WzSDK 2014.3

Installation Guide

WzSDK Installation with CMake

Weinzierl Engineering GmbH

Abstract

The WzSDK libraries are cross platform and can be compiled using the Visual Studio MSVC compiler or the GNU C++ compiler (among others). The build system uses CMake. This document is designed to get you to build the WzSDK in the shortest possible time without discussing the background of the build system in detail. For more information about CMake visit the CMake website and read the excellent OGRE C++ wiki on building with CMake, which you can find *here*[1].

1. Introduction 1
2. CMake 1
3. Windows 2
4. Linux 6
5. Important Preprocessor Defines 7

A. Revision History **8**

1. Introduction

Weinzierl Engineering has several Software Development Kits available that use CMake as the build system. The instructions outlined in this document are provided as a guide to getting started with CMake.

Please Note

the SDKs vary slightly in their dependencies on additional libraries, such as the BOOST C++ headers. Please check the your SDK documentation to determine the SDK dependencies.

Special thanks too to the OGRE wiki for providing the basis of this document, which can be found *here*[2].

2. CMake

CMake is a cross-platform build system - or perhaps more accurately a build configurator. It is a program which, from a set of CMake scripts, creates a native build system for your platform that allows you to build your SDK. The build process is configurable via CMake. The SDK provides several options which you can use to customise your build.

[1] http://www.ogre3d.org/wiki/index.php/Building_With_CMake
[2] http://www.ogre3d.org/wiki/index.php/CMake_Quick_Start_Guide

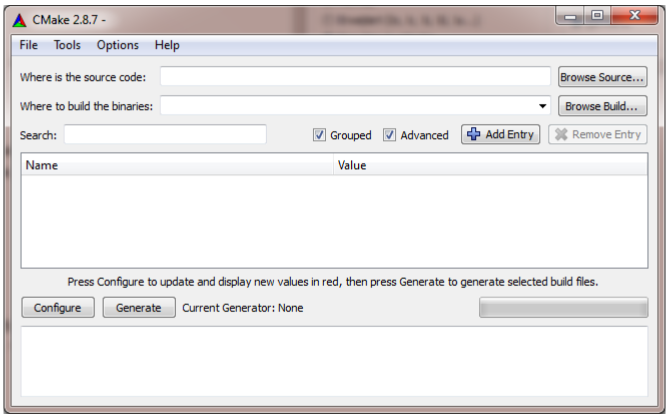
To build the SDK you need to download and install (if you haven't already) the CMake build system from *here*[3]

3. Windows

Download CMake. You want the 'Win32 installer' release in the binary distribution section.

Run the CMake installer, install wherever you like.

Launch CMake via Start > Program Files > CMake 2.8 > CMake.



Where is the source code:

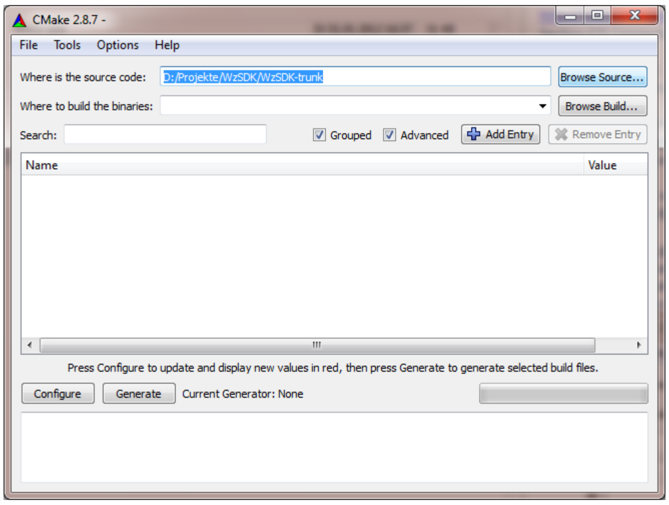
Where to build the binaries:

☑ Grouped ☑ Advanced

Press Configure to update and display new values in red, then press Generate to generate selected build files.

