INVRS

interactive networked virtual reality system

–

Installation Instructions

Roland Landertshamer and Christoph Anthes

June 10, 2009

Abstract

The *inVRs* framework was created to ease the design and the development of Networked Virtual Environments. This document describes how to install *inVRs*.
This is just a draft version covering the basic installation on Linux, Windows and Mac systems.

Contents

Chapter 1

Introduction

Installation can often be cumbersome with free software packages. This document describes in detail how to install the *inVRs* framework on any supported operating system. The framework is available and has been successfully tested under Microsoft Windows, Mac OS X and Linux in general.

1.1 Prerequisites

Before *inVRs* can be installed the following open source packages have to be installed on your system:

- CMake version 2.4 or above
- OpenSG version 1.8
  Install OpenSG with glut support as well as graphics format support for gif, tif, jpg, png

1.2 Download

The *inVRs* sources are publicly available under an LGPL license and can be downloaded from:

- http://www.inVRs.org

1.3 Outline

The document is organized in the following chapters:

- Chapter 2 – Installation on Linux
  The installation under a standard Linux operating system is described in this chapter. Additionally information on how to get an *inVRs* Ecplise project running is explained.
- Chapter 3 – Installation on Microsoft Windows
  This chapter provides information on how to run *inVRs* under a Windows operating system. As an IDE Visual Studio 2003/2005 are supported.
- Chapter 4 – Installation on Mac OS X
  The framework has be tested under the latest Mac OS X (Leopard). It is described how to install it in this chapter.

Chapter 2

Installation on Linux

2.1 Linux: Installation via the Console

2.1.1 Step 1: Generate Makefiles using CMake

In the first step CMake is used for the generation of Makefiles which are used in the following for building the *inVRs* libraries.
To be able to create these Makefiles CMake tries to automatically find several paths which are required by *inVRs* (like the path to the OpenSG libraries). In a default installation setup CMake should be able to find all paths by itself. But in some cases some paths can not be found by CMake and it will abort and report an error message. In such a case proceed to Step 1a (Section 2.1.2) where the manual setting of paths is described.
But before try if CMake can do the job without help: Enter the `build` subdirectory of the *inVRs* path and call the *cmake* command:

```
cd <inVRs_dir>/build
cmake ../
```

If the *cmake* command finished without errors you can skip Step 1a (Section 2.1.2) and proceed to Step 2 (Section 2.1.3).

2.1.2 Step 1a (Optional): Set paths for CMake

This section describes how to define paths which should be used by CMake. The instructions here will show how to pass the path to the OpenSG configuration binary file `osg-config`, but can be applied for any other path as well.
To pass the path to CMake two possible options are available:

1. Pass path to CMake via command line

The first option is to pass the required path directly to the *cmake* command. For example let's assume that CMake was not able to identify the path where the OpenSG configuration file `osg-config` is installed. In this case CMake explains about not finding the path **OPENSG_BIN_DIR**.
This path can be passed to CMake by attaching it to the *cmake* command:

```
cmake ../ -DOPENSG_BIN_DIR=/usr/local/bin
```

The drawback of this method is that the path has to be passed every time the CMake cache is cleaned.

2. Enter path in user.cmake

To set the missing path you can also enter it in the file `user.cmake` which can be found in the *inVRs* main directory. This file is read by CMake at the beginning and allows to define paths and other user-specific CMake commands.
For entering the path to the OpenSG configuration file just open the `user.cmake` file in the editor of your choice and enter the following line:

```
...
set (OPENSG_BIN_DIR /usr/local/bin)
...
```

Next you can execute the *cmake* command again as described in Step 1.

2.1.3 Step 2: Build inVRs

After CMake successfully created your build structure you can start building *inVRs*. First enter the `build` subdirectory of *inVRs* and start building *inVRs* by calling the make command:

```
cd <inVRs_dir>/build
make
```

This will build the *inVRs* libraries in the `build` subdirectory. In order to simplify the usage of the *inVRs* libraries from external projects they have to be installed into a common library directory. This is the same for the *inVRs* header files which are needed for developing *inVRs* applications. Therefore the **install** make target is available which installs these files into the subdirectories `include` and `lib` of the *inVRs* path:

```
cd <inVRs_dir>/build
make install
```

This will install the *inVRs* libraries and headers into the *inVRs* directory in `<inVRs_dir>/lib` and `<inVRs_dir>/include`.

2.2 Linux: Installation via Eclipse

Eclipse is an open source IDE which is freely available under:

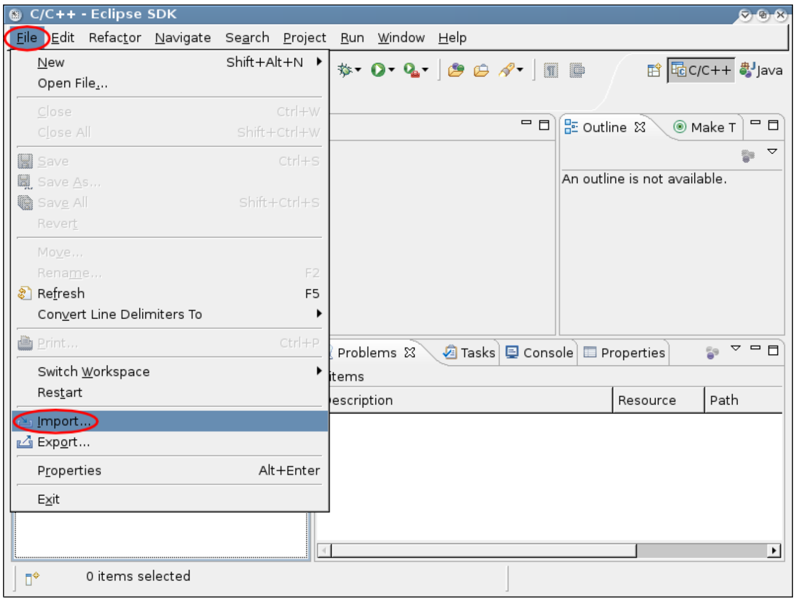
- `http://www.eclipse.org/`

In order to use Eclipse you have to have at least version 3.4 installed on your system. Three basic steps are required to run *inVRs* in combination with the Eclipse framework.

