Geant4/CMake/CTest/CDash Working Session Intro

- Topics for 9.6 and X
- Input from you!
- Pick and plan key topics for 9.6 and X
- Identify impacts for X (Plenary 8)

Pre-Session Summary

For 9.6 (Autumn)

- Documentation Update
- geant4-config
- Geant4Config.cmake
- "Geant4.gmk"
- Data/UI/Vis Bugs
- Tests and Examples

Towards X (Post 9.6)

- New G4 CMake API
  - Interface Design?
  - Library Structure?
- Testing System
- GNUmake Retirement?
- **Impacts : Plenary 8**

Topics for Release 9.6

Documentation

- Remaining for 9.6: Bugs 1204, 1280, 1291
- For 9.6, focus should be on users (developer guides for "X"?)
- Expand and contract as needed - but K.I.S.S

Installation Guide

What is good, what has caused confusion/issues?

Integrate Walkthroughs?

Move "how to build apps"?

Remove "developers" section"

App. Dev. Guide

Make first chapter "Your first App"?

Appendices on CMake and GNUmake?

README.txt

Familiar "Quickstart" Summary of Inst. Guide?

"Developer Guide"

Needed/Wanted?
If so, what is scope?

Geant4Config.cmake

- “ProjectConfig” module for Geant4
- Remaining for 9.6:
  - Documentation!
  - Use of “components” vs “options” or both?
  - Mainly for UI/Vis driver selection.

geant4-config

- Unix (bash) interface for non-CMake builds
- Remaining for 9.6: Bugs 1203, 1290, 1328

- Add "--data-dir", "--g4make-file" interfaces?
- Add man page(s)?

“Geant4.GNUmake”

- Remaining for 9.6: Bugs 1232
- Location -> lib/Geant4-9.6.0 (arch dependent)
- Use GNUmake fragment over environment?
    - Advantageous, but... deprecation looming?

Data Installs

- Remaining for 9.6: Bug 1285
- Data can now be installed in custom location
- Now implementing reuse of preinstalled data
  - Build/install mix - CMake “data API”?
- Open issues: binary packages, C++ (versions)

UI/Vis Config

- Remaining for 9.6: Bug 1320
  - Triaged (OpenInventor debug/release)
  - Fix should be straightforward
- Anything special for Mountain Lion or Win7?

Tests

- Stable for 9.6?
- Testing/Shifts for 9.6?
- Feedback on shifts?

Examples/Custom Modules

- Test case for "Geant4Config.cmake" updates
- Use of custom modules, e.g. "FindAIDA.cmake"
    - Use of svn:externals for sharing??
- Balance integration vs testing vs standalone

Topics for Release “X”

Integrate Documents?

- Integrate guides into build ("make doc")?
  - Track changes for "X" with tags(?)
- Need XSL processor and Doxygen (others?)
  - O.k. if it's optional?

GNUmake Retirement?

- Do we want to do this for "X"?
  - 9.6 => robust CMake/bash interfaces
- If so, needs a clear timetable and migration programme for developers and users.
  - I would say has to be in "X"-beta.
- Support? Objections?

G4MT in "X"

- Need **early** input here - concerns for build:
  - Cross-platform (*NIX + Win32)?
  - Compiler flags?
  - Sequential vs MT build (both ala Boost)?
- Internal/external MT dependencies?

Geant4 CMake "API"

- Basically, improving "sources.cmake" for developers, plus other tools
- Several interlocking topics
- Buildsystem AND architecture aspects!

```
# - Include paths…
include_directories(${MYEXT_INCLUDE_DIRS})
include_directories(${PROJECT_SOURCE_DIR}/source/global/
management/include)
include_directories(${PROJECT_SOURCE_DIR}/source/
intercoms/include)

# - Define the Module
geant4_define_module(G4foo
  HEADERS
    G4Foo.hh
  SOURCES
    G4Foo.cc
  GRANULAR_DEPENDENCIES
    G4globman
    G4intercoms
  GLOBAL_DEPENDENCIES
    G4global
    G4intercoms
  LINK_LIBRARIES
    ${MYEXT_LIBRARIES}
)
```

Transient Dependencies

- You #include "foo.hh", but this #includes "bar.hh"
  - So include path to "bar.hh" also needed.
- Minimize these!!!!! Two/Three aspects....

Forward Declarations

```
#include "bar.hh"

class foo {
...
 private:
  bar f_;
};
```

$\longrightarrow$

```
class bar;

class foo {
...
 private:
  bar* f_;
};
```

- Use as much as possible to hide deps
- It can also affect how linking is done

“Modularization”

- Neither global nor granular libraries ideal
- Prefer single structure for clarity (MT?)
- Merge some libraries, break up others?
    - G4global + G4intercoms + ... = “G4Core”?
    - G4processes = “G4EMProcesses” + ...?
- Useful examples from Qt, ITK, Boost?

“Public API”

- “Hide” headers of implementation details in subdirectories:

```
include/                    include/
  foo.hh                      foo.hh
  foo_detail.hh               detail/
  foo_impl.hh                   foo_detail.hh
                              private/
                                foo_impl.hh
```

- Reduces install footprint, clarifies actual API, paves way for “tighter” libraries

sources.cmake

- Transient deps and explicit listing are "hottest" topics!
- How to handle? What interface?
- Things to watch
    - Maximise use of "Vanilla CMake"
    - Generator neutral (Make vs Xcode vs...)
    - Reliable and robust developer workflow

```
# - Optional sources
if(GEANT4_IS_MT)
  set(MT_SOURCES src/G4FooMT.cc)
  if(WIN32)
    list(APPEND MT_SOURCES src/G4FooMT_win32.cc)
  endif()
endif()

# - Define the Module
geant4_add_module(G4foo
  PUBLIC_HEADERS
    include/G4Foo.hh
    include/detail/G4Foo_detail.hh
  PRIVATE_HEADERS
    include/private/G4Foo_impl.hh
  SOURCES
    src/G4Foo.cc
    src/G4Foo_impl.cc
    ${MT_SOURCES}
  LINK_INTERFACE_LIBRARIES
    G4global
    G4intercoms
    MyExt
)
```

Testing System

- Expand unit testing
  - Investigate use of Google Test?
- Documentation for developers
- Tidy up tests/ subdirectory

Binary Packaging

- “Easy” with CPack, and a couple of things to think about:
  - How to handle external dependencies - we do expose some external interfaces
  - How to handle data - download when installing?