IEEE Nuclear Science Symposium and Medical Imaging Conference

Short Course

Simulation Techniques Using Geant4

Maria Grazia Pia (*INFN Genova, Italy)*
MariaGrazia.Pia@ge.infn.it

Dresden, 18 October 2008

http://www.ge.infn.it/geant4/events/nss2008/geant4course.html

This course exploits training material developed by several Geant4 Collaboration members: thanks to all of them!

Guided tour of a user application

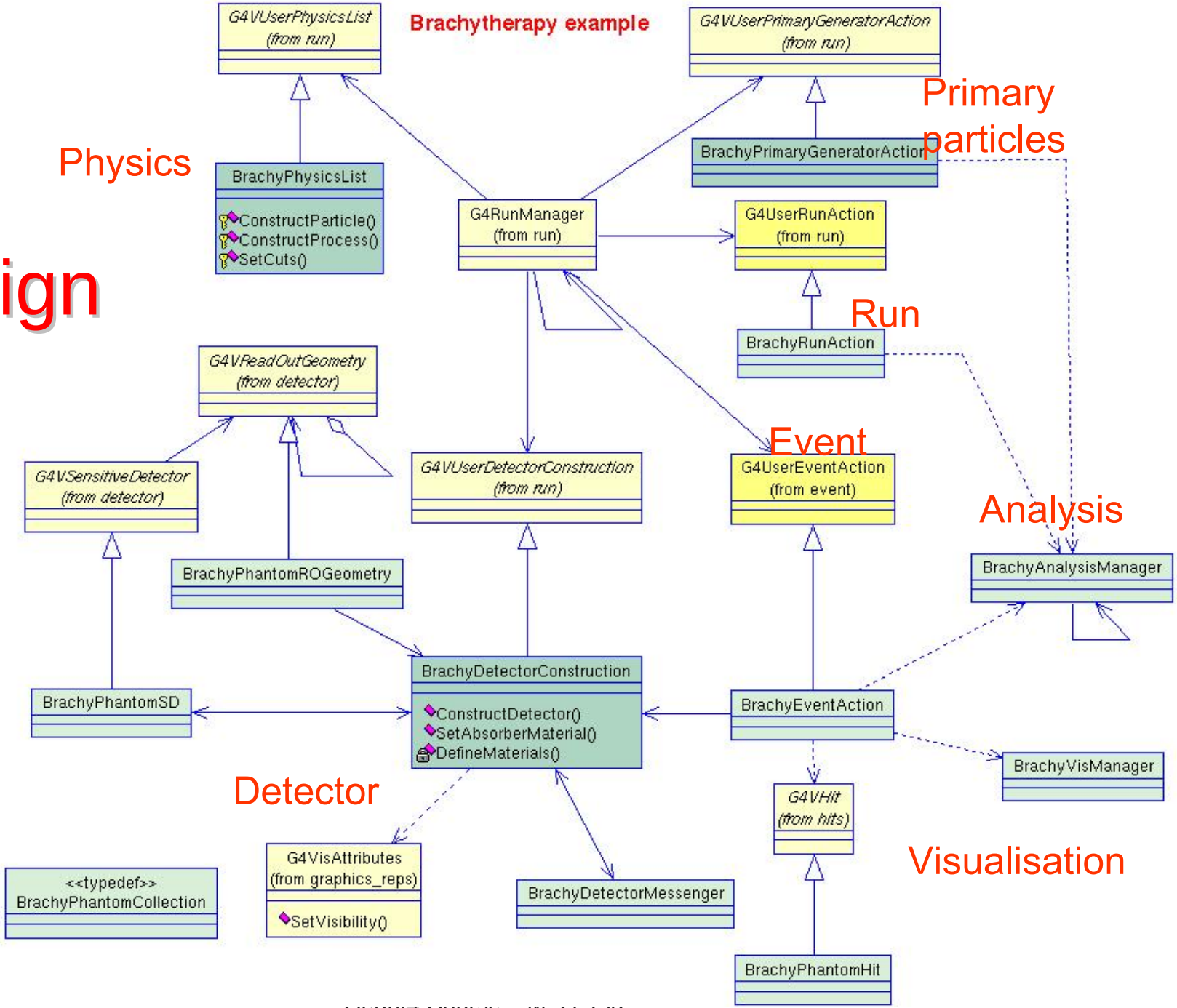
Basic Geant4 concepts in practice

Simple application

- Geant3 brachytherapy advanced example
  - You can find it in Geant4 public distribution Geant4/examples/advanced/brachytherapy
- A radioactive source placed at the center of a cube of water
- The example has some flexibility to select among different types of sources (I, Ir) and different geometries
  - Here we will not go into such details of changing configuration, we only look at the basics

OOAD

Design


Brachytherapy example
G4VUserPhysicsList (from run)
G4VUserPrimaryGeneratorAction (from run)
Physics
Primary particles
BrachyPhysicsList
ConstructParticle()
ConstructProcess()
SetCuts()
G4RunManager (from run)
BrachyPrimaryGeneratorAction
G4UserRunAction (from run)
Run
BrachyRunAction
G4VReadOutGeometry (from detector)
Event
G4VSensitiveDetector (from detector)
G4VUserDetectorConstruction (from run)
G4UserEventAction (from event)
Analysis
BrachyPhantomROGeometry
BrachyAnalysisManager
BrachyDetectorConstruction
ConstructDetector()
SetAbsorberMaterial()
DefineMaterials()
BrachyPhantomSD
BrachyEventAction
BrachyVisManager
Detector
G4VHit (from hits)
Visualisation
<<typedef>> BrachyPhantomCollection
G4VisAttributes (from graphics_reps)
SetVisibility()
BrachyDetectorMessenger
BrachyPhantomHit

