



Modern CMake

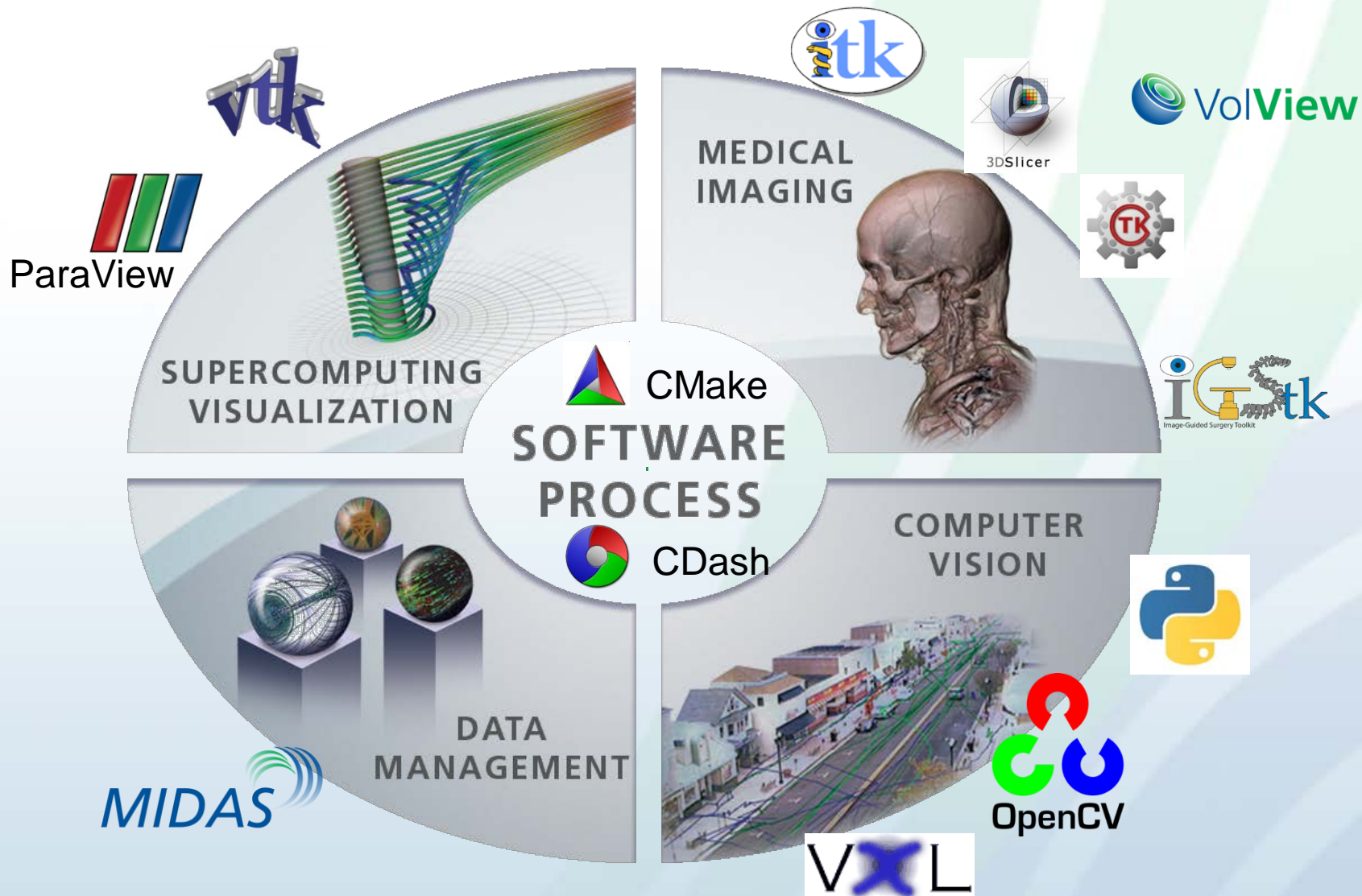
Open Source Tools to Build Test and Deploy C++ Software