2.2.1 Step 1: Import the *inVRs* project in Eclipse

At first the *inVRs* eclipse project has to be imported into the IDE:

First click on **File** → **Import**:

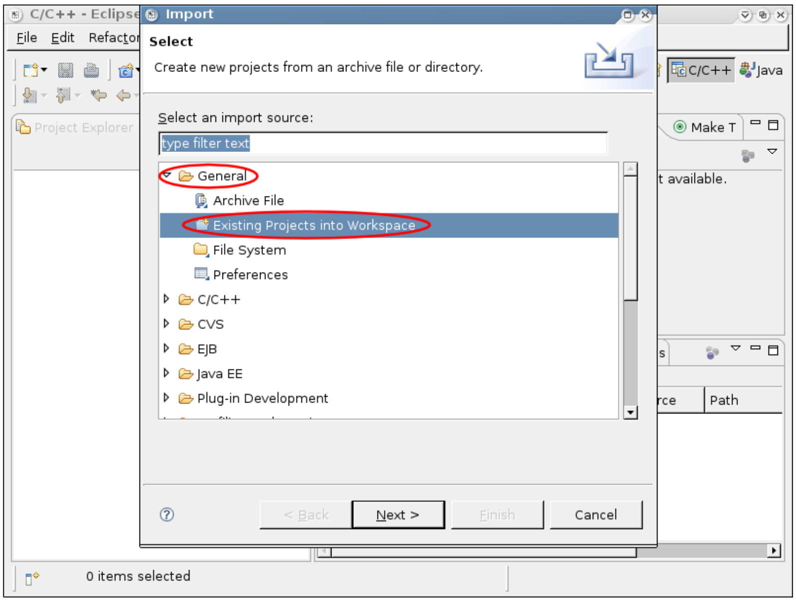


An outline is not available.

0 items selected

Figure 2.1: File → Import

Choose **General** → **Existing Project into Workspace**:



Select

Create new projects from an archive file or directory.

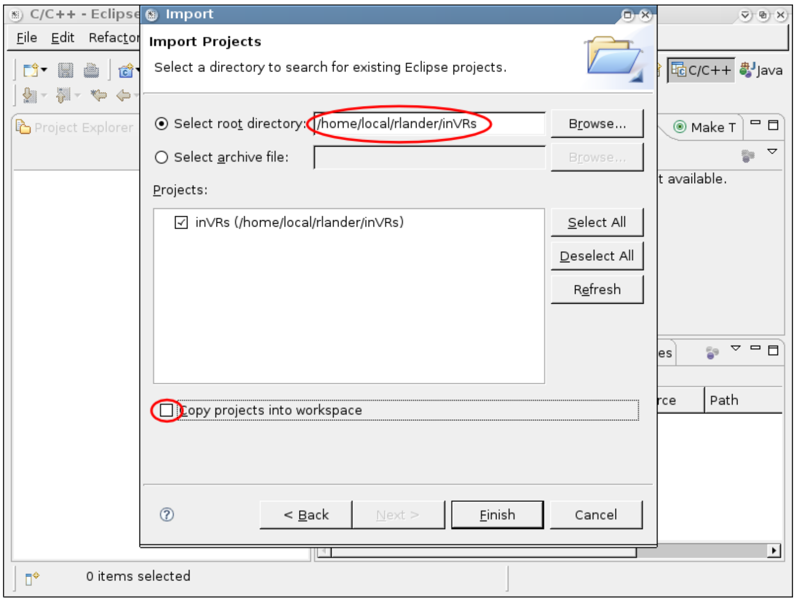
Select an import source:

t available.

0 items selected

Figure 2.2: General → Existing Project into Workspace

Then **Browse** to the location where you installed the *inVRs* sources. Be sure to deactivate the selection box **Copy Projects into Workspace**:



Import Projects

Select a directory to search for existing Eclipse projects.

● Select root directory: /home/local/rlander/inVRs

○ Select archive file:

Projects:

☑ inVRs (/home/local/rlander/inVRs)

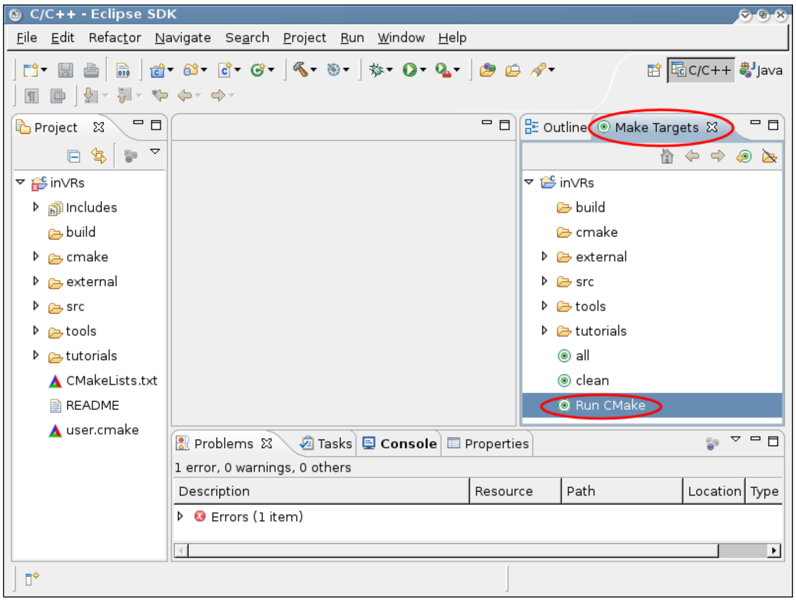
□ Copy projects into workspace

0 items selected

Figure 2.3: Copy Projects into Workspace

2.2.2 Step 2: Generate Makefiles using CMake

Before *inVRs* can be built the Makefiles which are used for building have to be created. This is done by executing the *cmake* command. To simplify the usage the *inVRs* eclipse project provides a build target for running CMake. This target can be built by switching to the **Make Targets** tab and building the **Run CMake** target:

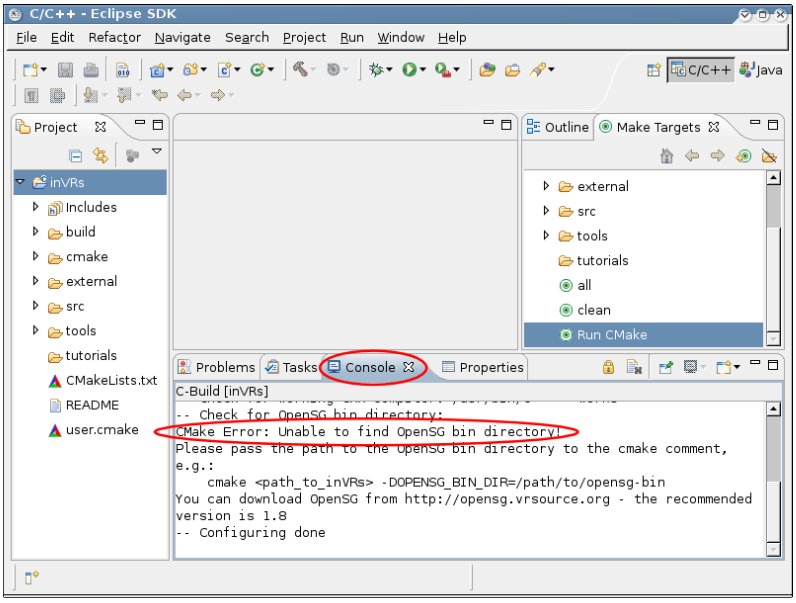


1 error, 0 warnings, 0 others

Errors (1 item)

