem Sie die .pro-Datei mit dem QtCreator öffnen oder mittels CMake für eine Entwicklungsumgebung Ihrer Wahl eine Projektmappe erstellen (siehe Kapitel „CMake Projektdateien generieren“).

QtCreator

Neben Visual Studio, Xcode etc. liefert Qt seine eigene IDE, den QtCreator. Die bereitgestellten Projekte können durch öffnen der „.pro“-Datei geöffnet werden.

Schauen Sie im QtCreator unter „Extras“ > „Einstellungen“ > „Erstellung und Ausführung“, dass unter den Reitern „Kits“, „Qt Versionen“, „Compiler“, „Debugger“ und „CMake“ alles für Ihre Konfiguration richtig eingestellt ist. Die Listen dürfen nicht leer sein und Ihre benutzte Konfiguration darf kein Warn- oder Stopschild anzeigen. Wenn Sie den Mauszeiger über einen fehlerhaften Eintrag ruhen lasst, wird ein Popup angezeigt, das nähere Informationen enthält. **Ausnahme ist, dass sich der QtCreator beschwert, bei dem 2017 Kit den Compiler von Visual Studio 2019 zu nehmen. Es sollte trotzdem funktionieren!**





Linux (apt)

Warnung: Linux Mint <= 19.2 scheint auf diesem Weg das Modul Qt5OpenGL nicht zu installieren. Bauen Sie Qt5 nach der Anleitung im nächsten Kapitel einfach selber.

Öffnet Sie ein Terminal-Fenster und installieren Sie die Pakete:

```
sudo apt install git clang cmake qt5-default qtcreator
```

Linux (Source)

Hinweis: Für eine vollständige Installation werden zum Kompilieren mehr als 30GB Festplatten-speicher benötigt und der Vorgang kann einen ganzen Tag dauern.

Adaptiert von: https://wiki.qt.io/Building_Qt_5_from_Git#Getting_the_source_code

Öffnet Sie ein Terminal-Fenster und geben Sie folgende Befehle ein:

```
sudo apt install build-essential git clang cmake qtcreator

cd ~
git clone git://code.qt.io/qt/qt5.git -b 5.14.2
cd qt5
perl init-repository

./configure -developer-build -opensource -nomake examples -nomake tests \
-confirm-license -prefix /opt/Qt5.14.2 \
sudo make -j4 && make install
```

macOS (Homebrew)

Installieren Sie Xcode (https://itunes.apple.com/de/app/xcode/id497799835). Starten Sie Xcode einmal und akzeptieren die Lizenzbedingungen. Danach kann Xcode wieder geschlossen werden. Öffnen Sie im Browser die Seite https://brew.sh und kopieren den Installationsbefehl ins Terminal:

```
/bin/bash -c "$(curl -fsSL https://raw.githubusercontent.com/Homebrew/install/master/install.sh)"
```

Nach erfolgreicher Installation kann das standardmäßig eingeschaltete Google Analytics abgeschaltet, sowie die Pakete qt5, qt-creator und CMake im Terminal installiert werden:

```
brew analytics off
brew install cmake qt5
brew cask install qt-creator
```

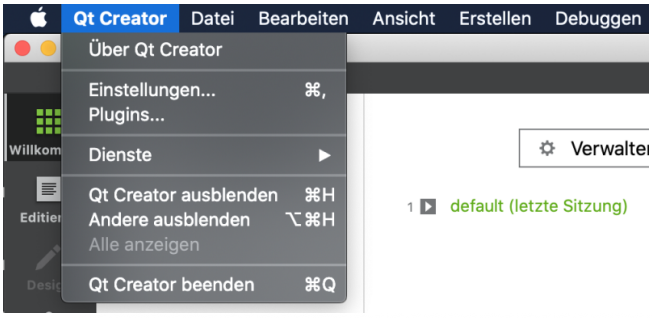
Bei der Homebrew Installation werden manchmal Compiler-Pfade abgeändert. Setzen Sie den Xcode-Compiler mit folgendem Befehl wieder zurück:

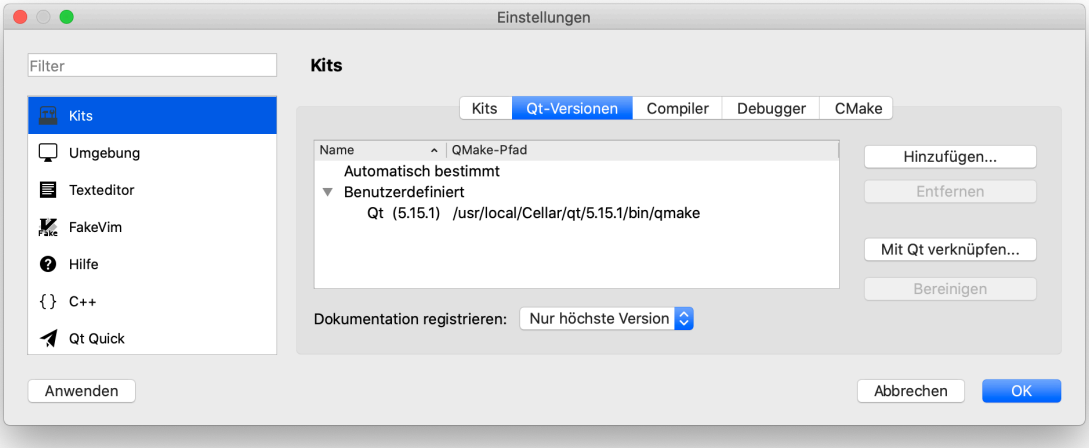
```
sudo xcode-select -r
```

Öffnen Sie im QtCreator „Qt Creator“ > „Einstellungen…“ (CMD + ,) und fügen folgenden Pfad bei „Qt-Versionen“ hinzu, wobei **YOUR_VERSION** mit der Qt5 Versionsnummer ersetzt werden muss:

```
/usr/local/Cellar/qt/YOUR_VERSION/bin/qmake
```

Warnung: Nach `brew update` kann eine neuere Qt5-Version installiert werden. Der beschriebene Qt-Pfad muss dann per Hand angepasst werden!