Current Generator: None

In the "Where is the source code" box, type or browse to the root folder of the SDK.



Where is the source code: D:/Projekte/WzSDK/WzSDK-trunk

Where to build the binaries:

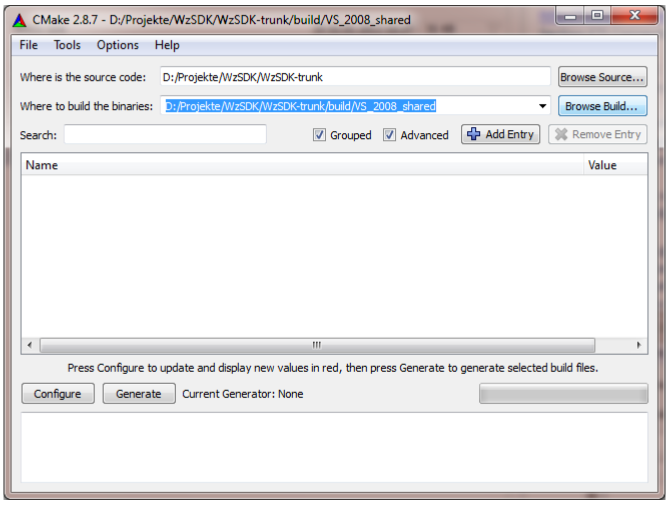
☑ Grouped ☑ Advanced

Press Configure to update and display new values in red, then press Generate to generate selected build files.

Current Generator: None

In the "Where to build the binaries" box, type or browse to any folder you like - this will be where the build output will go (libraries, headers, and sample exes). The folder does not have to exist yet.

[3] http://www.cmake.org/cmake/resources/software.html



CMake 2.8.7 - D:/Projekte/WzSDK/WzSDK-trunk/build/VS_2008_shared

Where is the source code: D:/Projekte/WzSDK/WzSDK-trunk

Where to build the binaries: D:/Projekte/WzSDK/WzSDK-trunk/build/VS_2008_shared

Grouped ✓ Advanced ✓

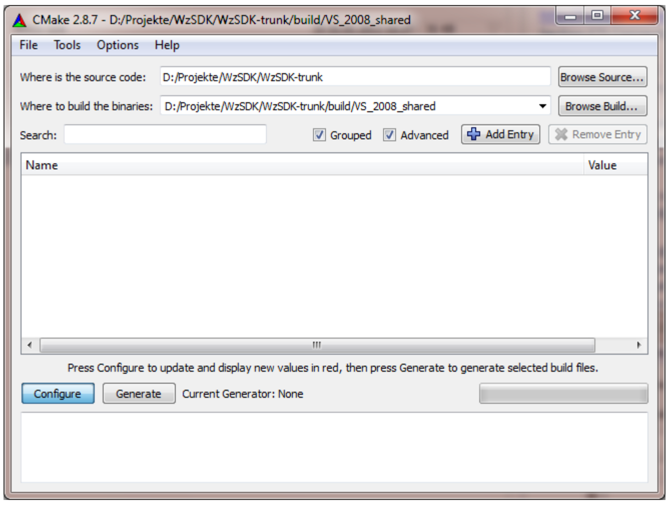
Press Configure to update and display new values in red, then press Generate to generate selected build files.

Current Generator: None

Note

You can run this process more than once with different output folders if you want, i.e. for static / shared / 32-bit / 64-bit builds etc. We typically use "build" and additional sub-folders for each configuration type.

Hit the 'Configure' button near the bottom of the screen.



