

Work On Extended Examples

I. Hrivnacova, IPN Orsay

17th Geant4 Collaboration Meeting,
10 - 14 September 2012, Chartres

Outline

- Examples Review
- Applying Coding Guidelines
- CMake build
- Shared Classes
- Documentation

Examples Reviews

- List of items to be reviewed, available at
 - <https://twiki.cern.ch/twiki/bin/view/Geant4/ExtendedExamplesReview>
- Review outcome:
 - 10 % of examples reviewed by owners (7 from 77)
 - 36% of examples reviewed by not owners (28 from 77)
- Most of reviews outcome were not yet addressed by the examples owners

Examples Reviews Proposal For Next Steps

1) Examples which review has been done:

→ Fix the problems reported in the reviews (by owners)

→ Respond to the review:

- list of the fixed items, proposed planning for those which require more time
- If disagreement with reviewer: inform the WG coordinator/deputy

2) Example not yet reviewed or reviewed only by their owners

→ Add in the work plan for the next year

→ Repeat a call in the whole collaboration for volunteers

→ Do not insist on the review by owner (?)

Applying Coding Guidelines

- The coding guidelines for the examples agreed in 2011 and applied in new basic examples
- Applying them in extended examples requires an effort of all examples owners as the code was already written
 - A schedule for applying rule by rule together with a table for monitoring a progress was proposed on the wiki page:
 - <https://twiki.cern.ch/twiki/bin/view/Geant4/ApplyingCodingGuidelines>
 - 23% of examples (18 of 77) declared as done with the exception of the last guideline (classes description)
 - 48% of examples (37 of 77) declared as in progress (which stops at the rule 1.5 or 2.1)
 - 29% of examples not yet started
 - We have not an automatic tool for checking the rules and monitoring a progress

Applying Coding Guidelines

Next Steps

- How to proceed with the examples whose owners did not participate in the effort ?
 - Biasing (2), eventGenerator/HepMC(3), field (3), geometry/olap (1), parallel(4), parameterisations (1), runAndEvent (4) -> 18 examples
- The guideline 4.3 (adding class description) not done in many cases
 - Do you find it too heavy, should it be removed ?
- Half of examples done partly
 - Could they be finished for 9.6 ?
- A first look for a suitable tool for automatic verification was not successful:
 - A tool used eg. by Root project is not free
 - Other tools, eg. by Google also prescribe the rules

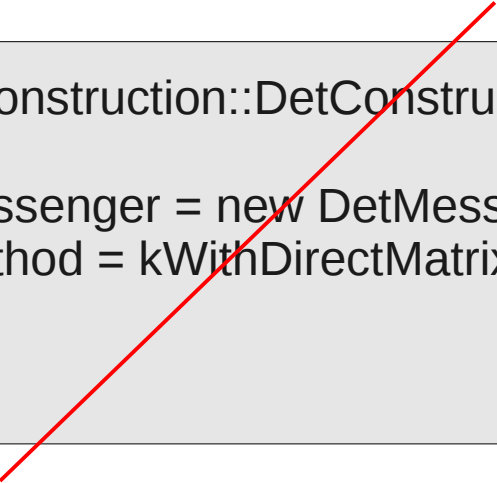
Applying Coding Guidelines

Mis-Interpretation

The initialization list of the class data members (Guideline 2.2)

```
DetConstruction::DetConstruction()
: G4VUserDetectorConstruction(),
  fMessenger(0),
  fMethod(kWithDirectMatrix)
{
  fMessenger = new DetMessenger(this);
}
```

```
DetConstruction::DetConstruction()
{
  fMessenger = new DetMessenger(this);
  fmethod = kWithDirectMatrix;
}
```



CMake Build

- New CMakeLists.txt files follow the uniform format
- ➔ Need agreement on the suggested way how to run example:
 - A) In the build area ?
 - Requires copying macros in the build area; suitable for tests on multiple platforms with using the same source area
 - B) In the source area?
 - Everything (macros, source code) is available; need to define PATH to the executable which is either in the build or in the installation area
- CMakeLists.txt will be updated centrally (by me)
 - The updates require synchronization with ctests
- ➔ When available in SVN, please check and let me (or Peter) know if you find any problems

Shared Classes

- Classes from common
 - Used in eventgenerator/pythia/decayer6, g3tog4/clGeometry (both examples are mine)
 - ExG4HbookAnalysisManager – used in electromagnetic/TestEm*
 - But instead of being used from common, the code is duplicated in each example
- Shared classes
 - Defined in the category/shared and are be used by several examples of one category
 - Analysis/AnaEx01,02,03 – share all classes but one: HistoManager
- More details in the presentation at WG meeting in June 2012
 - <http://willyou.typewith.me/p/tR8JMDtMIV>
- No interest in other examples in reusing the code in this way?

Documentation Configuration Files

- All configuration files in examples/.doxygen
- Combined “global” and “standalone” documentations:
 - “global” documentation: all examples with classes with unique names (A01DetectorConstruction etc.)
 - “standalone” documentation is generated for each example with classes with generic names (DetectorConstruction etc.)
 - Necessary as Doxygen does not support multiple classes with the same name within one project
- The links from “global” to “standalone” documentations implemented in the “module” pages

Documentation Configuration Files (2)

- [Doxyfile](#), [Doxyfile_standalone](#)
 - Doxygen configuration file and the template for generating standalone documentation
- [Doxymodules_category.h](#) files
 - Define the modules per each examples category
 - A module defines the list of classes for the examples which are included in the global documentation
 - And implement links to standalone documentations
- The script for generating standalone documentations:
[generate_standalone.sh](#)
- [Doxyman.h](#) - the main page

Documentation Configuration Files (3)

- Require manual maintenance:
 - When a new example is added or an existing example is removed
 - When classes are added or removed in the examples with global documentation
- *When adding/removing classes in an example, please, let me or Peter know so that the changes are reflected also in the documentation*

How To Generate Documentation

- The steps how to generate Doxygen documentation for all examples are described in [examples/.doxygen/README](#)
 - 1) Check in Doxyfile, Doxyfile_standalone the value of `STRIP_FROM_PATH` variable and adjust it if needed
 - 2) Generate documentation for all examples with unique class names (altogether):
`doxygen >& doxygen.out`
 - 3) Generate documentation for examples with non-unique class names (a standalone documentation will be generated for each example)
`./generate_standalone.sh`

Tips For Further Improvements

- *All macros* provided with an example should be documented in the example README page
- *All commands* implemented in the examples messengers should be documented either in the README page or in the messenger classes (which should be then linked to the page)
 - Can we add these two items in the work plan for the next year?
- *Use the exact class name* when referring to a class in README use the exact class name
 - Eg. [A01DetectorConstruction](#) instead of `DetectorConstruction`
- *Use the class::method name* when referring to a class method in README
 - Eg. [A01DetectorConstruction::Construct\(\)](#) instead of `Construct()`