Figure 2.4: Run CMake

After this command was executed open the **Console** tab on the bottom to view if CMake worked properly or if CMake is unable to find some packages. The following picture shows an example output in case that the OpenSG binary path could not be found:



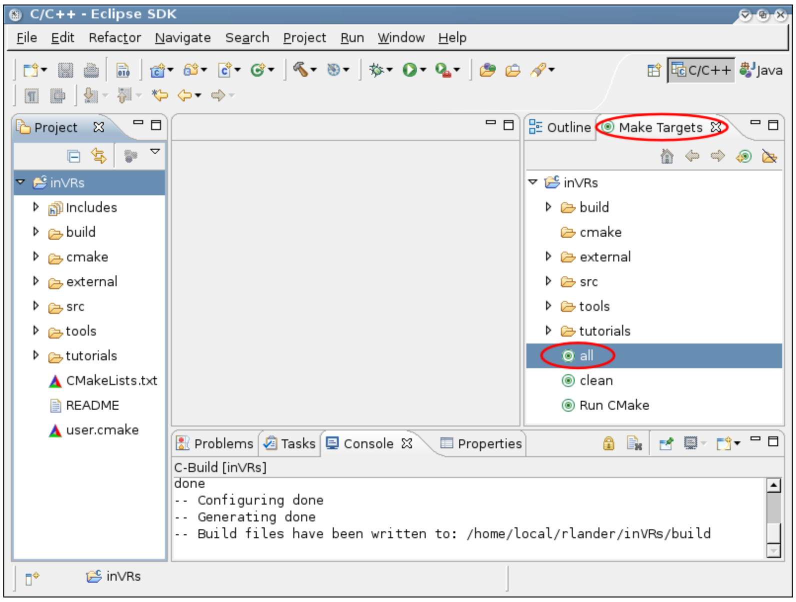
```
C-Build [inVRs]
-- Check for OpenSG bin directory:
CMake Error: Unable to find OpenSG bin directory!
Please pass the path to the OpenSG bin directory to the cmake comment,
e.g.:
    cmake <path_to_inVRs> -DOPENSG_BIN_DIR=/path/to/opensg-bin
You can download OpenSG from http://opensg.vrsource.org - the recommended
version is 1.8
-- Configuring done
```

Figure 2.5: Check for CMake errors

If there are some problems with finding paths you can modify the `user.cmake` file in the *inVRs* main directory. For more detailed instructions on passing paths to CMake see Section 2.1.2.

2.2.3 Step 3: Build inVRs

After CMake created the Makefiles successfully the *inVRs* project can be built:



```
C-Build [inVRs]
done
-- Configuring done
-- Generating done
-- Build files have been written to: /home/local/rlander/inVRs/build
```

Figure 2.6: Run Make All

Therefore switch to the **Make Targets** tab again and execute the **all** target (see Figure 2.6). This will automatically build all *inVRs* libraries and also install the include files and built libraries into the `include` and `lib` subdirectories of *inVRs*.

Chapter 3

Installation on Microsoft Windows XP

3.1 Requirements

3.1.1 CMake (version 2.4 or 2.6)

The CMake package is used for creating the Microsoft Visual Studio Project Files. It can be found at:

- `http://www.cmake.org`

3.1.2 OpenSG 1.8

For visualization inVRs uses the OpenSG scenegraph. This package can be downloaded from:

- `http://opensg.vrsource.org/trac`

You can either build OpenSG from source or download the prepackaged version. When downloading the prepackaged version take care to use the file which matches to the Visual Studio version you are using.

3.1.3 Microsoft Visual Studio 2003/2005

For building inVRs on Windows XP the Microsoft Visual Studio IDE is used. Besides Visual Studio you will also have to install the Microsoft Windows Platform SDK. Both packages can be downloaded from the Microsoft Homepage:

- `http://www.microsoft.com`

Additionally you will have to configure Visual Studio for using the Platform SDK. For instructions how to configure Visual Studio 2005 for Platform SDK compatibility see:

- `http://msdn.microsoft.com/en-us/library/ms235626(VS.80).aspx`

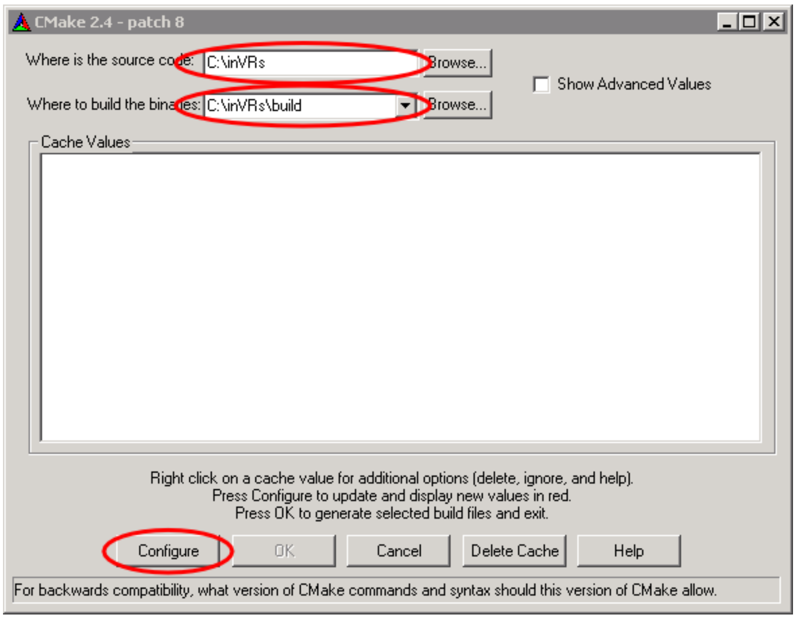
3.1.4 UNSUPPORTED: Microsoft Visual Studio 2008

In general inVRs should also be buildable using Microsoft Visual Studio 2008. The reason why this is not supported here is because there is no binary version of OpenSG 1.8 built with this version of Visual Studio at this time. Using an OpenSG build which was created with another Visual Studio version will most probably cause problems with the Microsoft Windows runtime libraries. Thus Microsoft Visual Studio 2008 is not officially supported.

