Prototype GEANT4 Service for ATHENA framework

ATLAS Software Workshop

Dec 3 2001

Architecture Session

Charles Leggett <CGLeggett@lbl.gov>

Objectives

- Access GEANT4 services from ATHENA framework
- Use HepMC event as produced by Generators
- Understand various geometry formats:
  - basic GEANT4 C++ geometry classes
  - AGDD
  - DOM
- Use GEANT4 Physics Lists
- Preserve GEANT4 Hits and Tracks for further processing
- Visualization
- Use ATHENA framework facilities for persistification
- Use ATHENA framework facilities for histograms
- Use any other ATHENA framework component...

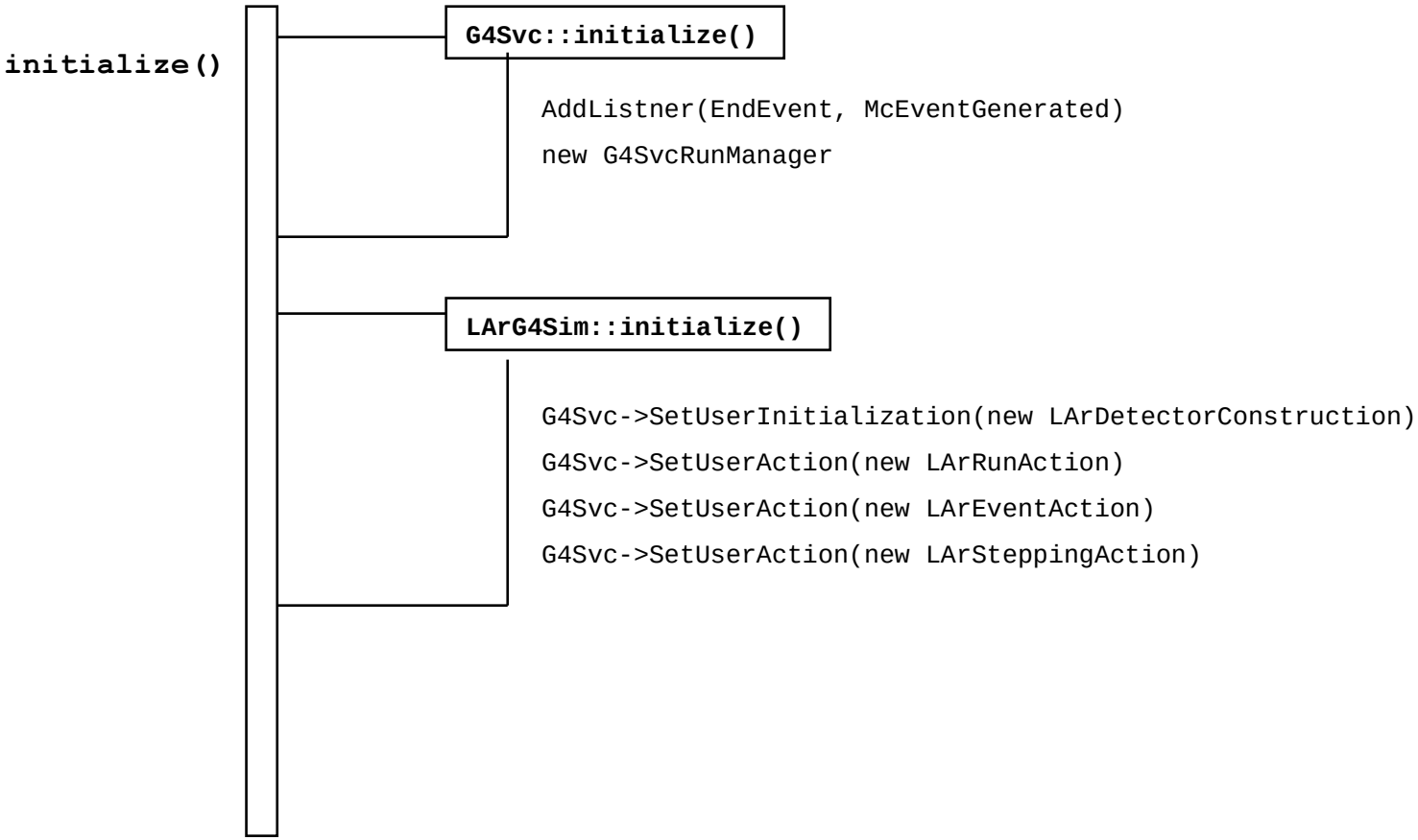
Components

- G4Svc: ATHENA/GAUDI service
  - G4Svc
    - inherits from IService, IG4Svc.
    - usually, only thing the user needs to talk to
  - G4SvcRunManager
    - inherits from G4RunManager
    - extends functionality
    - splits up event loop so G4 event/hit store doesn't get cleared until end of Athena event
  - AthenaHepMCtoG4EventAction
    - converts HepMC event in StoreGate to G4Event
- AGDD geometry builder
  - reads XML files and builds geometry/materials specifications
- DOM geometry builder
  - reads other XML files using DOM model