Implementation

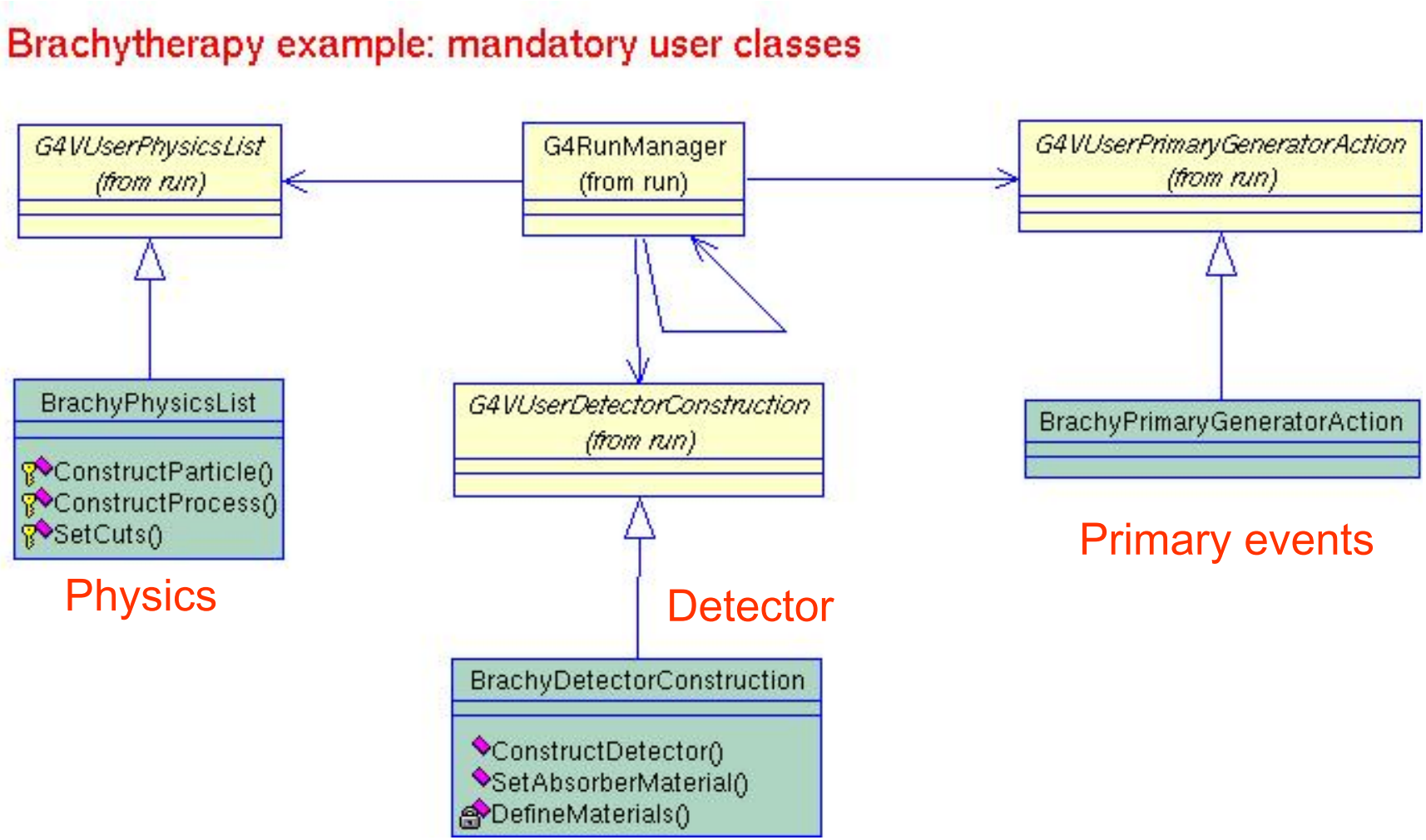
Brachytherapy example

header files in include/*.hh, source code in src/ *.cc
main in Brachy.cc
macro: VisualisationMacro.mac

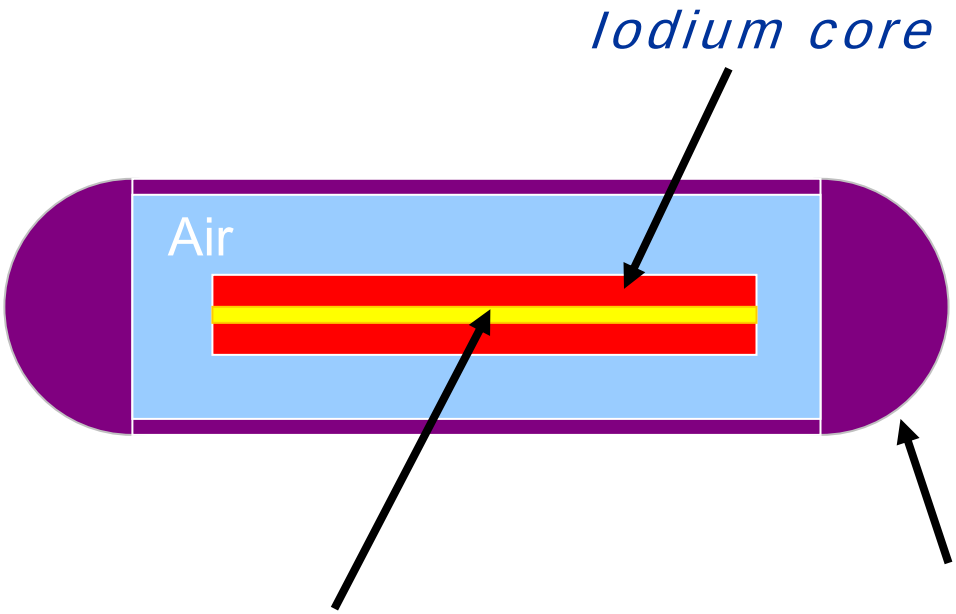
Classes

- BrachyPrimaryGeneratorAction
- BrachyRunAction
- BrachyEventAction
- BrachyPhysicsList
- BrachyVisManager
- BrachyAnalysisManager
- BrachyDetectorConstruction
- BrachyMaterial
- BrachyDetectorMessenger
- BrachyPhantomROGeometry
- BrachyPhantomSD
- BrachyPhantomHit

Mandatory user classes



BrachyDetectorConstruction



Model of a I-125 brachytherapic source geometry and materials

Titanium capsule tip:
Semisphere
radius:0.40mm

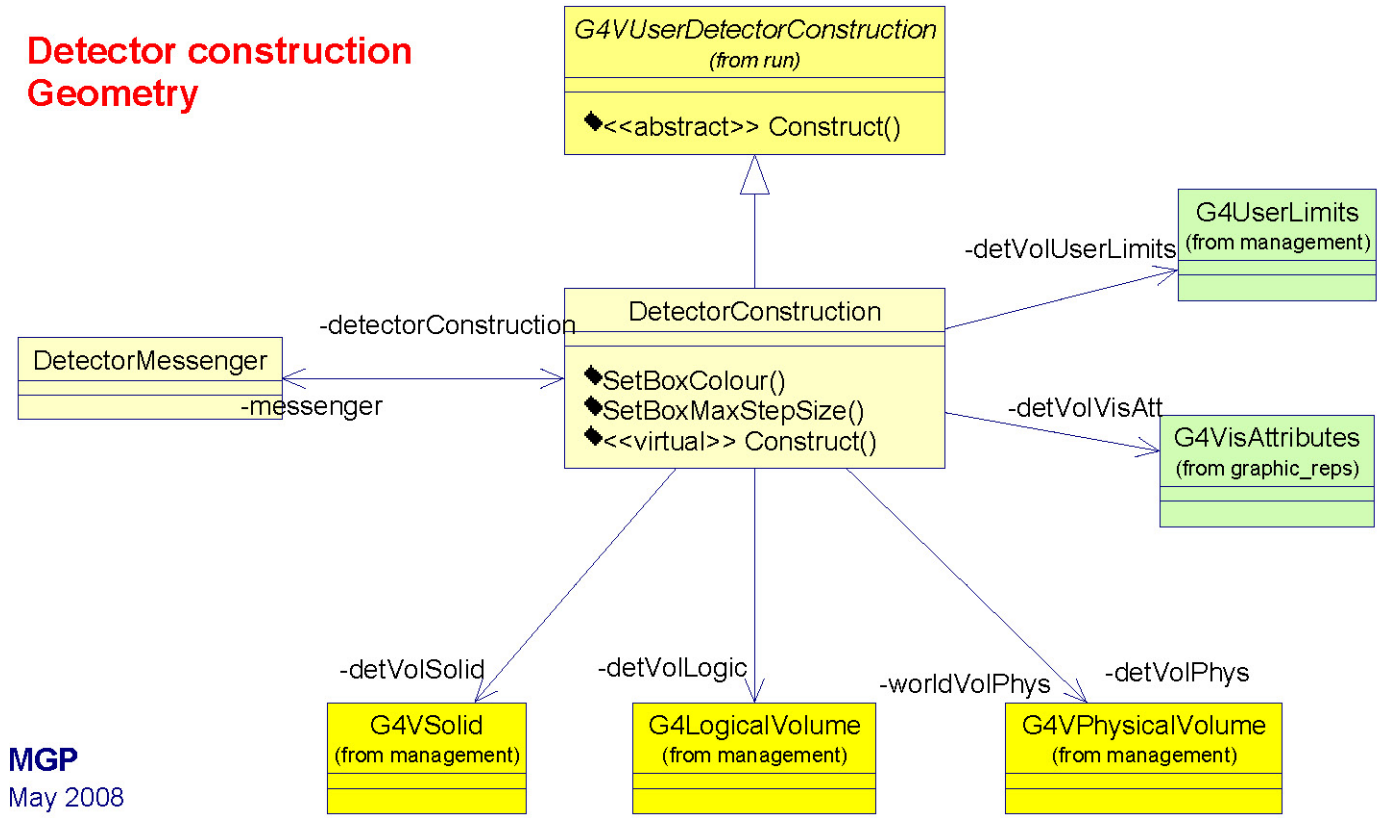
Titanium tube:
Outer radius:0.40mm
Half length:1.84mm