CMake 2.8.7 - D:/Projekte/WzSDK/WzSDK-trunk/build/VS_2008_shared

Where is the source code: D:/Projekte/WzSDK/WzSDK-trunk

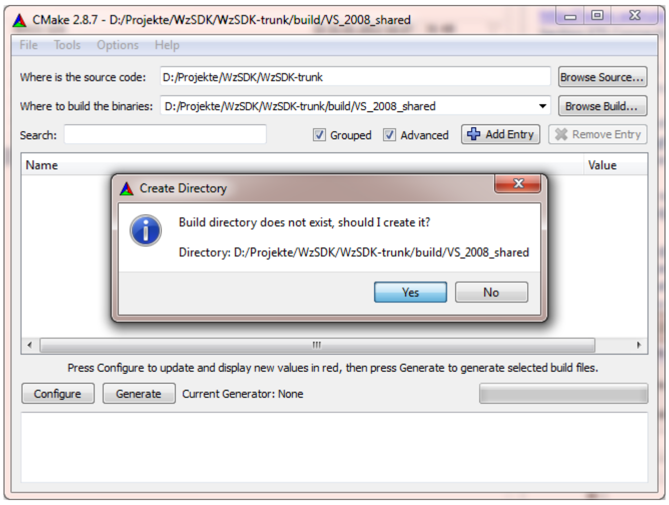
Where to build the binaries: D:/Projekte/WzSDK/WzSDK-trunk/build/VS_2008_shared

Grouped ✓ Advanced ✓

Press Configure to update and display new values in red, then press Generate to generate selected build files.

Current Generator: None

Answer 'Ok' when asked if you want to create the build directory.


CMake 2.8.7 - D:/Projekte/WzSDK/WzSDK-trunk/build/VS_2008_shared

Where is the source code: D:/Projekte/WzSDK/WzSDK-trunk

Where to build the binaries: D:/Projekte/WzSDK/WzSDK-trunk/build/VS_2008_shared

Search: [✓] Grouped [✓] Advanced

Create Directory

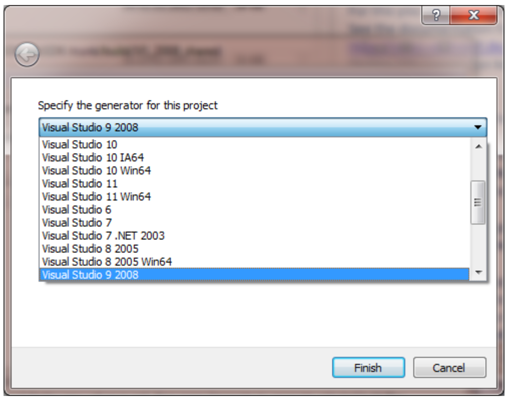
Build directory does not exist, should I create it?

Directory: D:/Projekte/WzSDK/WzSDK-trunk/build/VS_2008_shared

Press Configure to update and display new values in red, then press Generate to generate selected build files.

Current Generator: None

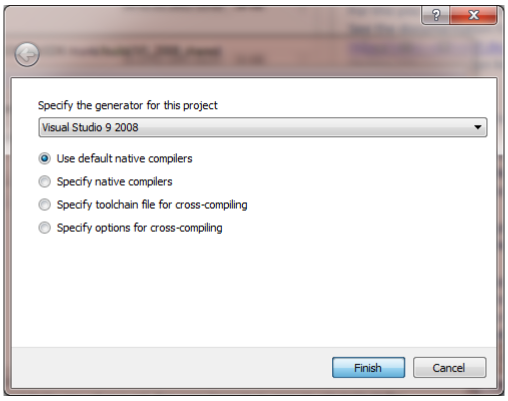
Pick your target compiler, e.g. "Visual Studio 9 2008".


Specify the generator for this project

Visual Studio 9 2008

Visual Studio 10
Visual Studio 10 IA64
Visual Studio 10 Win64
Visual Studio 11
Visual Studio 11 Win64
Visual Studio 6
Visual Studio 7
Visual Studio 7 .NET 2003
Visual Studio 8 2005
Visual Studio 8 2005 Win64
Visual Studio 9 2008

Close the dialog with finish.