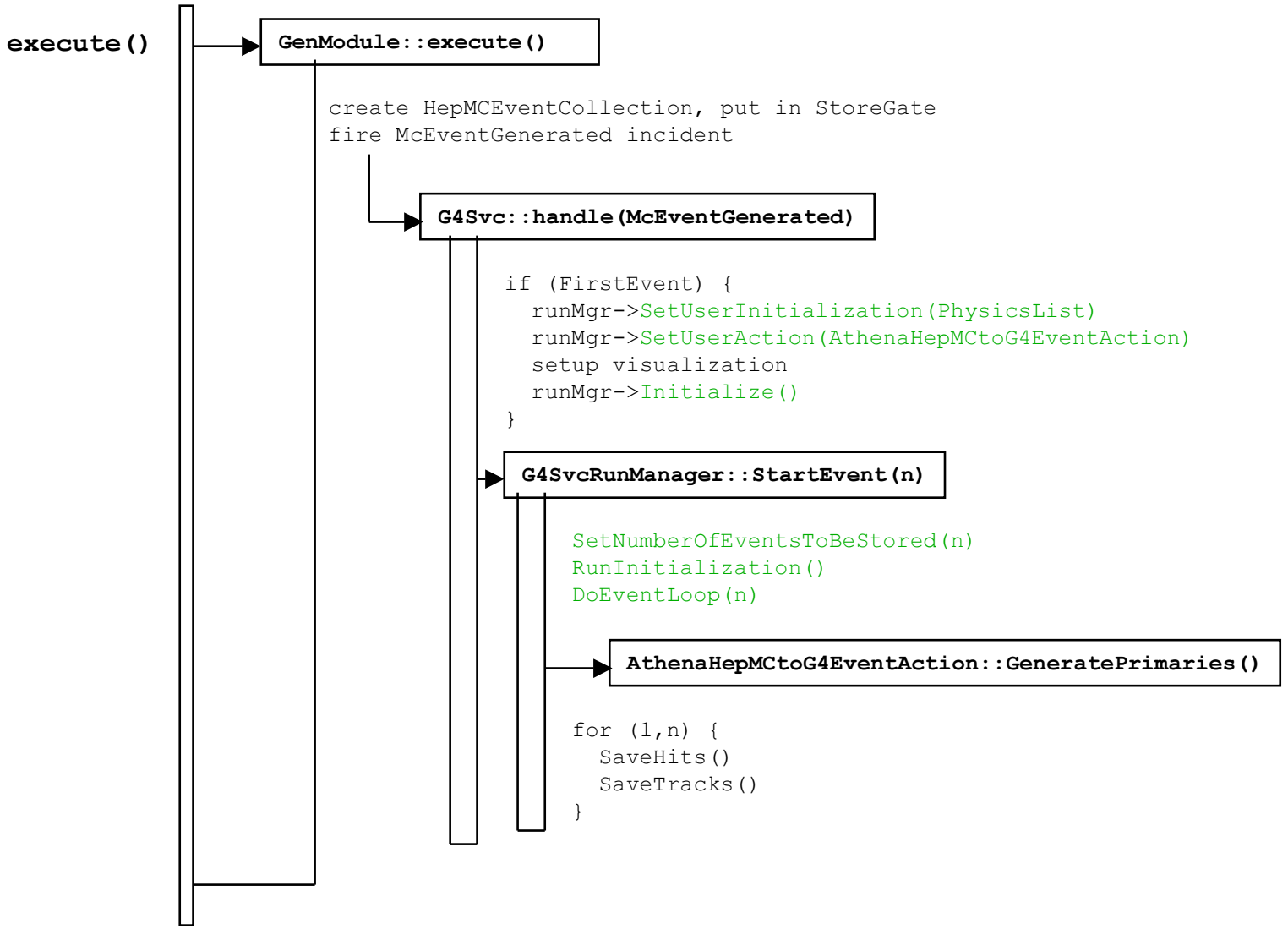
Access to GEANT4 Services

- pure GEANT4 facilities can be accessed in several different ways:
  - using pre-defined G4Svc access methods, such as:
    - `SetUserAction( G4VUserEventAction );`
    - `SetUserInitialization(G4VUserDetectorConstruction* );`
  - direct access to G4RunManager
  - via command line userInterface session
  - sending text commands to the UImanager
  - get hold of various other G4 objects, such as G4Event* and talk directly to them

Process Loop: initialize()