Bill Hoffman

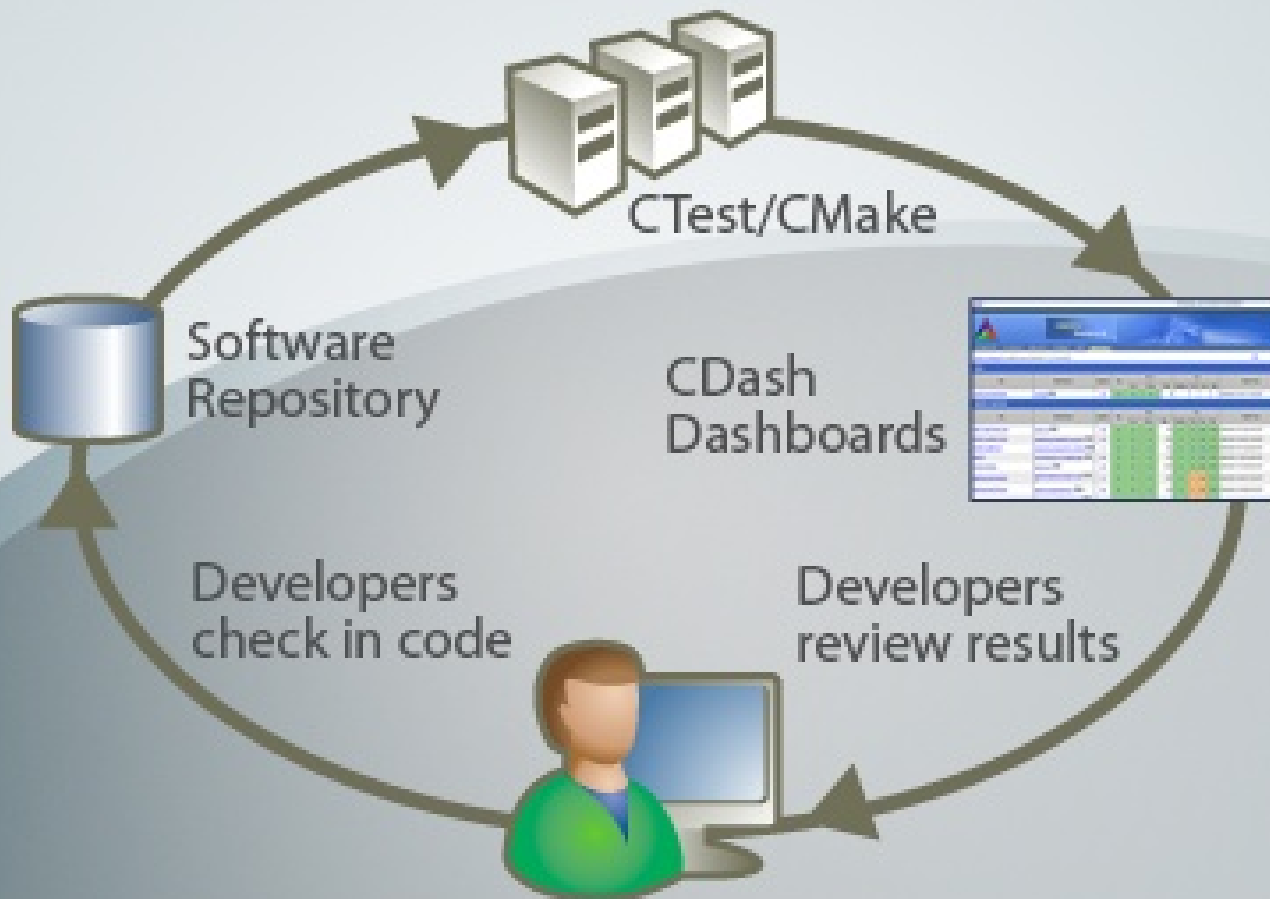
bill.hoffman@kitware.com

Alexander Neundorf

neundorf@kde.org

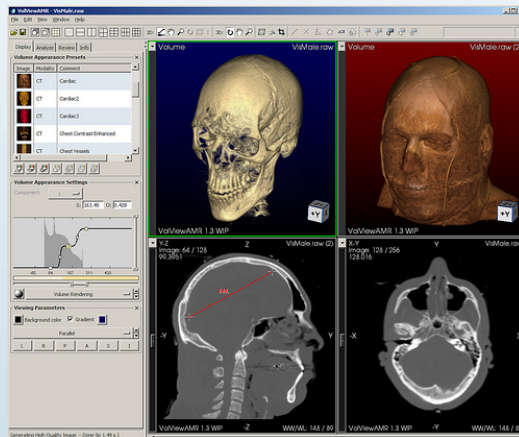


Kitware Quality Software Process



CMake: History

- Built for the Insight Segmentation and Registration Toolkit (ITK)
<http://www.itk.org>
 - Funded by National Library of Medicine (NLM): part of the Visible Human Project
 - CMake
- Release-1-0 branch created in late 2001
- Change the way “everyone” builds c++.