Iodium core:
Inner radius :0
Outer radius: 0.30mm
Half length:1.75mm

Golden marker:
Inner radius :0
Outer radius: 0.085 mm
Half length:1.75mm

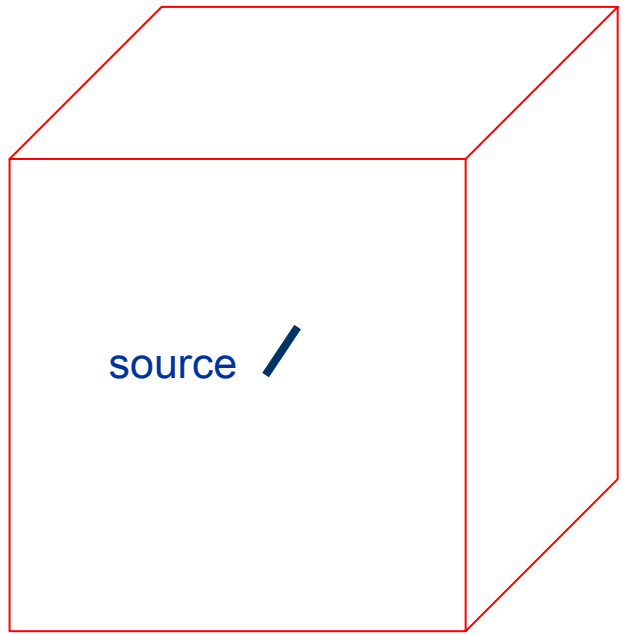
Air:
Outer radius:0.35mm
half length:1.84mm

Refresher: geometry



DetectorConstruction

```
G4VPhysicalVolume* BrachyDetectorConstruction::Construct()
{
  pMaterial-> DefineMaterials();
  ConstructSource();
  ConstructPhantom();
  ConstructSensitiveDetector();
  return worldPhysicalVolume;
}
```



ConstructSource()

// source Bebig Isoseed I-125

construct iodium core and golden marker

the mother volume is an air cylinder

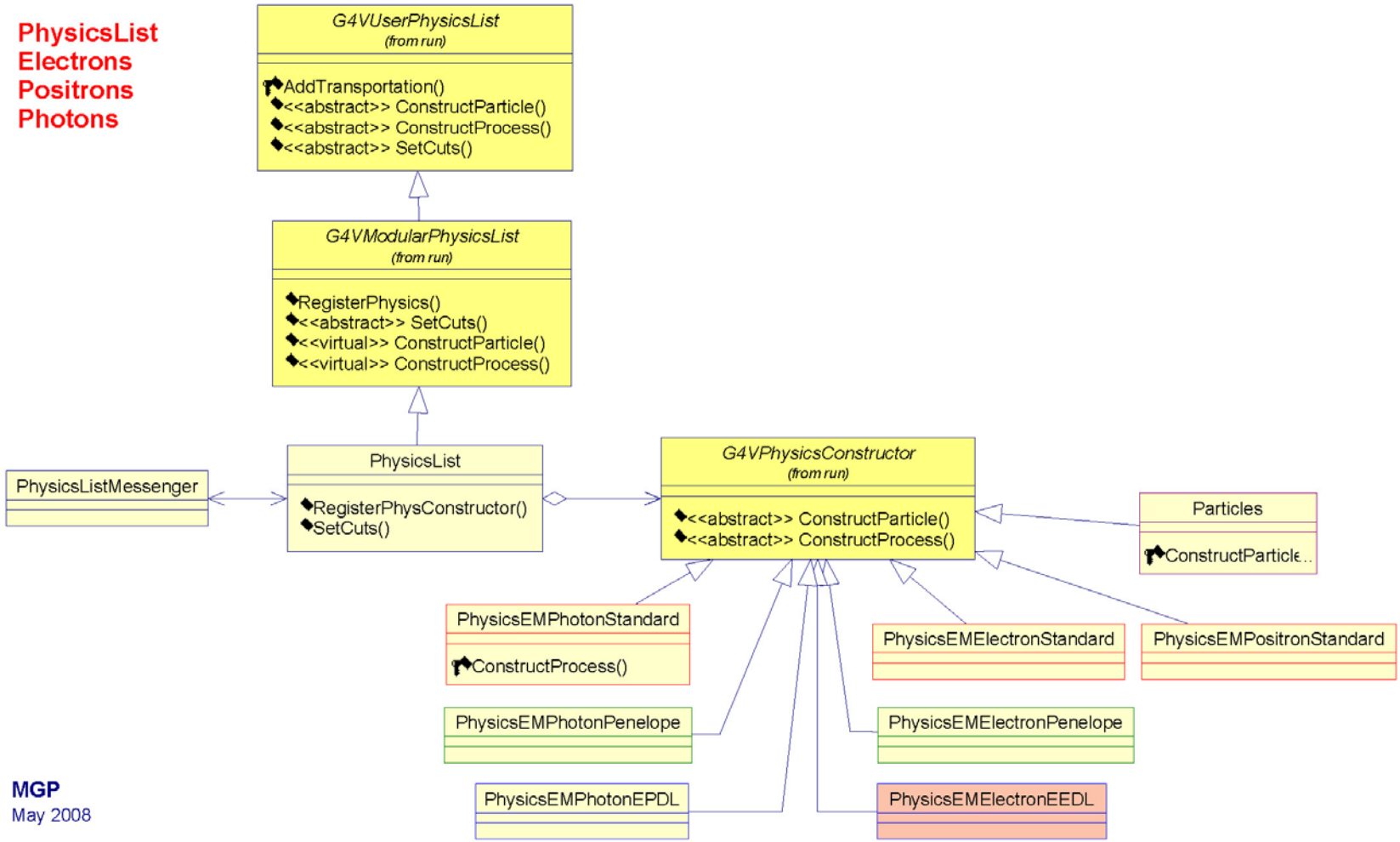
```
// Iodium core
 iodiumCore = new G4Tubs("ICore",0.085*mm,0.35*mm,1.75*mm,0.*deg,360.*deg);
 iodiumCoreLog = new G4LogicalVolume(iodiumCore,iodium,"iodiumCoreLog");
 iodiumCorePhys = new G4PVPlacement(0,  G4ThreeVector(0.,0.,0.), "iodiumCorePhys",
                                    iodiumCoreLog,  defaultTubPhys, false,  0);
// Golden marker
 marker = new G4Tubs("GoldenMarker",0.*mm,0.085*mm,1.75*mm,0.*deg,360.*deg);
 markerLog = new G4LogicalVolume(marker,gold,"MarkerLog");
 markerPhys = new G4PVPlacement(0, G4ThreeVector(0.,0.,0.), "MarkerPhys", markerLog,
                                defaultTubPhys, false, 0);
```