3.2 Installation Instructions

3.2.1 Step 1: Create Visual Studio Project files with CMake

Open CMake and set the source path to the inVRs directory and the binary path to the `build` subdirectory of inVRs. Press Configure and select as target build type the Microsoft Visual Studio Version you are using. Next press OK. CMake will now try to create the Visual C++ Project files inside the `build` directory. To be able to create the project files CMake tries to automatically find



CMake 2.4 - patch 8

Where is the source code: C:\inVRs

Where to build the binaries: C:\inVRs\build

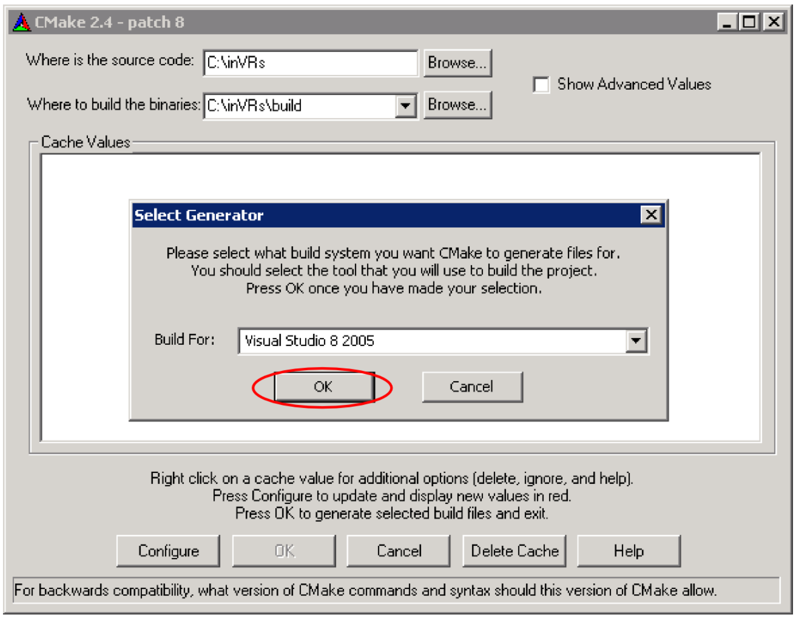
□ Show Advanced Values

Cache Values

Right click on a cache value for additional options (delete, ignore, and help).
Press Configure to update and display new values in red.
Press OK to generate selected build files and exit.

For backwards compatibility, what version of CMake commands and syntax should this version of CMake allow.

Figure 3.1: Open CMake and set inVRs paths



CMake 2.4 - patch 8

Where is the source code: C:\inVRs

Where to build the binaries: C:\inVRs\build

□ Show Advanced Values

Cache Values

Select Generator

Please select what build system you want CMake to generate files for.
You should select the tool that you will use to build the project.
Press OK once you have made your selection.

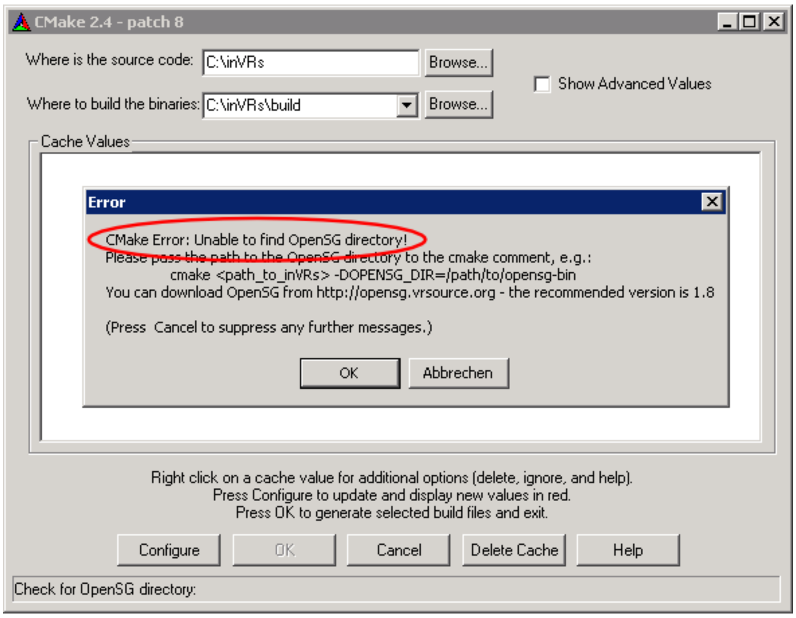
Build For: Visual Studio 8 2005

Right click on a cache value for additional options (delete, ignore, and help).
Press Configure to update and display new values in red.
Press OK to generate selected build files and exit.

For backwards compatibility, what version of CMake commands and syntax should this version of CMake allow.

Figure 3.2: Select target project type

several paths which are required by *inVRs* (like the path to the OpenSG libraries). In a default installation setup CMake should be able to find all paths by itself. But in some cases some paths can not be found by CMake and it will abort and report an error message. The following figure for example shows the error message which occurs if CMake has problems finding the OpenSG path:



CMake 2.4 - patch 8

Where is the source code: C:\inVRs

Where to build the binaries: C:\inVRs\build

□ Show Advanced Values

Cache Values

Error

CMake Error: Unable to find OpenSG directory!
Please pass the path to the OpenSG directory to the cmake comment, e.g.:
cmake <path_to_inVRs> -DOPENSG_DIR=/path/to/opensg-bin
You can download OpenSG from http://opensg.vrsource.org - the recommended version is 1.8

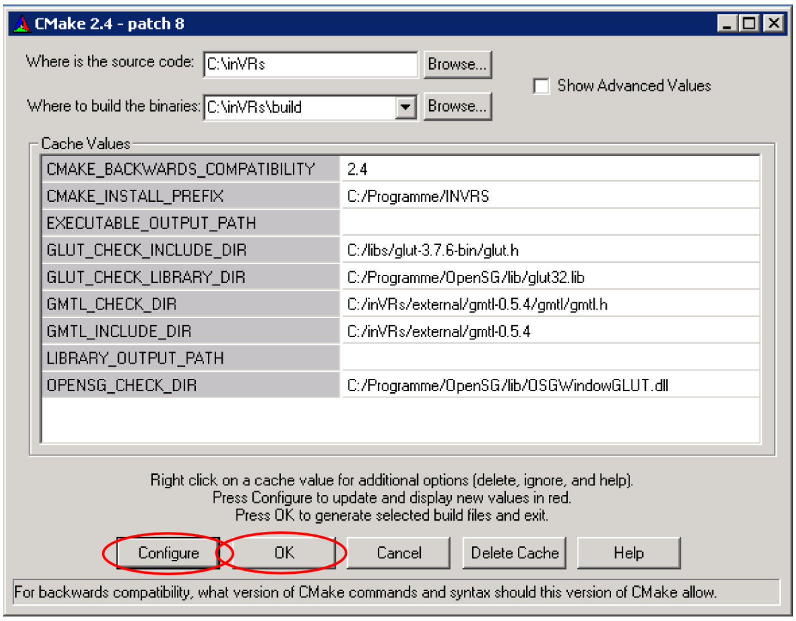
(Press Cancel to suppress any further messages.)