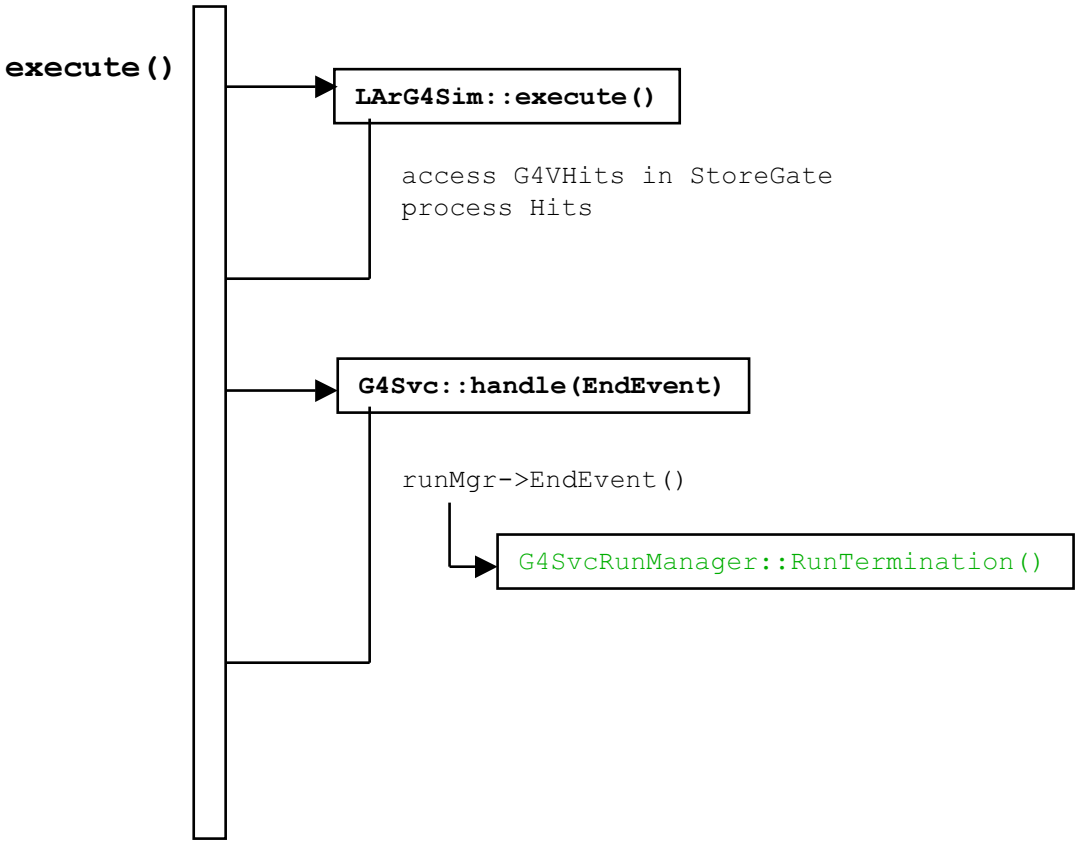


Process Loop: execute()



Process Loop: execute()

cont.



MC Event Conversion

- Convert HepMC::GenEvent as produced by Generators into a G4Event, using a class that inherits from G4VUserPrimaryGeneratorAction
- Can do multiple events per event
- Trivial to do with SingleParticleGun
- For Pythia events, don't need all the particles, can ignore unstable ones
- Problem: particle tree becomes disconnected.
  - put all particles into one primary vertex?
  - create many primary vertices, one for each HepMC::GenVertex
    - causes problems with GEANT when have lots of vertices (segfault)

Physics Lists

- Very simple: copy directly from GEANT4 examples.
- Select via jobOption
  - G4Svc.PhysicsList = “string”
- Available lists:
  - Geantino (ExN01, ExN02)
  - ElectroMagnetic (ExN03)
  - Full (ExN04)
  - “none” → user must supply their own
- Users can also create their own lists, and load them via
  - G4Svc->SetUserInitialization(PhysicsList*)

Saving G4VHits

```
G4SvcRunManager::SaveHits(G4Event* event) {
  event->GetHCofThisEvent()
  new vector<string> hit_keys
  for (HitCollections in event) {
    new vector<G4VHit*> v_HC
    hit_keys->push_back(HitCollection->GetName())
    for (G4VHits in HitCollection) {
      v_HC->push_back(G4VHit)
    }
    StoreGate->record(v_HC,HitCollectionName)