BrachyDetectorMessenger

```
BrachyDetectorMessenger::BrachyDetectorMessenger( BrachyDetectorConstruction* Det):
detector(Det)

{   detectorDir = new G4UIdirectory("/phantom/");
  detectorDir->SetGuidance(" phantom control.");
  phantomMaterialCmd = new G4UIcmdWithAString("/phantom/selectMaterial",this);
  phantomMaterialCmd->SetGuidance("Select Material of the detector.");
  phantomMaterialCmd->SetParameterName("choice",false);
  phantomMaterialCmd->AvailableForStates(G4State_Idle);
 }

void BrachyDetectorMessenger::SetNewValue(G4UIcommand* command,G4String newValue)
{
  if ( command == phantomMaterialCmd )
   { detector->SetPhantomMaterial(newValue);}
}
```

How to change phantom absorber material

Refresher: electron-photon physics



BrachyPhysicsList

```
BrachyPhysicsList::BrachyPhysicsList():
G4VUserPhysicsList()
{
defaultCutValue = 0.1*mm;
…
}

BrachyPhysicsList::~BrachyPhysicsList(){}

void BrachyPhysicsList::ConstructParticle()
{
  ConstructBosons();
  ConstructLeptons();
}

void BrachyPhysicsList::ConstructBosons()
{
  G4Gamma::GammaDefinition();
}
void BrachyPhysicsList::ConstructLeptons()
{
  G4Electron::ElectronDefinition();
  G4Positron::PositronDefinition();
}
```

```
void
BrachyPhysicsList::ConstructProcess()
{
  AddTransportation();
  ConstructEM();
}
```

Add electromagnetic processes

BrachyPhysicsList

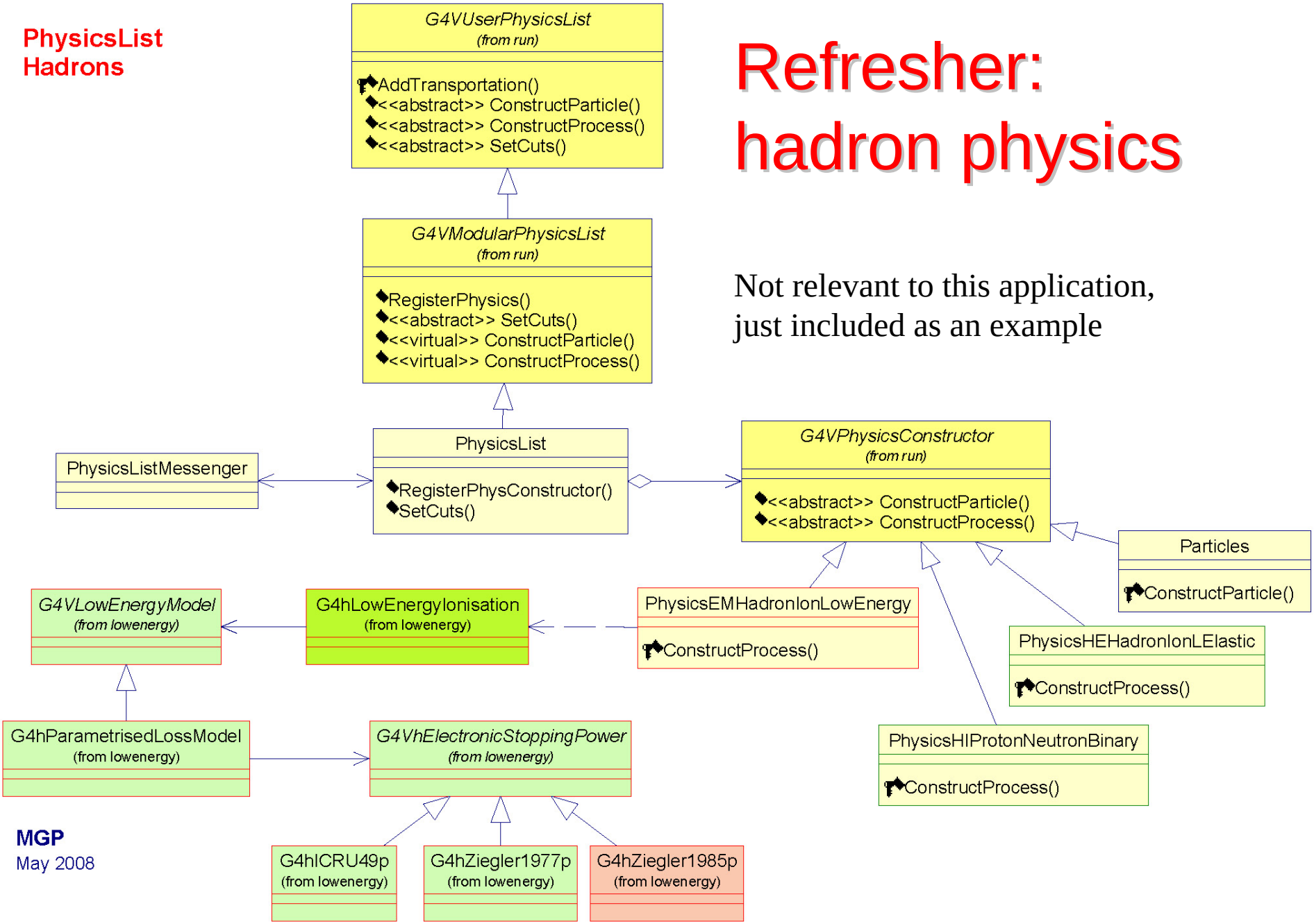
```
void BrachyPhysicsList::ConstructEM()
{ theParticleIterator->reset();
  while( (*theParticleIterator)() ){
    G4ParticleDefinition* particle = theParticleIterator->value();
    G4ProcessManager* processManager = particle->GetProcessManager();
    G4String particleName = particle->GetParticleName();
if (particleName == "gamma") {
    processManager->AddDiscreteProcess(new G4LowEnergyRayleigh);
      processManager->AddDiscreteProcess(new  G4LowEnergyPhotoElectric);
      processManager->AddDiscreteProcess(new G4LowEnergyCompton);
      processManager->AddDiscreteProcess(new G4LowEnergyGammaConversion);
    } else if (particleName == "e-") {
      loweIon  = new G4LowEnergyIonisation("LowEnergyIoni");
      loweBrem = new G4LowEnergyBremsstrahlung("LowEnBrem");
      processManager->AddProcess(new G4MultipleScattering, -1, 1,1);
      processManager->AddProcess(loweIon,     -1, 2,2);
      processManager->AddProcess(loweBrem,    -1,-1,3);
} else if (particleName == "e+"){…}
…
}
```