Right click on a cache value for additional options (delete, ignore, and help).
Press Configure to update and display new values in red.
Press OK to generate selected build files and exit.

Check for OpenSG directory:

Figure 3.3: CMake error message when OpenSG path is not found

In such a case proceed to Step 1a (Section 3.2.2) where the manual setting of paths is described. In case Configure finished without errors press Configure again and afterwards press OK (or Generate on newer CMake versions):



CMake 2.4 - patch 8

Where is the source code: C:\inVRs

Where to build the binaries: C:\inVRs\build

□ Show Advanced Values

Cache Values

CMAKE_BACKWARDS_COMPATIBILITY	2.4
CMAKE_INSTALL_PREFIX	C:/Programme/INVRS
EXECUTABLE_OUTPUT_PATH	
GLUT_CHECK_INCLUDE_DIR	C:/libs/glut-3.7.6-bin/glut.h
GLUT_CHECK_LIBRARY_DIR	C:/Programme/OpenSG/lib/glut32.lib
GMTL_CHECK_DIR	C:/inVRs/external/gmtl-0.5.4/gmtl/gmtl.h
GMTL_INCLUDE_DIR	C:/inVRs/external/gmtl-0.5.4
LIBRARY_OUTPUT_PATH	
OPENSG_CHECK_DIR	C:/Programme/OpenSG/lib/OSGWindowGLUT.dll

Right click on a cache value for additional options (delete, ignore, and help).
Press Configure to update and display new values in red.
Press OK to generate selected build files and exit.

For backwards compatibility, what version of CMake commands and syntax should this version of CMake allow.

Figure 3.4: Finalize configuration

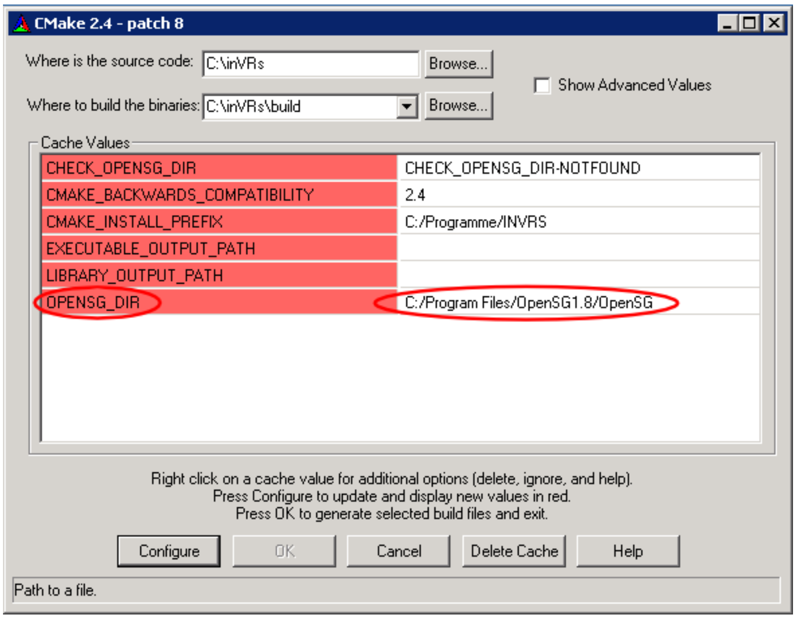
Afterwards you can proceed with Step 2 (Section 3.2.3).

3.2.2 Step 1a (Optional): Set paths for CMake

This section describes how to define paths which should be used by CMake. The instructions here will show how to pass the paths to the OpenSG header and library files, but can be applied for any other path as well.
To pass the path to CMake two possible options are available:

1. Enter the path in the CMake GUI

The first option is to enter the required path directly in the CMake GUI. For example let's assume that CMake was not able to identify the path where the OpenSG files can be found. In this case CMake explains about not finding the path **OPENSG_DIR**. This path can be passed to CMake by entering it in the correct line in the interface:



Where is the source code: C:\inVRs

Where to build the binaries: C:\inVRs\build

□ Show Advanced Values

Cache Values

CHECK_OPENSG_DIR	CHECK_OPENSG_DIR-NOTFOUND
CMAKE_BACKWARDS_COMPATIBILITY	2.4
CMAKE_INSTALL_PREFIX	C:/Programme/INVRS
EXECUTABLE_OUTPUT_PATH	
LIBRARY_OUTPUT_PATH	
OPENSG_DIR	C:/Program Files/OpenSG1.8/OpenSG

Right click on a cache value for additional options (delete, ignore, and help).
Press Configure to update and display new values in red.
Press OK to generate selected build files and exit.

Path to a file.

Figure 3.5: Enter correct path in GUI

The drawback of this method is that the path has to be entered every time the CMake cache is cleaned.

2. Enter path in user.cmake

To set the missing path you can also enter it in the file `user.cmake` which can be found in the *inVRs* main directory. This file is read by CMake at the beginning and allows to define paths and other user-specific CMake commands.
For entering the path to the OpenSG installation just open the `user.cmake` file in the editor of your choice and enter the following entry:

```
# FOR WINDOWS SYSTEMS:
# DEFINES OPENSG DIRECTORY
# By uncommenting the following line you can specify the path where OpenSG
# is installed.
# If this entry is not set cmake tries to find the path by itself.
set (OPENSG_DIR C:/Program\ Files/OpenSG)
```

NOTE: When setting the paths please ensure to use slashes instead of backslashes for separators and escape characters like spaces or brackets by putting a backslash before them. Otherwise CMake will have problems with the identification of the paths!