Why CMake? It's easy, and works well

- A build system that just works
- A build system that is easy to use cross platform

Typical Project without CMake (curl)

```
$ ls
CHANGES          RELEASE-NOTES curl-config.in missing
CMake             acinclude.m4  curl-style.el  mkinstalldirs
CMakeLists.txt    aclocal.m4    depcomp        notes
build            docs          notes~
COPYING           buildconf     include        packages
CVS               buildconf.bat install-sh     reconf
ChangeLog         compile      lib            sample.emacs
Makefile          config.guess libcurl.pc.in  src
Makefile.am       config.sub    ltmain.sh      tests
Makefile.in       configure     m4             vc6curl.dsw
README           configure.ac  maketgz
```



```
$ ls src/
CMakeLists.txt  Makefile.riscos  curlsrc.dsp  hugehelp.h  version.h
CVS             Makefile.vc6     curlsrc.dsw  macos       writeenv.c
Makefile.Watcom Makefile.vc8     curlutil.c  main.c      writeenv.h
Makefile.am     config-amigaos.h curlutil.h   makefile.amiga writeout.c
Makefile.b32    config-mac.h     getpass.c   makefile.dj  writeout.h
Makefile.in     config-riscos.h getpass.h    mkhelp.pl
Makefile.inc    config-win32.h   homedir.c   setup.h
Makefile.m32    config.h.in      homedir.h   urlglob.c
Makefile.netware curl.rc          hugehelp.c  urlglob.h
```

Why CMake? It's fast

<http://blog.qgis.org/?q=node/16> : “I was quite surprised with the speed of building Quantum GIS codebase in comparison to Autotools. “

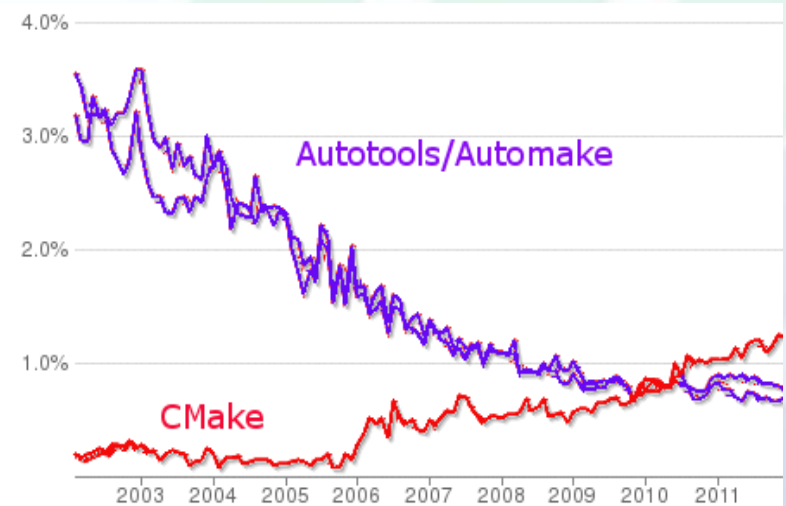
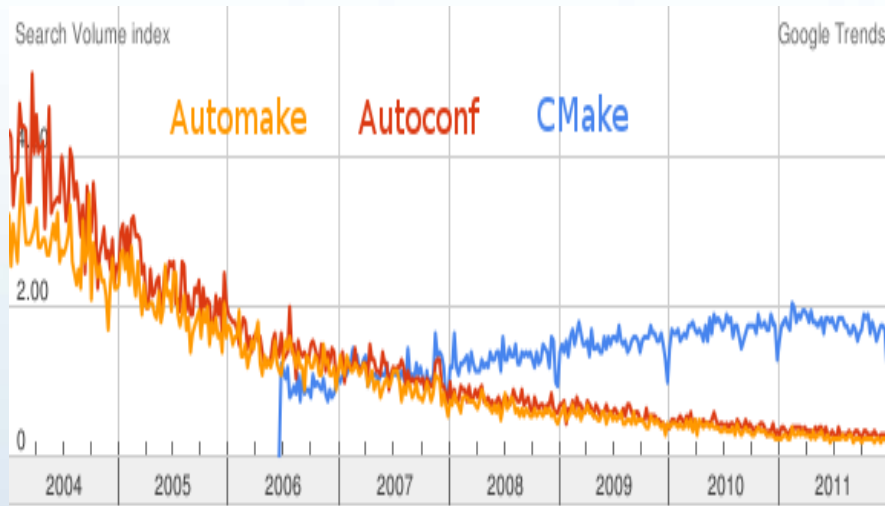
Task	CMake	Autotools
Configure	0:08	Automake: 0:41 Configure: 0:20
Make	12:15	21:16
Install	0:20	0:36
Total	12:43	22:43