Set EM processes for e-, e+, gamma

PhysicsList
Hadrons

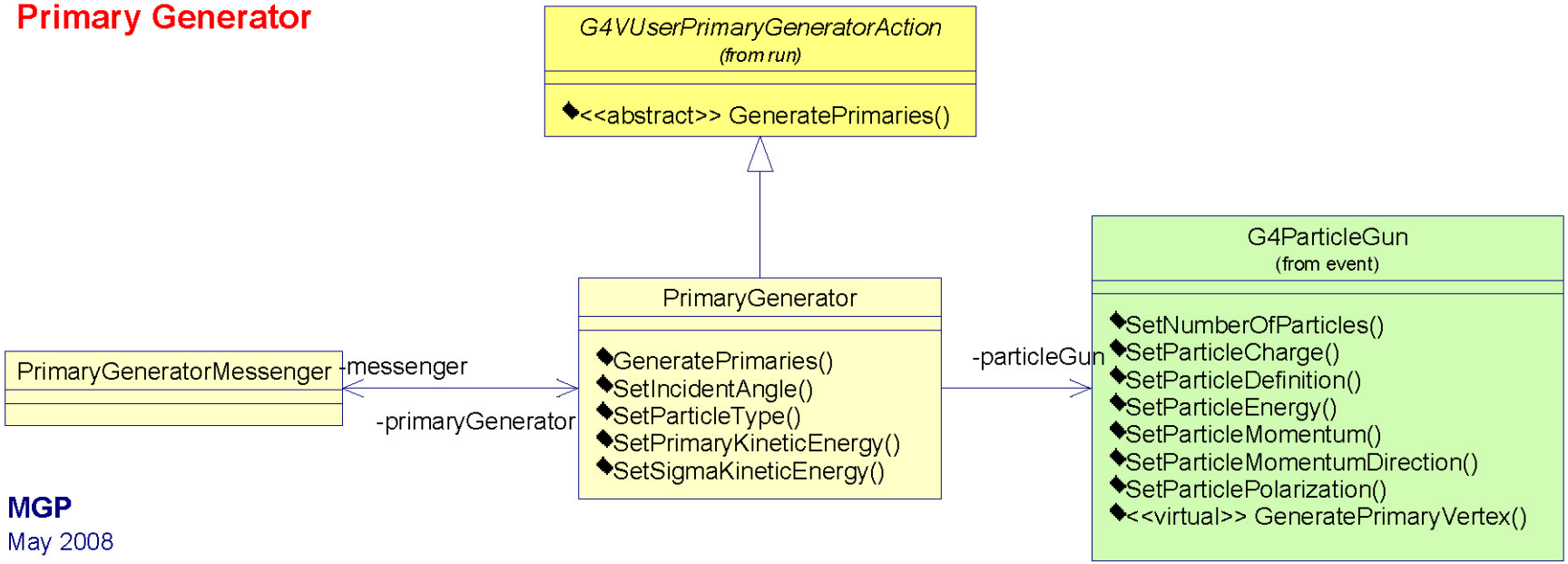
Refresher: hadron physics

Not relevant to this application, just included as an example



Refresher: primary particle generation

Primary Generator



MGP
May 2008

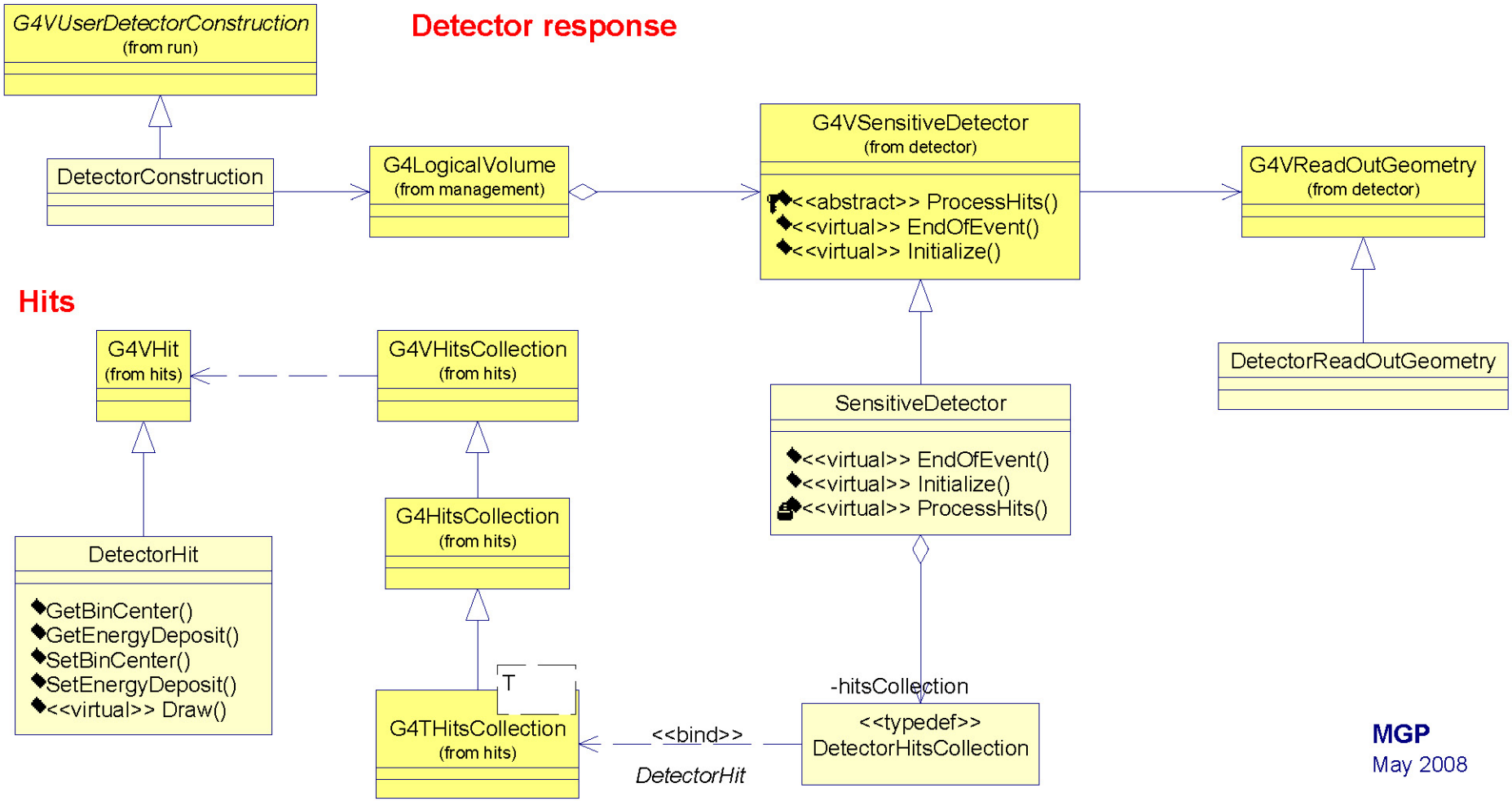
BrachyPrimaryGeneratorAction

- I-125 delivers gamma
- Gamma Energy Spectrum
- Random direction
- Random position inside the iodium core