3.2.3 Step 2: Open inVRs Project in Visual Studio

After CMake successfully generated the Microsoft Visual Studio project files the *inVRs* libraries can be built. Therefore open Visual Studio and open the inVRs project from `<inVRs>/build/INVRS.SLN`. This will open a set of subprojects, each for a separate inVRs library:

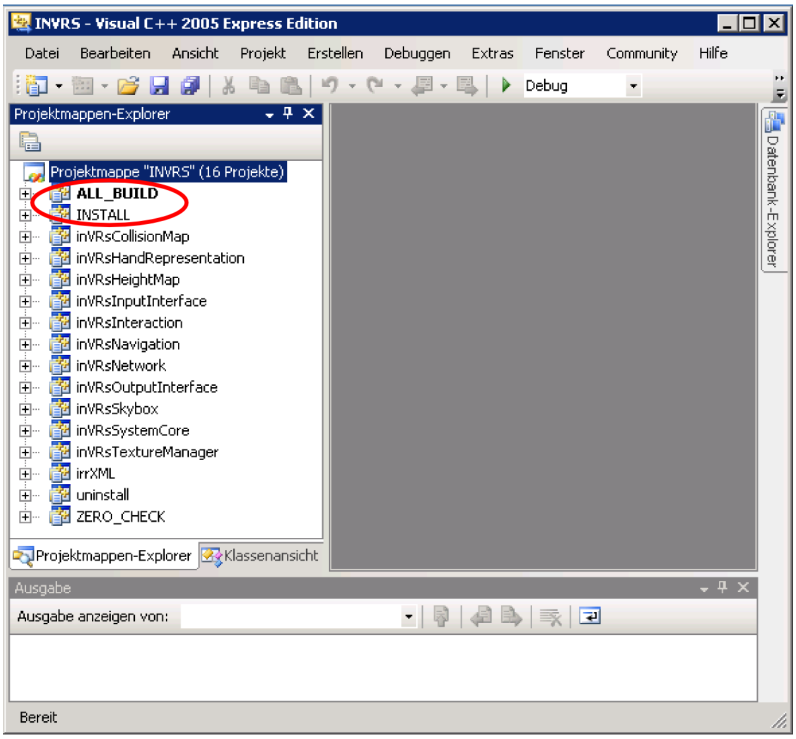


Figure 3.6: Build inVRs in Visual C++ 2005

To create all *inVRs* libraries build the target **ALL_BUILD**.
In order to use *inVRs* from separate applications you also have to build the target **INSTALL**. This will copy all built libraries into the folder `<inVRs>/lib` and all include files into `<inVRs>/include`.

Chapter 4

Installation on Mac OS X

4.1 Requirements

4.1.1 CMake (version 2.4 or 2.6)

The CMake package is used for creating the Makefiles which are needed for building inVRs. It can be found at:

- `http://www.cmake.org`

4.1.2 OpenSG 1.8

For visualization inVRs uses the OpenSG scenegraph. This package can be downloaded from:

- `http://opensg.vrsource.org/trac`

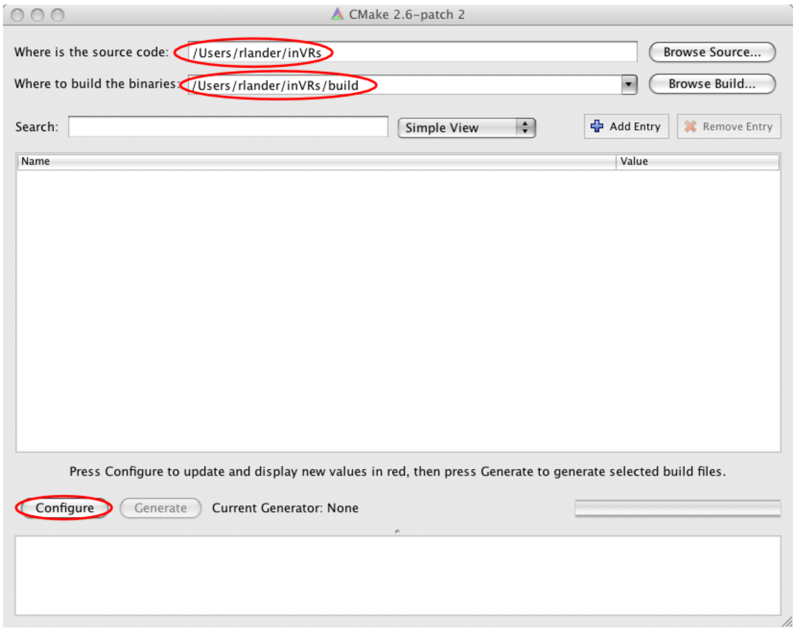
Instructions for building OpenSG 1.8 on Mac OS X can be found at:

- `http://opensg.vrsource.org/trac/wiki/BuildMacOSX`

4.2 Installation Instructions

4.2.1 Step 1: Create Makefiles with CMake

Open CMake and set the source path to the inVRs directory and the binary path to the `build` subdirectory of inVRs. Afterwards press Configure:



CMake 2.6-patch 2

Where is the source code: /Users/rlander/inVRs

Where to build the binaries: /Users/rlander/inVRs/build

Search:

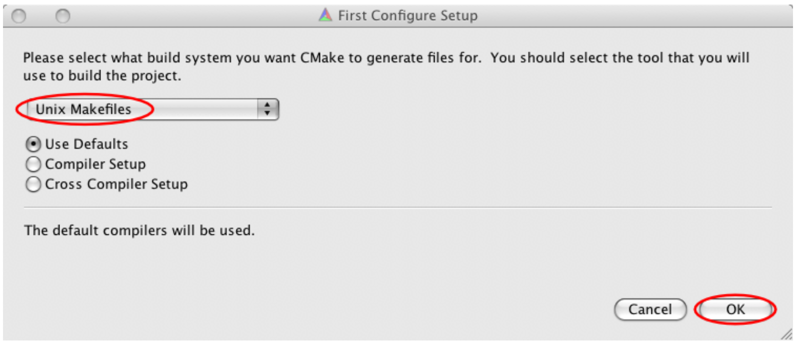
Simple View

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

Current Generator: None

Figure 4.1: Open CMake and set inVRs paths

Now you can select the desired target build system. Currently only Unix Makefiles are tested but other types may work as well. After you selected the target build system press OK:



First Configure Setup

Please select what build system you want CMake to generate files for. You should select the tool that you will use to build the project.

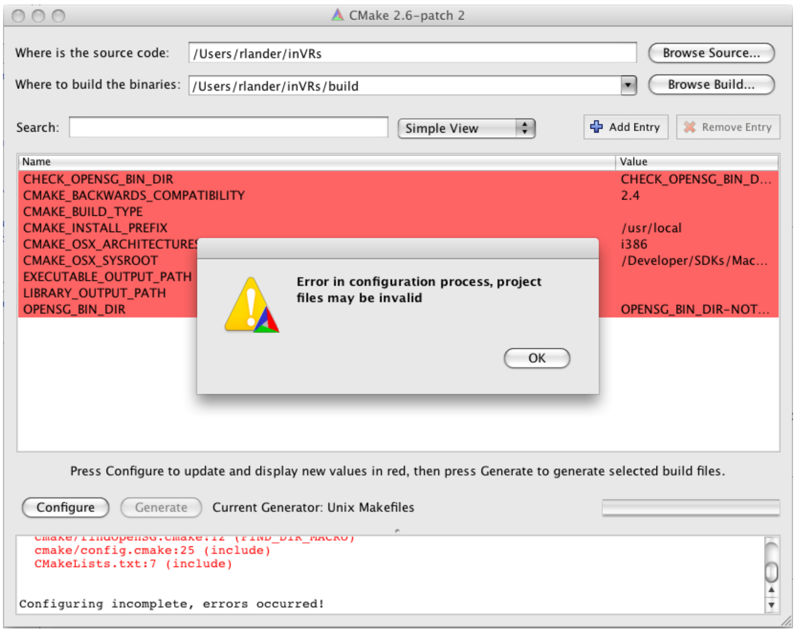
Unix Makefiles

● Use Defaults
○ Compiler Setup
○ Cross Compiler Setup

The default compilers will be used.