<http://taskwarrior.org/projects/taskwarrior/news>

(t i sec)	HDD		SSD	
	autohell	cmake	autohell	cmake
configuration	7.015 10.399	2.592	5.613 7.804	1.589
	17.414	2.592	13.417	1.589
make	92.756	29.790	68.732	17.462
Total	110.170	32.382	82.149	19.051

Why CMake? Everyone is using it

KDE 2006 – Tipping Point!



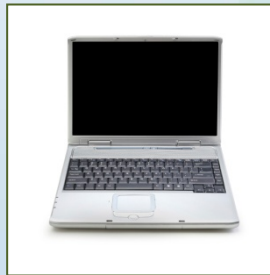
- Google Search Trends and ohloh comparisons with auto*
- 1400+ downloads per day from www.cmake.org
- Major Linux distributions and Cygwin provide CMake packages
- KDE, Second Life, Boost (Experimentally), many others

Why CMake? Quickly adapt to new technologies

- New build IDE's and compilers
 - Visual Studio releases supported weeks after beta comes out
 - Xcode releases supported weeks after beta comes out
 - ninja (command line build tool from google) support contributed to CMake as ninja matured
- New compiler support
 - clang
 - gcc versions

How CMake Changes The Way We Build C++

- Boost aims to give C++ a set of useful libraries like Java, Python, and C#
- CMake aims to give C++ compile portability like the compile once and run everywhere of Java, Python, and C#
 - Same build tool and files for all platforms
 - Easy to mix both large and small libraries



CMake is no longer SCREAM MAKE

- Commands may be uppercase or lowercase

`ADD_EXECUTABLE(Tutorial tutorial.cxx)`
is equivalent to
`add_executable(Tutorial tutorial.cxx)`

- No need to repeat variables
 - `endforeach(MYVAR),`
`endif(THIS AND THAT OR`
`THEOTHER),`
`endmacro(DoReallyCoolStuff),`
`endfunction(DoBetterStuff)`
 - `endforeach(), endif(), endmacro(),`
`endfunction()`

Convert CMake-language commands to lower case

```
author      Kitware Robot <kwrobot@kitware.com>
            Mon, 13 Aug 2012 17:47:32 +0000 (13:47 -0400)
committer   Brad King <brad.king@kitware.com>
            Mon, 13 Aug 2012 18:19:16 +0000 (14:19 -0400)
commit      77543bde41b0e52c3959016698b529835945d62d
tree        ff63e5fbec326c4a5d821e7496c6d2cb52f75b92   tree | snapshot
parent      7bbaa4283de26864b2e55e819db0884771585467   commit | diff
```

Convert CMake-language commands to lower case

Ancient CMake versions required upper-case commands. Later command names became case-insensitive. Now the preferred style is lower-case.

Run the following shell code:

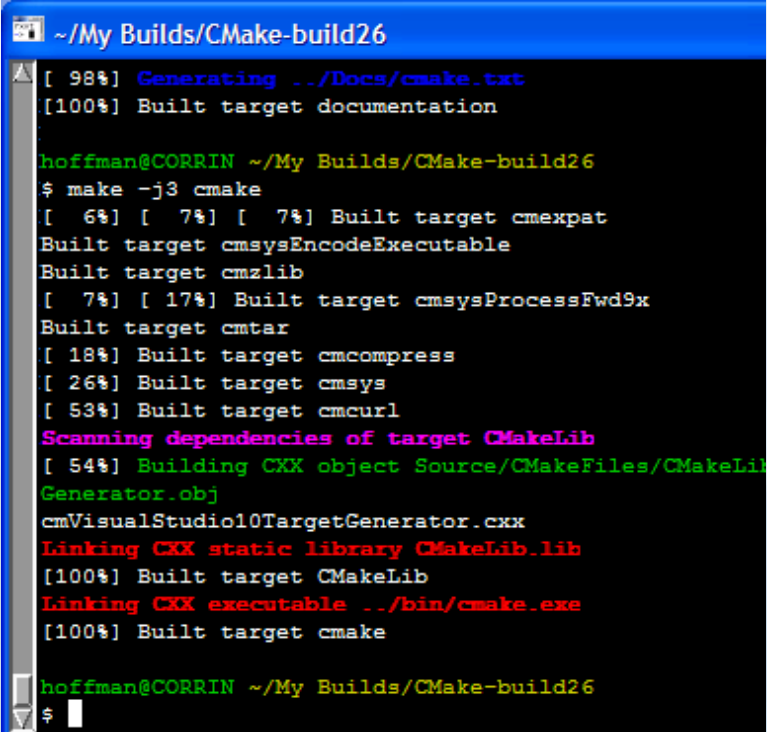
```
cmake --help-command-list |
grep -v "cmake version" |
while read c; do
    echo 's/\b'"$(echo $c | tr '[:lower:]' '[:upper:]')'"/(\s*)'/'"$c"'1(/g'
done >convert.sed &&
git ls-files -z -- bootstrap '*.cmake' '*.cmake.in' '*CMakeLists.txt' |
egrep -z -v '^(Utilities/cm|Source/kwsys/)' |
xargs -0 sed -i -f convert.sed &&
rm convert.sed
```

547 files changed:

CMake Features - continued

- Automatic analysis
 - Implicit dependencies (C, C++, Fortran)
 - Transitive link dependencies
 - Ordering of linker search path and RPATH

- Advanced Makefile generation
 - Modular, Fast, Parallel
 - Color and progress display
 - Help targets – make help
 - Preprocessor targets – make foo.i
 - Assembly targets – make foo.s

A terminal window titled '~ /My Builds/CMake-build26' showing the output of a CMake build. The output includes progress bars for generating documentation, building various targets like cmexpat, cmsysEncodeExecutable, cmzlib, cmsysProcessFwd9x, cmstar, cmcompress, cmsys, cmcurl, and CMakeLib, and linking the final executable. The user 'hoffman@CORRIN' is shown at the prompt.

```
~/My Builds/CMake-build26
[ 98%] Generating ../Docs/cmake.txt
[100%] Built target documentation

hoffman@CORRIN ~/My Builds/CMake-build26
$ make -j3 cmake
[ 6%] [ 7%] [ 7%] Built target cmexpat
Built target cmsysEncodeExecutable
Built target cmzlib
[ 7%] [ 17%] Built target cmsysProcessFwd9x
Built target cmstar
[ 18%] Built target cmcompress
[ 26%] Built target cmsys
[ 53%] Built target cmcurl
Scanning dependencies of target CMakeLib
[ 54%] Building CXX object Source/CMakeFiles/CMakeLib
Generator.obj
cmVisualStudio10TargetGenerator.cxx
Linking CXX static library CMakeLib.lib
[100%] Built target CMakeLib
Linking CXX executable ../bin/cmake.exe
[100%] Built target cmake

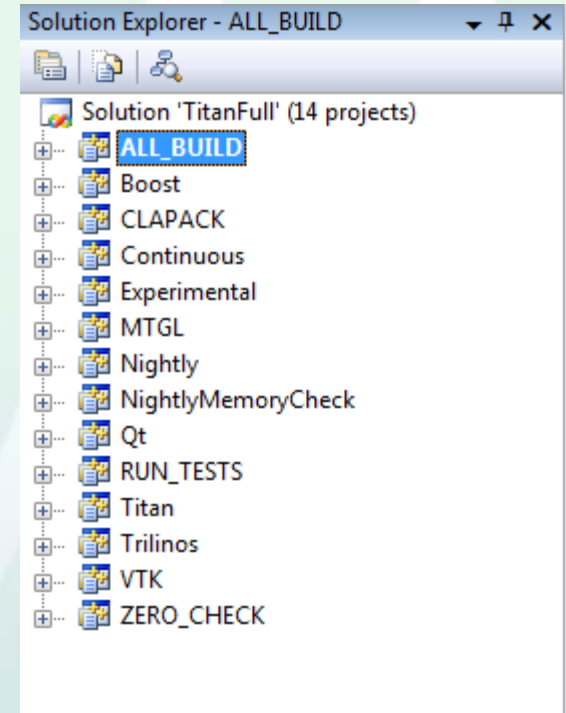
hoffman@CORRIN ~/My Builds/CMake-build26
$
```

