EM Examples Review

I. Hrivnacova, IPN Orsay

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

EM Examples Review

- Category a bit special
  - The biggest category in extended examples
  - Most of examples use the same patterns
- It was proposed by V. Ivantchenko to review all the examples all together and by the same persons
  - Me and Peter accepted to do this work
- Some of recommendations may apply also to other examples
- The review documents:
  - https://twiki.cern.ch/twiki/bin/view/Geant4/ExtendedExamplesReview
  - A text file following the review template (but one for all examples)
  - A PDF file with general recommendations

Material Definitions

- The material definitions can be shared via a class like Em10Materials in TestEm10
- One set of materials can be shared with all examples, the materials should be built preferable via NIST manager, explicitly defined materials can be demonstrated on materials which (for any reasons) cannot be built via NIST manager.

Detector construction

- Only few geometry setups are used in the examples:
  1. “Container” = World of box shape (TestEm0, 1, 6, 13, 14, 15, 17, 18)
  2. “Ecal” = World of a tube shape (TestEm2, 4)
  3. World with a box absorber optionally with tallies (TestEm5, 7)
  4. World of sphere shape with layers (TestEm12)
  5. And non trivial geometries: TestEm3, 8, 9, 10, 11
- Geometry setups 1 and 2 can be defined by shared classes
- Update geometry is in most examples implemented via reprocessing ConstructVolumes() once again and setting a new world PV to G4RunManager
  - This may be quite inefficient way if we deal with a realistic geometry. Why G4 standard way via G4RunManager::GeometryHasBeenModified() is not used?

Detector Construction Messengers

- The commands for setting the detector parameters (materials or dimensions of a selected volumes) can be generalized in a common way
- A prototype for G4UserParameters and G4UserParametersMessenger
  - User can just define the parameter by its name and properties
  - The set/get function and set command are then generated automatically – user need not to implement them in his code

Physics Lists

- Most of SetCuts() methods may be not needed as they are available in G4VUserPhysicsList base class;
- Also setting the cuts values via commands is available in G4 framework
  - The commands in messengers for setting cuts can be removed and used G4 command instead:

    /testem/phys/setGCut 1 mm ->

    /run/particle/setCutForAGivenParticle “gamma” 1 mm
- There are many variants of PLs without a sufficient explanation in README what is in the “local” physics list different from the standard PL or the standard builder in Geant4
- The variants of PhysListEmStandard with added suffices, eg. PhysListEmStandardSS, PhysListEmStandardNR etc. present in more examples vary from an example to another.
  - It could make users life easier, if these named variants are represented by the same (shared) class and the differences are applied by different setting of the class data

Accounted Data in User Action Classes

- Moving the action classes data members representing accounted data (eg. fEdeposit, fTrackLen etc.) and the methods for their handling in a user defined class (eg. DataManager) can improve code simplicity, readability and re-usability of user action classes:
  - User will find this information in all examples at the same place
  - Stacking, Stepping. Tracking, Event, Run actions can be then shared by most of examples as the data specific actions will be called in a generic way
  - As most of action classes have also their messengers, this can reduce the current number of classes from 18*8=144 to 18*2 + 8 = 44. (Though all examples do not implement all actions, the improvement in maintainability will be still considerable.)
- An example of data manager class and shared user actions was implemented for TestEm1

More ...

- Reduce HistoManager classes and remove HistoMessenger (using new developments in analysis)
- Other classes which can be shared:
  - StepMax, StepMaxMessenger, SteppingVerbose
- The example TestEm1 modified according to the presented recommendations (see the wiki page):
  - TestEm1: 19 classes (ExG4HbookAnalysisManager copied from common)
  - In TestEm1.new: 14 classes; in shared: 5 classes; but 5 classes of 14 still movable in shared
  - When sharing code: 9 classes in the example, 10 shared

G4UserParameters

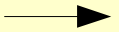
- The classes G4UserParameters, G4UserParametersMessenger are of general use
- Will be proposed to be included in geant4/source/intercoms

G4UserParameters

```
class DetConstruction : public ...{
  public:
     void          SetSize (G4double size);
     G4double   GetSize() connt;
  private:
     G4double   fBoxSize;
};

DetConstruction::DetConstruction() {
  fBoxSize = 10*m;
  ...
}

DetMessenger::DetMessenger(DetConstruction * Det) {
  ...
  fSizeCmd
    = new G4UIcmdWithADoubleAndUnit("/testem/det/se
  fSizeCmd->SetGuidance("Set size of the box");
  fSizeCmd->SetParameterName("Size",false);
  fSizeCmd->SetRange("Size>0.");
  fSizeCmd->SetUnitCategory("Length");
  fSizeCmd->AvailableForStates(G4State_PreInit,G4Sta
...
}
# in macro
/testem/det/set/size 5 m
```



```
class DetConstruction : public ... {
  private:
     G4UserParameters   fParameters;
};


DetConstruction::DetConstruction()  {
   fParameters.Add("boxSize", 10*m, "Length");
  ...
}

DetConstruction::otherFunction() {
  G4double boxSize
     = fParameters.GetDValue("boxSize");
  ...
}

#in macro
/param/boxSize 5 m
```

G4UserParameters

```
// methods to add parameters
//
void Add(const G4String& name, const G4String& value,
         const G4String& candidates = "");
void Add(const G4String& name, const G4int value,
         const G4String& range = "");
void Add(const G4String& name, G4double value,
         const G4String& unitCategory="", const G4String& range = "");
void Add(const G4String& name, const G4ThreeVector& value,
         const G4String& unitCategory="", const G4String& range = "");

// methods to set parameters
//
void Set(const G4String& name, const G4String& value);
void Set(const G4String& name, G4int value);
void Set(const G4String& name, G4double value);
void Set(const G4String& name, const G4ThreeVector& value);

// methods to get parameters
//
G4String GetSValue(const G4String& name) const;
G4int    GetIValue(const G4String& name) const;
G4double GetDValue(const G4String& name) const;
G4ThreeVector Get3VValue(const G4String& name) const;
G4String      GetUnitCategory(const G4String& name) const;
```

G4UserParameters Messenger

```
// some more parameters to test G4UserParameters class
fParameters.Add("myInt", 0, "myIntValue >= 0");
fParameters.Add("myDouble", 0.);
fParameters.Add("myEdep", 0*MeV, "Energy");
fParameters.Add("myEdep2", 5*MeV, "Energy", "myEdep2Value > 0");
fParameters.Add("myField", G4ThreeVector(0, 0, 0.2*tesla), "Magnetic flux density");
fParameters.Add("myDirection", G4ThreeVector(1,0,0));
fParameters.Add("myPosition", G4ThreeVector(1*cm,1*cm,1*cm),
                "Length",
                "myPositionValueX > 1 && myPositionValueY > 1 && myPositionValueZ > 1");
```

Commands generated:

/param/myInt value

/param/myDouble value

/param/myEdep value [unit]

...