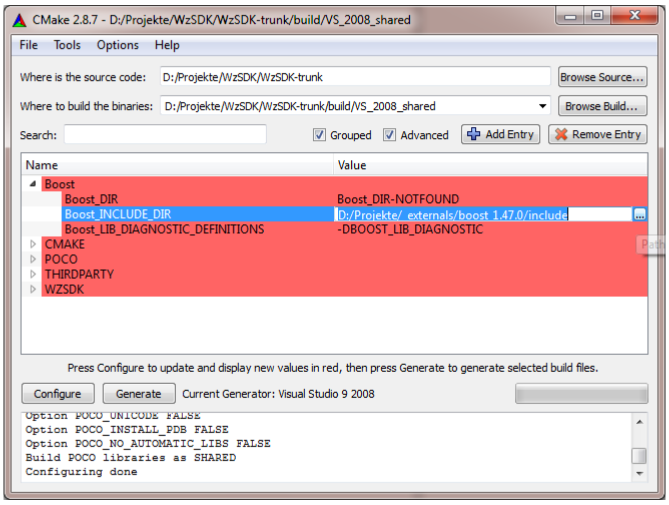

Specify the generator for this project

Visual Studio 9 2008

● Use default native compilers
○ Specify native compilers
○ Specify toolchain file for cross-compiling
○ Specify options for cross-compiling

Wait for the configure process to finish.

If the SDK requires the C++ BOOST header files then do the following: Click on the Check Boxes next to the Search line labeled "Grouped View" and "Advanced" or on the Combo Box and select "Grouped View" (it depends on your cmake-gui version). Select Boost and enter the Boost header files location for the entry Boost_INCLUDE_DIR. The windows SDK may contain a Boost header distribution and if you are not using your own boost headers, simply select WzSDK/boost (where WzSDK is the root SDK directory).



CMake 2.8.7 - D:/Projekte/WzSDK/WzSDK-trunk/build/VS_2008_shared

Where is the source code: D:/Projekte/WzSDK/WzSDK-trunk

Where to build the binaries: D:/Projekte/WzSDK/WzSDK-trunk/build/VS_2008_shared

Search: ☑ Grouped ☑ Advanced

Name	Value
Boost	
Boost_DIR	Boost_DIR-NOTFOUND
Boost_INCLUDE_DIR	D:/Projekte/_externals/boost_1.47.0/include
Boost_LIB_DIAGNOSTIC_DEFINITIONS	-DBOOST_LIB_DIAGNOSTIC
CMAKE	
POCO	
THIRDPARTY	
WZSDK	

Press Configure to update and display new values in red, then press Generate to generate selected build files.

Current Generator: Visual Studio 9 2008

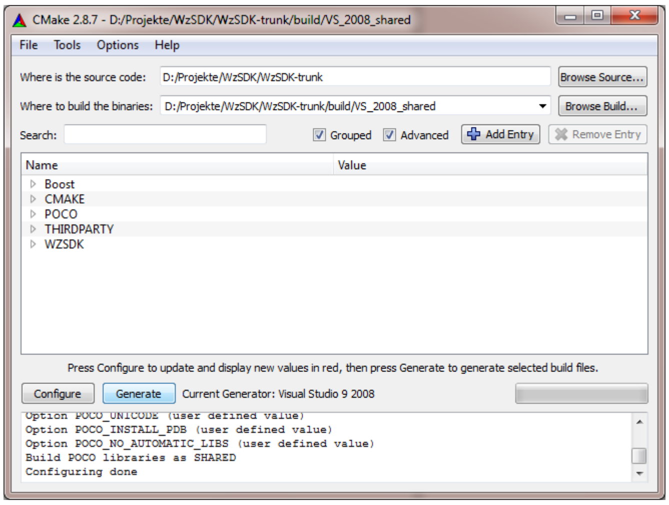
```
Option POCO_UNICODE FALSE
Option POCO_INSTALL_PDB FALSE
Option POCO_NO_AUTOMATIC_LIBS FALSE
Build POCO libraries as SHARED
Configuring done
```

Note

An alternative is to set the environment variable BOOST_ROOT to the root of the boost folder.
Open the Control Panel and navigate to System an Security -> System -> Advanced system settings
In the System Properties window select Advanced tab and click on Enviroment Variables... .
In the Enviroment Variables window you click on the New button.
Type as variable name BOOST_ROOT and the path to the Boost root for variable value. e.g. D:\Projekte_externals\boost_1.47.0

The screen will now have a bunch of configuration options on it, which will be red (this is to indicate this is the first time you've seen them). You can see the potential for customising your build here, but for now just click the 'Configure' button again.