CMake Scripts

- `cmake -E` command
 - Cross platform command line utility
 - Ex. Copy file, Remove file, Compare and conditionally copy, time etc
- `cmake -P script.cmake`
 - Cross platform scripting utility
 - Does not generate `cmake_cache`
 - Ignores commands specific to generating build environment

ExternalProject_add

- Module introduced in CMake 2.8
 - Allows the download, configure, build and install of software via custom commands
- Kitware Source Article: October 2009
 - <http://www.kitware.com/products/html/BuildingExternalProjectsWithCMake2.8.html>



Google
Protocol
buffers

CLAPCK

VTK

Qt

Trilinos

Curl

Boost

Titan

Testing with CMake, CTest and CDash

- Testing command in CMake
 - `add_test (testname exename arg1 arg2 arg3 ...)`
 - Executable is expected to return 0 for passed
 - Can set other test passing conditions based on output matching.
- `ctest` – an executable that is distributed with cmake that can run tests in a project.
 - Used for continuous integration testing
 - Client for CDash
 - Can be use for both CMake based projects and other build systems

CDash Dashboard www.cdash.org

CDash - CMake - Mozilla Firefox

http://public.kitware.com/CDash/index.php?project=CMake

Kitware WIKI Dell CDash Dash1old CDash Public Road Runner Web Mail Kitware Calendar Kitware Inc. public.kitware.com: D... CNN.com - Breaking ... CMake Cross Platfor...

Google ugly duckling Search Bookmarks Check AutoLink AutoFill Send to ugly duckling Settings

Start Ora... TravelSer... CDash... can anyon... Communic... PackageM... Re: postfli... LOCAPHO... http:...dges Qt 4.3: Q... Hotel Nov... Google Maps

CMAKE Dashboard

DASHBOARD CALENDAR PREVIOUS CURRENT NEXT PROJECT

[Nightly Changes](#) as of 2008-02-20 21:00:00 EST

Style [Nightly Expected] [Nightly 2.4 Release] [Nightly] [Continuous] [Experimental] [Coverage] [Dynamic Analysis]

Site	Build Name	Update	Cfg	Build			Test					Build Date
				Error	Warn	Min	NotRun	Fail	Pass	NA	Min	
insight.journal.kitware	KWStyle	7	0	0	0	0						2008-02-21 02:28:33 EST

Nightly Expected [Style] [Nightly 2.4 Release] [Nightly] [Continuous] [Experimental] [Coverage] [Dynamic Analysis]