  }
  StoreGate->record(hit_keys,"HitKeys")
}
```

Saving G4VTrajectories

- Very similar to G4VHits, except all stored in one container

```
G4SvcRunManager::SaveTracks(G4Event* event) {

  vector<G4VTrajectory*> *v_TC = new vector<G4VTrajectory*>;
  G4TrajectoryContainer* TC = event->GetTrajectoryContainer();

  for (G4VTrajectory* in TrajectoryContainer) {
      v_TC->push_back(G4VTrajectory*)
  }

  storeGate->record(v_TC,"G4VTrajectory")
}
```

- debug mode prints out all G4TrajectoryPoints in each G4VTrajectory. Turn off when using Pythia!

Accessing the G4Svc

- Tell the jobOptions about it:

```
ApplicationMgr.DLLs    += { "G4Svc" };
ApplicationMgr.ExtSvc  += { "G4Svc" };
```

- Get hold of the service inside your algorithm:

```
#include "G4Svc/IG4Svc.h"

IG4Svc* p_G4Svc;
StatusCode status = service("G4Svc",p_G4Svc);
```

G4Svc Job Options

- G4Svc.PhysicsList:
  - "ExN01", "ExN02", "ExN03", "ExN04 "
  - "Geantino", "EM", "Full", "none"
- G4Svc.DefaultPhysicsCut
  - for a PhysicsList
- G4Svc.Visualize
  - turn on visualization
- G4Svc.VisType
  - default VRML
- G4Svc.Tracking
  - G4RunManager->SetSaveTracks
- G4Svc.RunVerbosity, EventVerbosity, TrackingVerbosity
  - how much G4 output to print out

G4Svc UserInterface

- Can access the G4UI text user interface at any time:
  - `p_G4Svc->StartUISession();`