Energy(keV)	Probability
27.4	0.783913
31.4	0.170416
35.5	0.045671

```
void BrachyPrimaryGeneratorAction::GeneratePrimaries(G4Event* anEvent)
{
…..
particleGun->SetParticlePosition(position);
particleGun -> SetParticleDirection(direction);
particleGun -> SetParticleEnergy(energy);
particleGun->GeneratePrimaryVertex(anEvent);
}
```

Refresher: detector response



MGP
May 2008

Energy deposit

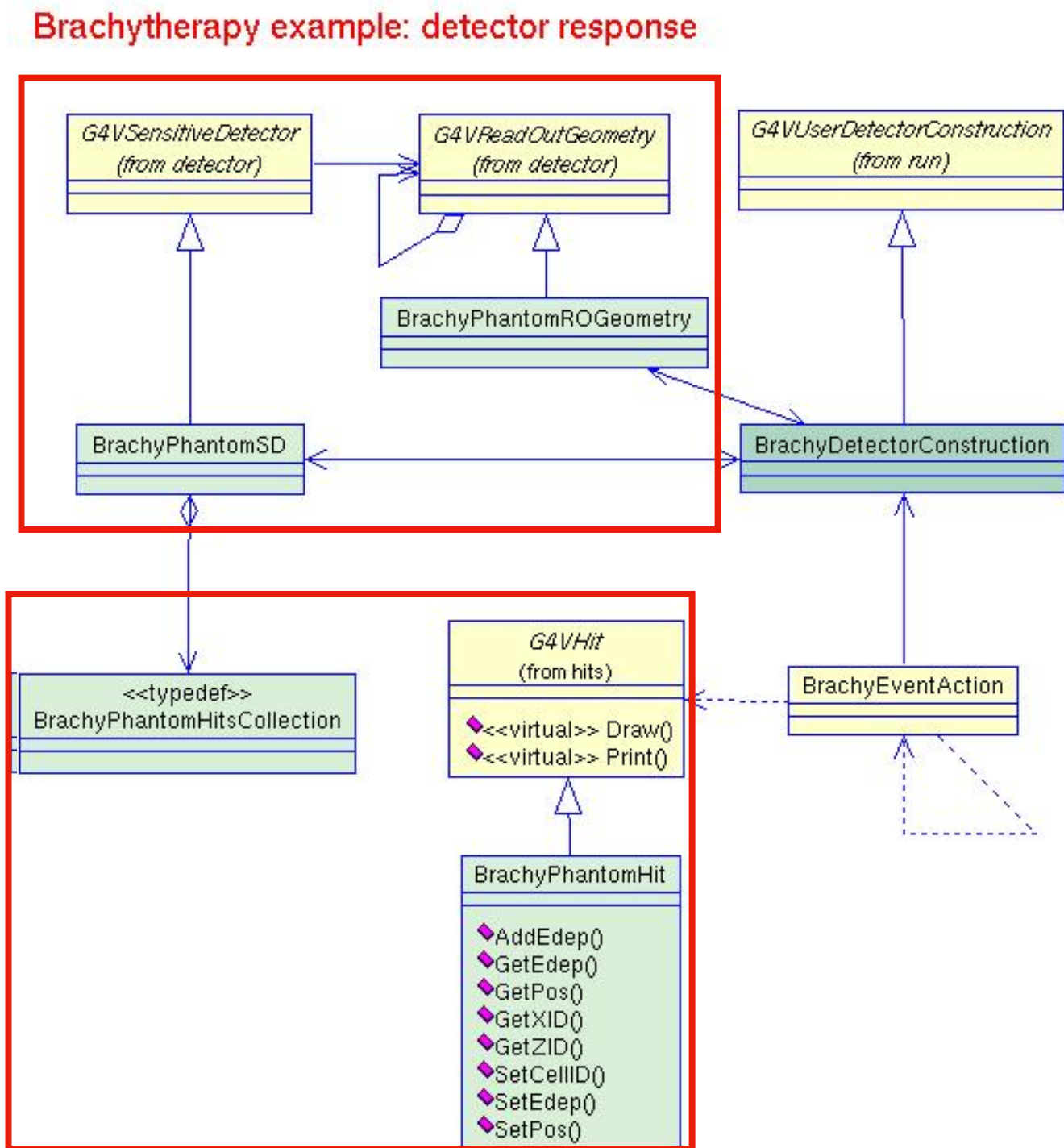
How to retrieve the energy deposit in the phantom

Concepts:

– Hits

– Sensitive Detector

– Readout Geometry



Sensitive Detector and RO Geometry

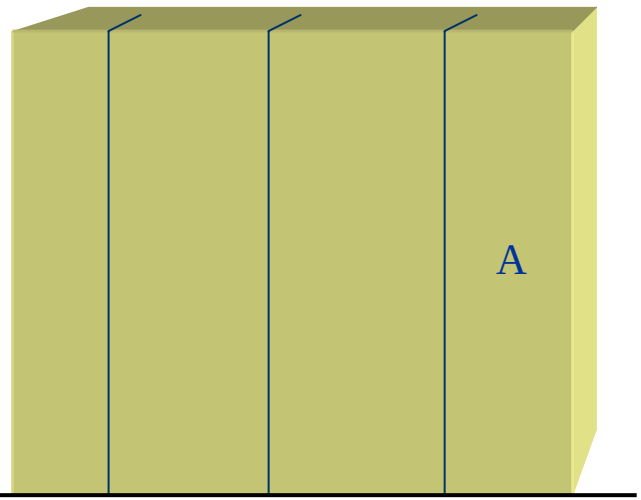
```
void  BrachyDetectorConstruction::ConstructSensitiveDetector()
{
  G4SDManager* pSDManager = G4SDManager::GetSDMpointer();
  if (!phantomSD){
    phantomSD = new BrachyPhantomSD (sensitiveDetectorName,numberOfVoxelsAlongX,
                                                    numberOfVoxelsAlongZ);
    G4String ROGeometryName = "PhantomROGeometry";
    phantomROGeometry = new BrachyPhantomROGeometry(ROGeometryName,
        phantomDimensionX,phantomDimensionZ,numberOfVoxelsAlongX,numberOfVoxelsAlongZ);
    phantomROGeometry->BuildROGeometry();
    phantomSD->SetROgeometry(phantomROGeometry);
    pSDManager->AddNewDetector(phantomSD);
    phantomLog->SetSensitiveDetector(phantomSD);
  }
}
```

In PhantomDetectorConstruction

The phantom is devided in voxels

RO Geometry

```
BrachyPhantomROGeometry::BrachyPhantomROGeometry() {}

BrachyROGeometry::~BrachyROGeometry() {}

G4VPhysicalVolume*  BrachyPhantomROGeometry :: Build()
{
 // example : X division
 ROPhantomXDivision = new G4Box( ….);

 ROPhantomXDivisionLog = newG4LogicalVolume(….);

 ROPhantomXDivisionPhys = new G4PVReplica(….);
……..
}
```



Sensitive Detector and Hit

```
G4bool BrachyPhantomSD::ProcessHits(G4Step* step,G4TouchableHistory*
history)
{…
 G4double energyDeposit = step->GetTotalEnergyDeposit();
…
G4VPhysicalVolume* physVol = history->GetVolume();
  // Read voxel indexes: i is the x index, k is the z index
  G4int k = history->GetReplicaNumber(1);
  G4int i = history->GetReplicaNumber(2);
  G4int j= history->GetReplicaNumber();
…
  BrachyPhantomHit* phantomHit = new BrachyPhantomHit(
                                          physVol ->GetLogicalVolume(), i,j,k)
  phantomHit->SetEdep(energyDeposit);
  phantomHit->SetPos(physVol->GetTranslation());
…
}
```