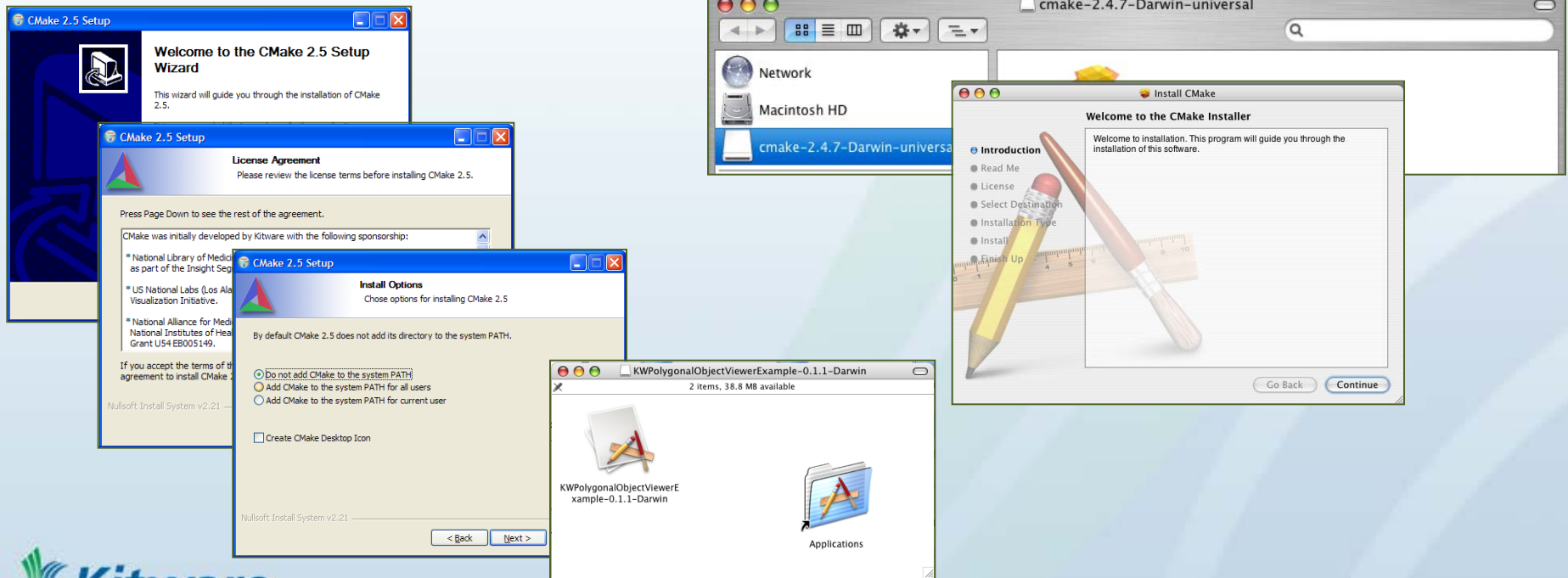
Site	Build Name	Update	Cfg	Build			Test					Build Date
				Error	Warn	Min	NotRun	Fail	Pass	NA	Min	
Titanium.IMTS.us	Linux64-Rocks-ICC-Rel	107	0	0	0	10.7	0	0	93	0	8.1	2008-02-21 10:23:00 EST
krondor.kitware	Darwin-c++	0	0	0	0	59.1	0	1	95	0	55.2	2008-02-21 09:56:00 EST
dash8.kitware	Linux64-g++332	0	0	0	0	6.3	0	0	95	0	18.3	2008-02-21 08:02:00 EST
RogueResearch3	Mac10.5-CMake-Xcode-dbg-ppc64	1	0	0	0	13.1	0	0	90	0	23.4	2008-02-21 05:15:00 EST
RogueResearch3	Mac10.5-CMake-Xcode-dbg-ppc	1	0	0	0	13	0	0	90	0	23.9	2008-02-21 04:28:00 EST

Find: qfilein Next Previous Highlight all Match case

Done

CPack

- CPack is bundled with CMake
- Creates professional platform specific installers
 - TGZ and Self extract TGZ (STGZ), NullSoft Scriptable Install System (NSIS), OSX PackageMaker, RPM, Deb



Simple Qt Example

```
cmake_minimum_required(VERSION 2.8)
project(helloQt)
# find required dependencies
find_package(Qt4 REQUIRED)
# create the executable
add_executable(helloQt WIN32 MACOSX_BUNDLE myqt.cxx )
target_link_libraries(helloQt ${QT_QTMAIN_LIBRARY} ${QT_LIBRARIES})
# installation and packaging
install(TARGETS helloQt DESTINATION bin)
include (InstallRequiredSystemLibraries)
set (CPACK_PACKAGE_VERSION_MAJOR "1")
set (CPACK_PACKAGE_VERSION_MINOR "0")
set(CPACK_PACKAGE_EXECUTABLES "helloQt" "Hello Qt")
include (CPack)
```

Simple Qt Example with Boost

```
cmake_minimum_required(VERSION 2.8)
project(helloQt)
# find required dependencies
find_package(Qt4 REQUIRED)
include(${QT_USE_FILE})
set( Boost_USE_STATIC_LIBS ON )
find_package(Boost REQUIRED signals)
include_directories(${Boost_INCLUDE_DIRS})
# create the executable
add_executable(helloQt WIN32 MACOSX_BUNDLE myqt.cxx )
target_link_libraries(helloQt ${QT_QTMAIN_LIBRARY} ${QT_LIBRARIES}
    ${Boost_LIBRARIES} )
# installation and packaging
install(TARGETS helloQt DESTINATION bin)
include (InstallRequiredSystemLibraries)
set (CPACK_PACKAGE_VERSION_MAJOR "1")
set (CPACK_PACKAGE_VERSION_MINOR "0")
set(CPACK_PACKAGE_EXECUTABLES "helloQt" "Hello Qt")
include (CPack)
```

Finding and using software

- targets with includes and links
- import/export targets
- Alex will talk about