- Can also pass commands directly to the UImanager:
  - `p_G4Svc->uiMgr()->ApplyCommand( "G4 command" );`

Accessing Hits and Tracks

- As shown before, G4VHits and G4VTrajectories are stored in StoreGate in vectors, keyed by name
- To retrieve hits:

```
const DataHandle< vector<string> > hit_keys;
vector<string>::const_iterator kitr;

StatusCode status = m_sgSvc->retrieve( hit_keys, "HitKeys" );

if (status.isSuccess()) {

   for (kitr=hit_keys->begin(); kitr!=hit_keys->end(); ++kitr) {
      const DataHandle< vector<G4VHit*> > hits;
      status = m_sgSvc->retrieve(hits, (*kitr));

      if (status.isSuccess()) {
         vector<G4VHit*>::const_iterator itr;
         for (itr=hits->begin(); itr!=hits->end(); ++itr) {
            (*itr) -> Print();
         }
      }
   }
}
```

Building Geometry from XML

- Several XML geom/material builders have been implemented:
  - Stan's G4Builder: G4Builder

```
ApplicationMgr.DLLs += { "G4Builder" };
ApplicationMgr.TopAlg = { ... ,  "G4BuilderAlg",  ... };

G4BuilderAlg.MaterialXML = "Material_AGDD.xml";
G4BuilderAlg.DetectorXML = "Atlas_AGDD.xml";
```

  - Jean-Francois's AGDDBuilder: G4AGDDBuilder

```
ApplicationMgr.DLLs += { "G4AGDDBuilder" };
ApplicationMgr.TopAlg = { ... ,  "G4AGDDBuilder",  ... };

G4AGDDBuilder.MaterialXML = "Material_AGDD.xml";
G4AGDDBuilder.DetectorXML = "Atlas_AGDD.xml";
```

  - Andrea's DOM model: G4DOMBuilder

```
ApplicationMgr.DLLs += { "G4DOMBuilder" };
ApplicationMgr.TopAlg = { ... ,  "G4DOMBuilder",  ... };

G4DOMBuilder.XML = { "SCTDesc.xml", "color.xml" };
```

Building AGDD Geometry

- Use AGDD to read in (uncompacted) XML

```
G4VPhysicalVolume* DetectorConstruction::Construct() {

  AGDD_Factory &f = AGDD_Factory::Expat_instance();
  f.build_detector_description (m_materialFile);

  BuildMaterial build_material;
  build_material.parseAGDD (f.get_detector_description() );

  f.build_detector_description (m_detectorFile);

  BuildGeometry build_geometry;
  build_geometry.parseAGDD (f.get_detector_description() );
}
```

Visualization

- Several vis models are currently supported:
  - VRML/vrweb
  - DAWN
  - will add the other G4 standards when I get the chance: had trouble compiling them in under Linux when I first started

- By turning visualization on via the jobOptions, the G4Svc will write out the appropriate visualization file at the end of every event.

Implementations

- GEANT4 novice examples 1 through 4
  - package G4Sim/G4Ex.
  - use Algorithm “G4ExN01”, “G4ExN02”, “G4ExN03”, “G4ExN04”
- Liquid Argon Calorimeter,
  - package G4Sim/LArG4Sim
  - uses C++ classes
- SCT with G4Builder
  - package G4Sim/G4Builder
- SCT,Muon,HEC with G4AGDDBuilder
  - package G4Sim/G4AGDDBuilder
- SCT with G4DOM
  - package G4Sim/G4DOMBuilder

Documentation

- online:
  - http://annwm.lbl.gov/G4
- **CVS:**
  - **currently at BNL**
  - **$CVSROOT =** /afs/rhic/usatlas/project/localcvs
- G4Sim container package contains:
  - GEANT4 (interface package for G4, using geant4.3.2)
  - G4Svc
  - G4Builder
  - G4AGDDBuilder
  - LArG4Sim
  - G4Ex