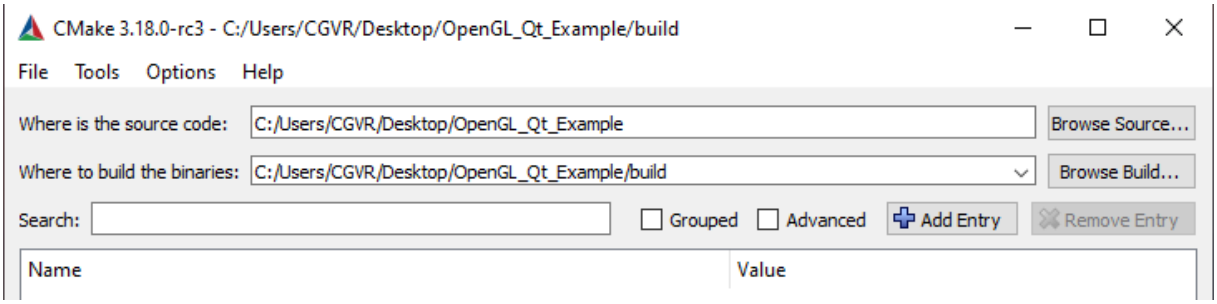




CMake Projektdateien generieren (Win/Linux/Mac)

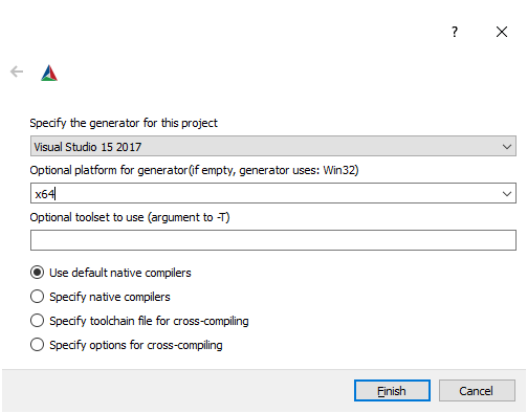
CMake kann für diverse Entwicklungsumgebungen, z.B. Visual Studio, Xcode etc., eine Projektmappe generieren, mit der dann entwickelt werden kann.

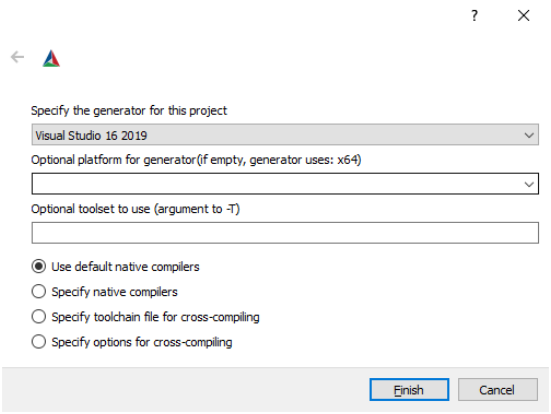
Starten Sie die CMake-gui. Stellen Sie bei der ersten Pfad-Auswahl ein, wo sich Ihr Projekt und damit die „CMakeLists.txt“ befindet. Stellen Sie bei der zweiten Pfad-Auswahl ein, wo die kompilierten Dateien abgelegt werden sollen. Es reicht, ein „/build“ anzuhängen, damit im Unterverzeichnis „build“ gebaut wird.



Für macOS klicken Sie auf den Knopf „Add Entry“, wählen „CMAKE_PREFIX_PATH“, Type: PATH und Value: „/usr/local/Cellar/qt/5.12.1/lib/cmake/Qt5“ ein (Version ggf. anpassen).

Klicken Sie auf den Knopf „**Configue**“ („Alt“ + „C“), wählen einmalig den den Projekttyp, z.B. „Visual Studio 16 2019“, „Visual Studio 15 2017“, „Xcode“ oder unter Linux „Makefile“. Bei Visual Studio 2017 muss bei „platform" zusätzlich **x64** ausgewählt werden.





Klicken Sie auf den Knopf „**Generate“** („Alt“ + „G“), damit die Projektmappe/Makefile etc. im Build-Ordner generiert wird.

Öffnen Sie mit „**Open Project**“ („Alt“ + „O“) das Projekt oder bauen es über den Weg Ihrer Wahl, z.B. mit dem make-Befehl. Im Build-Ordner kann die *.sln (Visual Studio Projektdatei) auf per Hand geöffnet werden.

In Visual Studio sollte darauf geachtet werden, dass das richtige Projekt als Startprojekt festgelegt ist. Man erkennt es daran, dass es in der Übersicht fettgedruckt steht. Häufig ist **ALL_BUILD** vorausgewählt. Mittels Rechtsklick auf das gewünschte Projekt und „Als Startprojekt festlegen“ (ziemlich mittig und mit Zahnrad-Symbol) lässt sich dies ändern. Nun kann oben über Run (grünes Dreieck) das Kompilieren/Ausführen des Programmes gestartet werden.