Store the energy deposit in one hit

BrachyPhantomHit (header file)

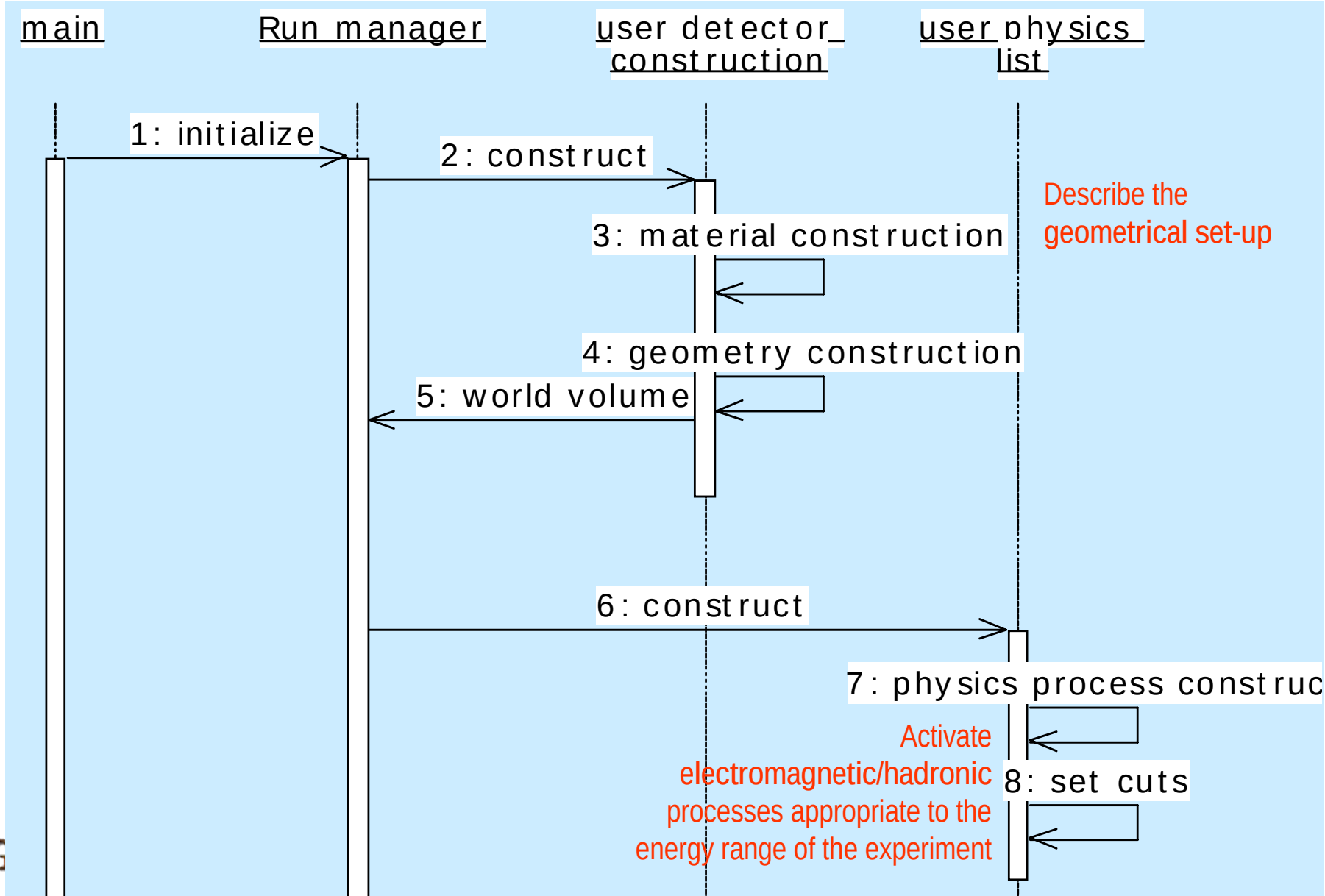
```
class BrachyPhantomHit : public G4VHit
{
public:
  BrachyPhantomHit(G4LogicalVolume* , G4int , G4int , G4int );
  ~BrachyPhantomHit();
…..

  inline G4int GetXID() {return xHitPosition;}  // Get hit x coordinate
  inline G4int GetYID() {return yHitPosition;}  // Get hit y coordinate
  inline G4int GetZID() {return zHitPosition;}  // Get hit z coordinate
  inline  G4double GetEdep() {return energyDeposit;}  // Get energy deposit
….}
```

```
void BrachyEventAction::EndOfEventAction(const G4Event* event)
{…
  G4HCofThisEvent* HCE = event->GetHCofThisEvent();
  BrachyPhantomHitsCollection* CHC = 0;
   if (HCE) CHC = HCE->GetHC(hitsCollectionID);
 if (CHC)
   {
     G4int hitCount = CHC->entries();
     for (G4int h = 0; h < hitCount; h++)
      {
        G4int i = ((*CHC)[h])->GetZID();
        G4int k = ((*CHC)[h])->GetXID();
        G4int j = ((*CHC)[h])->GetYID();
        G4double energyDeposit=((*CHC)[h]->GetEdep());
       …}
…}
…}
```