The values will turn grey, and the 'Generate' button at the bottom will now be enabled. Click 'Generate'.



CMake 2.8.7 - D:/Projekte/WzSDK/WzSDK-trunk/build/VS_2008_shared

Where is the source code: D:/Projekte/WzSDK/WzSDK-trunk

Where to build the binaries: D:/Projekte/WzSDK/WzSDK-trunk/build/VS_2008_shared

Search: ☑ Grouped ☑ Advanced

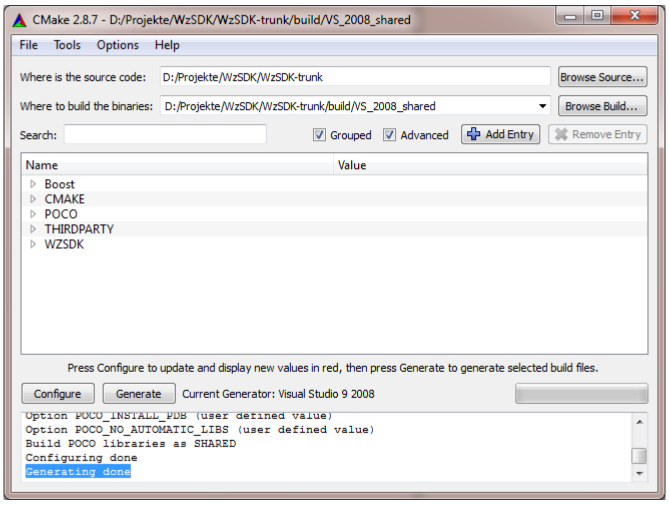
Name	Value
Boost	
CMAKE	
POCO	
THIRDPARTY	
WZSDK	

Press Configure to update and display new values in red, then press Generate to generate selected build files.

Current Generator: Visual Studio 9 2008

```
Option POCO_UNICODE (user defined value)
Option POCO_INSTALL_PDB (user defined value)
Option POCO_NO_AUTOMATIC_LIBS (user defined value)
Build POCO libraries as SHARED
Configuring done
```

Build files will now be generated in the location you picked.



CMake 2.8.7 - D:/Projekte/WzSDK/WzSDK-trunk/build/VS_2008_shared

Where is the source code: D:/Projekte/WzSDK/WzSDK-trunk

Where to build the binaries: D:/Projekte/WzSDK/WzSDK-trunk/build/VS_2008_shared

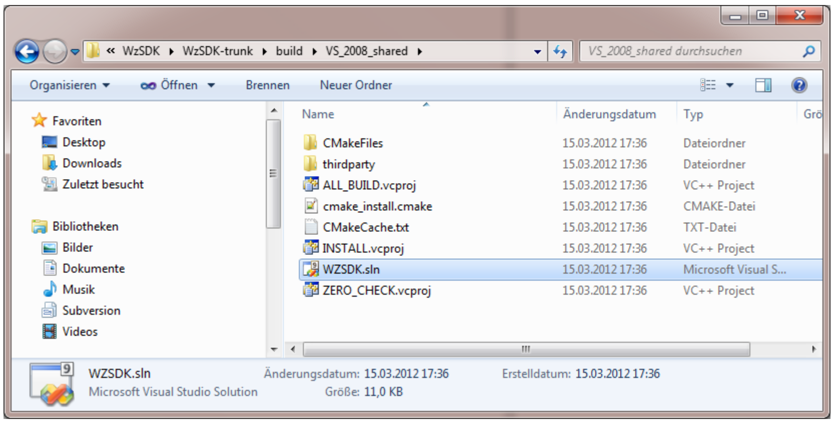
Name	Value
Boost	
CMAKE	
POCO	
THIRDPARTY	
WZSDK	

Press Configure to update and display new values in red, then press Generate to generate selected build files.