Figure 4.2: Select target project type

CMake will now try to create the Makefiles inside the `build` directory. To be able to create the project files CMake tries to automatically find several paths which are required by *inVRs* (like the path to the OpenSG libraries). In a default installation setup CMake should be able to find all paths by itself. But in some cases some paths can not be found by CMake and it will abort and report an error message. The following figure for example shows the error message which occurs if CMake has problems finding the OpenSG path:



Where is the source code: /Users/rlander/inVRs

Where to build the binaries: /Users/rlander/inVRs/build

Search:

Name	Value
CHECK_OPENSG_BIN_DIR	CHECK_OPENSG_BIN_D…
CMAKE_BACKWARDS_COMPATIBILITY	2.4
CMAKE_BUILD_TYPE	
CMAKE_INSTALL_PREFIX	/usr/local
CMAKE_OSX_ARCHITECTURES	i386
CMAKE_OSX_SYSROOT	/Developer/SDKs/Mac…
EXECUTABLE_OUTPUT_PATH	
LIBRARY_OUTPUT_PATH	
OPENSG_BIN_DIR	OPENSG_BIN_DIR-NOT…

Error in configuration process, project files may be invalid

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

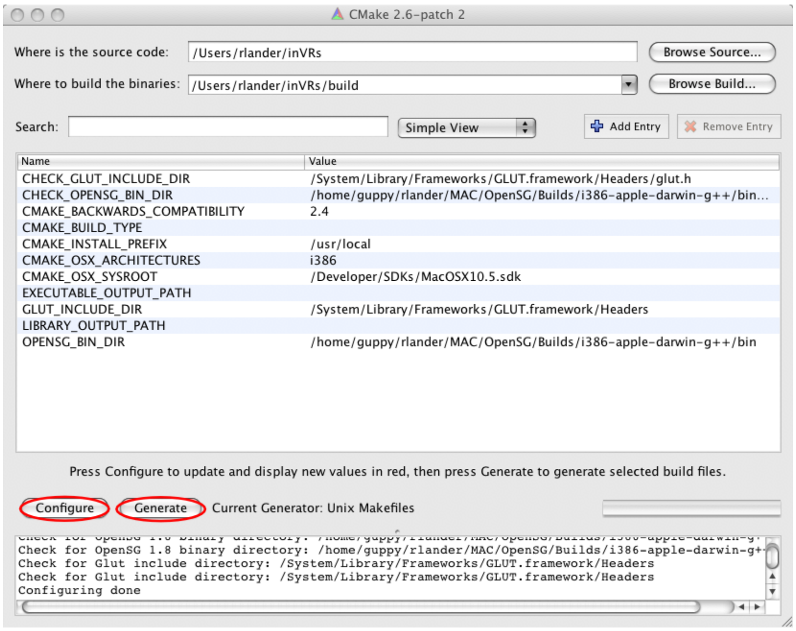
Configure Generate Current Generator: Unix Makefiles

```
  cmake/FindOpenSG.cmake:12 (FIND_DIR_MACRO)
  cmake/config.cmake:25 (include)
  CMakeLists.txt:7 (include)


Configuring incomplete, errors occurred!
```

Figure 4.3: CMake error message when OpenSG path is not found

In such a case proceed to Step 1a (Section 4.2.2) where the manual setting of paths is described. In case Configure finished without errors press Configure again and afterwards press Generate:



Where is the source code: /Users/rlander/inVRs

Where to build the binaries: /Users/rlander/inVRs/build

Search:

Name	Value
CHECK_GLUT_INCLUDE_DIR	/System/Library/Frameworks/GLUT.framework/Headers/glut.h
CHECK_OPENSG_BIN_DIR	/home/guppy/rlander/MAC/OpenSG/Builds/i386-apple-darwin-g++/bin…
CMAKE_BACKWARDS_COMPATIBILITY	2.4
CMAKE_BUILD_TYPE	
CMAKE_INSTALL_PREFIX	/usr/local
CMAKE_OSX_ARCHITECTURES	i386
CMAKE_OSX_SYSROOT	/Developer/SDKs/MacOSX10.5.sdk
EXECUTABLE_OUTPUT_PATH	
GLUT_INCLUDE_DIR	/System/Library/Frameworks/GLUT.framework/Headers
LIBRARY_OUTPUT_PATH	
OPENSG_BIN_DIR	/home/guppy/rlander/MAC/OpenSG/Builds/i386-apple-darwin-g++/bin

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

Configure Generate Current Generator: Unix Makefiles

```
Check for OpenSG 1.8 binary directory: /home/guppy/rlander/MAC/OpenSG/Builds/i386-apple-darwin-g+
Check for OpenSG 1.8 binary directory: /home/guppy/rlander/MAC/OpenSG/Builds/i386-apple-darwin-g+
Check for Glut include directory: /System/Library/Frameworks/GLUT.framework/Headers
Check for Glut include directory: /System/Library/Frameworks/GLUT.framework/Headers
Configuring done
```

Figure 4.4: Finalize configuration

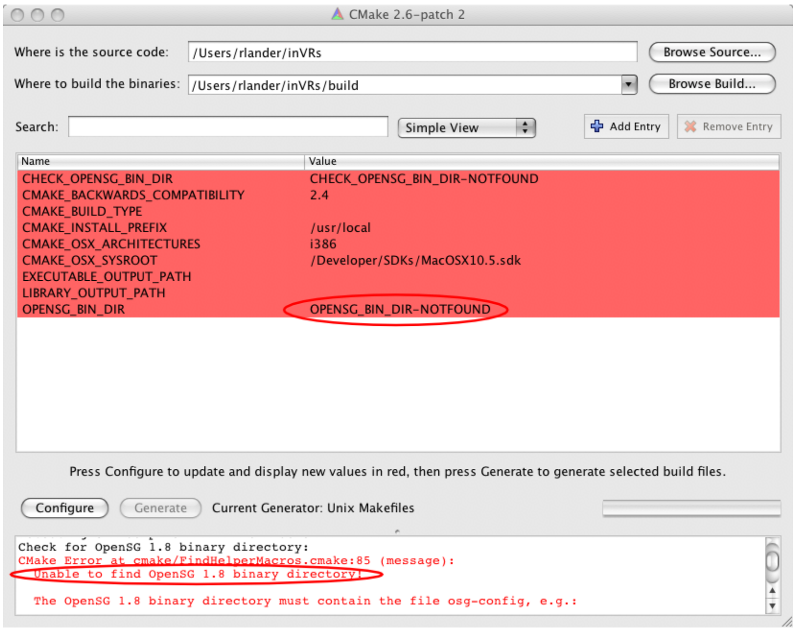
Afterwards you can proceed with Step 2 (Section 4.2.3).

