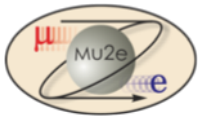

Mu2e Geant4 related Wishes

Rob Kutschke, Krzysztof Genser

Fermilab/SCD

Geant4 Technical Forum

March 27th, 2012

Introduction/List of Wishes

- Mu2e is a muon-to-electron-conversion experiment currently under design at Fermilab.
- Below are Mu2e Geant4 related Wishes:
  1. Refactor G4RunManager::BeamOn
  2. Provide a way to access all Geant4 extensions to the PDG particle information independent of the particle creation
  3. Include muon capture code being developed by Mu2e in a future Geant4 release (once all the provisions for that are met)
  4. Indicate when G4 does(not) take ownership of pointees.

See following pages for more details

G4RunManager::BeamOn

- Mu2e has chosen to use a non-G4 framework to drive our event loop
  - we use a framework named art, supported by Fermilab CS.
- G4 also wants to own the event loop. Our solution is sketched on the next transparency
  - we inherit from G4RunManager and refactor the the BeamOn method into four pieces. We understand that this is a common solution.

G4RunManager::BeamOn cont'd

```
class Mu2eG4RunManager : public G4RunManager{
  public:
    Mu2eG4RunManager();
    virtual ~Mu2eG4RunManager();
    virtual void BeamOnBeginRun ( unsigned int runNumber,
                                  const char* macroFile=0,
                                  G4int n_select=-1);
    virtual void BeamOnDoOneEvent( int eventNumber );
    virtual void BeamOnEndEvent();
    virtual void BeamOnEndRun();
};
```

G4RunManager::BeamOn cont'd

- We call these four BeamOn function variants from our framework, in the appropriate places. In this way, our framework drives the event loop.
  - The issue is that we need to potentially maintain this code every time we move to a new G4 release.
- It would be a great help to us if the G4 collaboration refactored BeamOn so that it is written in terms of a small number of other functions, not necessarily the ones shown earlier.
  - Those who wish to drive their own event loop can call these other functions directly.
  - This would remove the burden of maintaining our code with every new release of G4.
- We understand that it would require consultation with many experiments to choose the appropriate refactoring.

Access to Geant4 PDG Particle ID Code Extensions

- Mu2e frequently runs jobs that process events through G4 and write these events to disk. We then have code that reads back these events to do reconstruction and analysis. In the reconstruction and analysis phase, we do not initialize G4; therefore the G4 particle data table is not available to us. Instead we have our own particle data table (a wrapper around HEPPDT).
- We would like to extend our particle data table to include G4 particle data codes beyond those defined by the PDG table.
  - In particular we would like to include nuclei and ion codes, the corresponding, names, masses, A, Z, electric charge and so on.

Access to Geant4 PDG Particle ID Code Extensions cont'd

- We cannot easily get this information from the G4ParticleTable object as some of the particles are added to the table on an as-needed basis.
  - We can interrogate the table at the end of a long job but we can never be sure that we have all possible particle definitions.
- We would like the G4 Collaboration to either publish this PDG Particle information in a machine readable way or to provide a description of how to extract complete information in a programmatic way.

Include new Muon Capture Code in a Future Geant4 Release

- Mu2e/Fermilab Geant4 group is working on extending Geant4 to include Kevin Lynch's (now with CUNY/Mu2e) code/approach to muon capture and muonic (and possibly other exotic) atoms/molecules formation, decays and other related processes.
- We had a very positive initial discussion with Dennis Wright and are in a process of contacting other Geant4 experts regarding this matter.
- Mu2e would like the resulting code to be included in a future Geant4 release once all the provisions for that are met.

Ownership of Pointees

- In many calls to G4 methods the user passes an object by pointer.
  - In some cases, G4 takes ownership of the object and calls delete on the pointer at the appropriate time.
  - In other cases, the user must call delete.
- There is little documentation: you just gotta know.
- Can we have some method, even structured comments, to indicate which case holds where?
  - It would be OK if this happened over time.