Current Generator: Visual Studio 9 2008

```
Option POCO_INSTALL_PDB (user defined value)
Option POCO_NO_AUTOMATIC_LIBS (user defined value)
Build POCO libraries as SHARED
Configuring done
Generating done
```

You now have a set of Visual Studio (or whatever generator you selected) project files. Just browse to the build directory and open the main project file (e.g. WzSDK.sln). To build select the special target called 'ALL_BUILD' which is a quick way to build all targets for a specific configuration. All build output is now stored in a subfolder of this build folder (i.e. out of source build) - so you can create multiple builds from one source tree.



Name	Änderungsdatum	Typ
CMakeFiles	15.03.2012 17:36	Dateiordner
thirdparty	15.03.2012 17:36	Dateiordner
ALL_BUILD.vcproj	15.03.2012 17:36	VC++ Project
cmake_install.cmake	15.03.2012 17:36	CMAKE-Datei
CMakeCache.txt	15.03.2012 17:36	TXT-Datei
INSTALL.vcproj	15.03.2012 17:36	VC++ Project
WZSDK.sln	15.03.2012 17:36	Microsoft Visual S...
ZERO_CHECK.vcproj	15.03.2012 17:36	VC++ Project

WZSDK.sln Änderungsdatum: 15.03.2012 17:36 Erstelldatum: 15.03.2012 17:36
Microsoft Visual Studio Solution Größe: 11,0 KB

Note

Once you have built the SDK you can install the header and library files to somewhere convenient, for example the root directory of the SDK. To do this: modify the CMAKE_INSTALL_PREFIX option with the cmake-gui editor to Path/To/WzSDK and run the special INSTALL target (in Visual Studio this is the INSTALL project). By default the prefix is set to the path: Where to build the binaries / sdk. This has the advantage of including all the headers together into one location, so you don't have to reference each source folder individually in your project. When working with Visual Studio we typically add a property file (i.e. .vsprops) with a user macro to define the WZSDK root folder. In our projects we then refer to the WZSDK/include and WZSDK/lib files...

4. Linux

Install CMake via your distribution's package repository. For example, on Ubuntu type:

```
sudo apt-get install cmake
```

If your distribution does not provide an official CMake package, go to the CMake website and either download a binary package or download and compile CMake from source.

If your SDK requires boost, ensure you have the boost headers installed (if they are not already included in your SDK).

Create a build directory somewhere on your filesystem - this will be where the build output will go.

In a console, cd to the build directory and call cmake with the path to the source dir (i. e. where you extracted the WzSDK source files):

```
cmake -G "Unix Makefiles" /path/to/WzSDK
```

Compile WzSDK. Type:

```
make
```

Don't forget that make provides the -j option which you can use to spawn multiple make processes to speed up compiling on a multi-core PC. For example, this works well on a dual-core:

```
make -j3
```

Install WzSDK. Type as root:

```
make install
```

This will install the WzSDK libraries to /usr/local (You can change the inslaa location by setting CMAKE_INSTALL_PREFIX)

Note

Boost should be automatically found, however if in a non-standard location you can configure the boost headers location using the following command:

```
BOOST_ROOT=/path/to/boost/root cmake -G "Unix Makefiles" /path/to/WzSDK
```

Or alternatively you can use the cmake-gui application to edit the location of the boost headers.

CMake also supports various IDE's under linux, such as KDevelop and Eclipse. Call cmake from the command line to get the list of supported targets (i.e. in addition to "Unix Makefiles").

5. Important Preprocessor Defines

The following defines are automatically managed by the CMake system:

Table 1. Preprocessor Defines

Define	Description
WZ_STATIC	Indicates whether the kdrive libraries are statically or dynamically compiled. You need to set this in your application if you are using static libraries.
POCO_STATIC	Indicates whether the poco libraries are statically or dynamically compiled. This will typically be

Define	Description
	the same as WZ_STATIC. You need to set this in your application if you are using static libraries.

A. Revision History

Revision 0-0 **Thu May 5 2011** **Jason Richards**

Initial creation