4.2.2 Step 1a (Optional): Set paths for CMake

This section describes how to define paths which should be used by CMake. The instructions here will show how to pass the path to the OpenSG **osg-config** binary, but can be applied for any other path as well.
To pass the path to CMake two possible options are available:

1. Enter the path in the CMake GUI

The first option is to enter the required path directly in the CMake GUI. For example let's assume that CMake was not able to identify the path where the OpenSG files can be found. In this case CMake explains about not finding the path **OPENSG_DIR**:



Where is the source code: /Users/rlander/inVRs

Where to build the binaries: /Users/rlander/inVRs/build

Search:

Name	Value
CHECK_OPENSG_BIN_DIR	CHECK_OPENSG_BIN_DIR-NOTFOUND
CMAKE_BACKWARDS_COMPATIBILITY	2.4
CMAKE_BUILD_TYPE	
CMAKE_INSTALL_PREFIX	/usr/local
CMAKE_OSX_ARCHITECTURES	i386
CMAKE_OSX_SYSROOT	/Developer/SDKs/MacOSX10.5.sdk
EXECUTABLE_OUTPUT_PATH	
LIBRARY_OUTPUT_PATH	
OPENSG_BIN_DIR	OPENSG_BIN_DIR-NOTFOUND

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

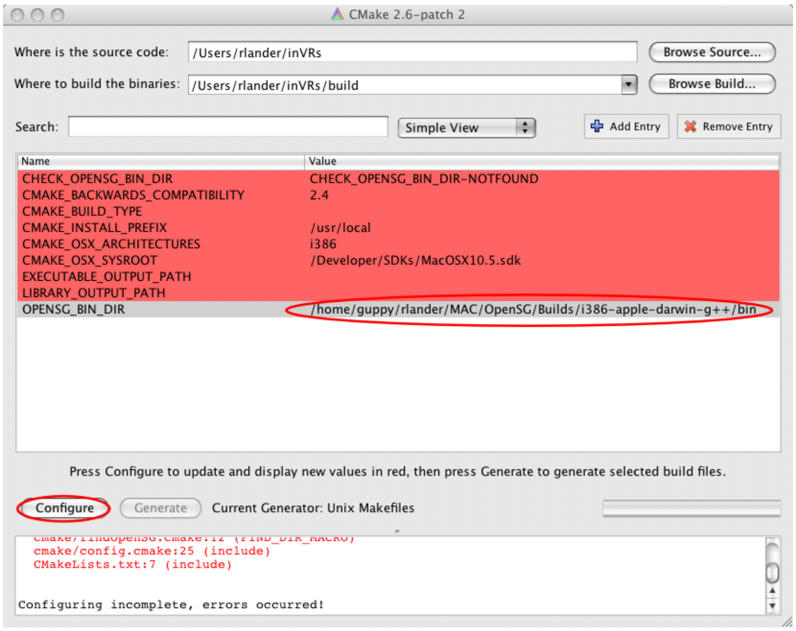
Current Generator: Unix Makefiles

```
Check for OpenSG 1.8 binary directory:
CMake Error at cmake/FindHelperMacros.cmake:85 (message):
  Unable to find OpenSG 1.8 binary directory!

  The OpenSG 1.8 binary directory must contain the file osg-config, e.g.:
```

Figure 4.5: Error when CMake can't find OpenSG path

This path can be passed to CMake by entering it in the correct line in the interface (see Figure 4.6).
The drawback of this method is that the path has to be entered every time the CMake cache is cleaned.



Where is the source code: /Users/rlander/inVRs

Where to build the binaries: /Users/rlander/inVRs/build

Search:

Simple View

Name	Value
CHECK_OPENSG_BIN_DIR	CHECK_OPENSG_BIN_DIR-NOTFOUND
CMAKE_BACKWARDS_COMPATIBILITY	2.4
CMAKE_BUILD_TYPE	
CMAKE_INSTALL_PREFIX	/usr/local
CMAKE_OSX_ARCHITECTURES	i386
CMAKE_OSX_SYSROOT	/Developer/SDKs/MacOSX10.5.sdk
EXECUTABLE_OUTPUT_PATH	
LIBRARY_OUTPUT_PATH	
OPENSG_BIN_DIR	/home/guppy/rlander/MAC/OpenSG/Builds/i386-apple-darwin-g++/bin

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

Current Generator: Unix Makefiles

```
  cmake/FindOpenSG.cmake:12 (FIND_DIR_MACRO)
  cmake/config.cmake:25 (include)
  CMakeLists.txt:7 (include)


Configuring incomplete, errors occurred!
```

Figure 4.6: Enter correct path in GUI

2. Enter path in user.cmake

To set the missing path you can also enter it in the file `user.cmake` which can be found in the *inVRs* main directory. This file is read by CMake at the beginning and allows to define paths and other user-specific CMake commands.
For entering the path to the OpenSG installation just open the `user.cmake` file in the editor of your choice and enter the following entry:

```
set (OPENSG_DIR /Users/rlander/OpenSG/Builds/i386-apple-darwin-g++/bin)
```

4.2.3 Step 2: Build inVRs via console

After CMake successfully created the Unix Makefiles you can start building *inVRs*. First enter the `build` subdirectory of *inVRs* and start building *inVRs* by calling the make command:

```
cd <inVRs_dir>/build
make
```

This will build the *inVRs* libraries in the `build` subdirectory. In order to simplify the usage of the *inVRs* libraries from external projects they have to be installed into a common library directory. This is the same for the *inVRs* header files which are needed for developing *inVRs* applications. Therefore the **install** make target is available which installs these files into the subdirectories `include` and `lib` of the *inVRs* path:

```
cd <inVRs_dir>/build
make install
```

This will install the *inVRs* libraries and headers into the *inVRs* directory in `<inVRs_dir>/lib` and `<inVRs_dir>/include`.

List of Figures