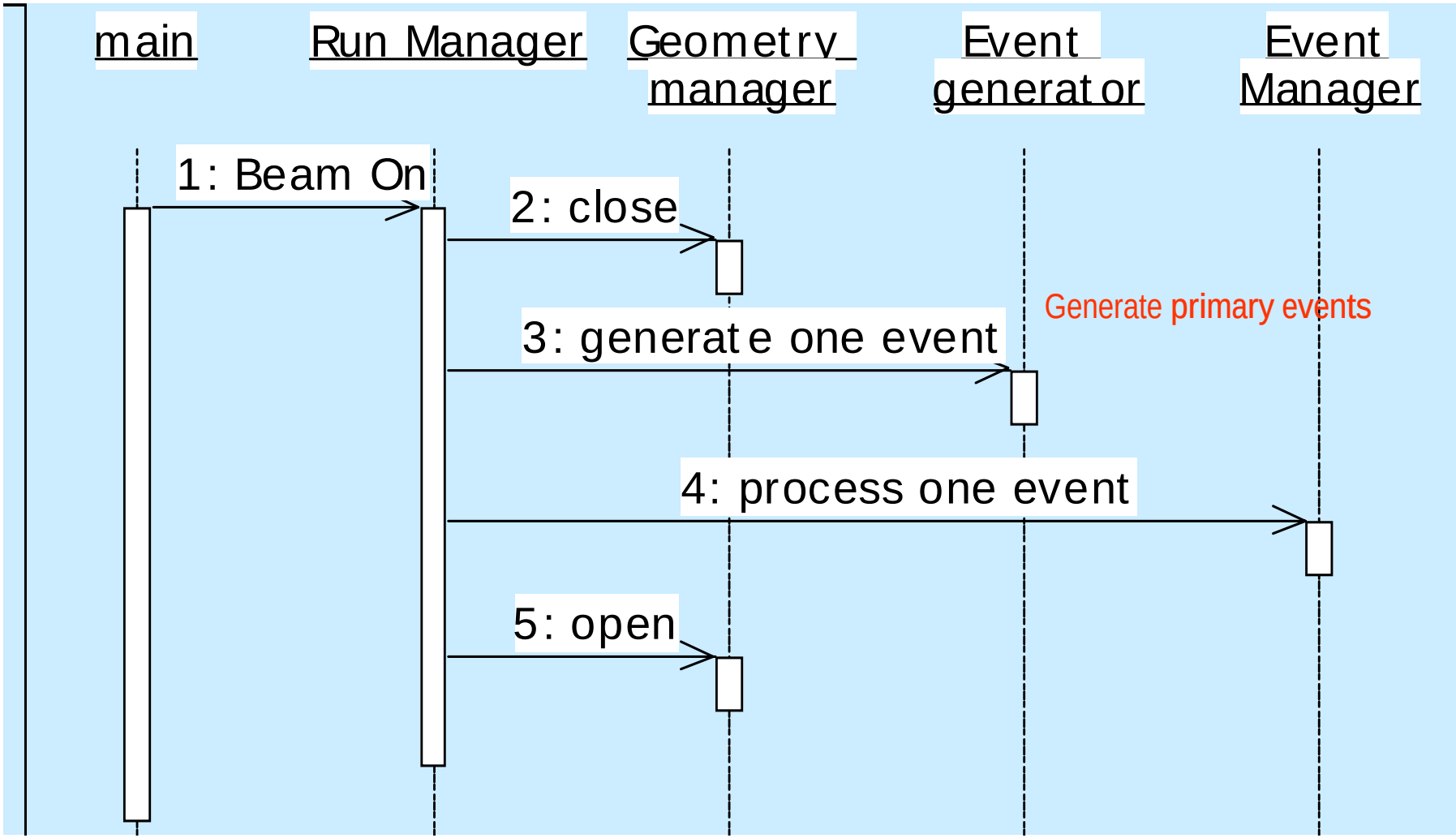
BrachyEventAction

Retrieve voxel identifier and energy deposit in the phantom from the hit

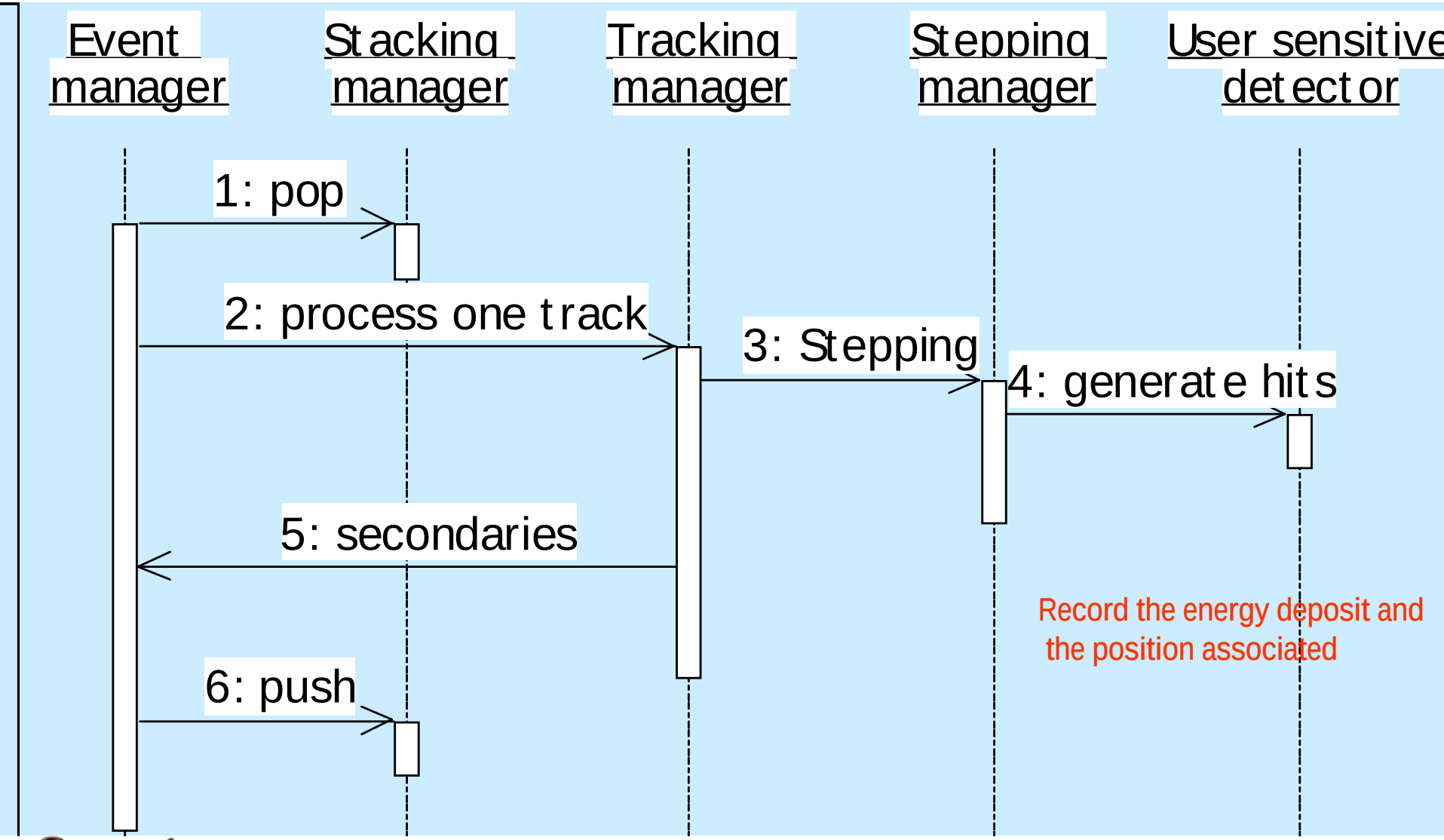
Initialisation



Beam On

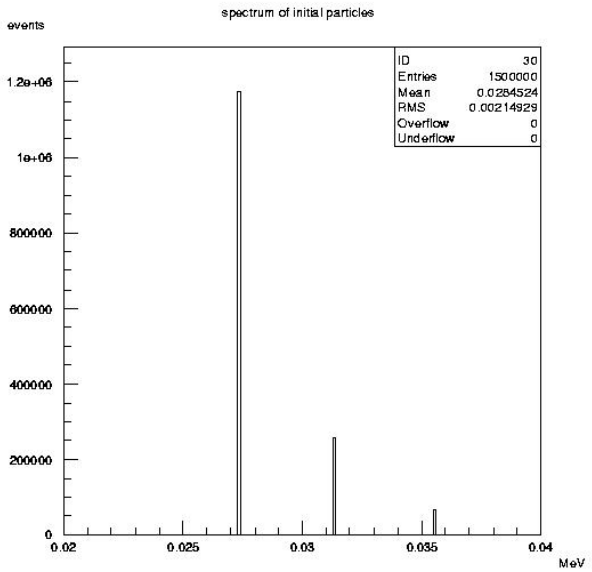

main
Run Manager
Geometry_ manager
Event generator
Event Manager
1: Beam On
2: close
Generate primary events
3: generate one event
4: process one event
5: open

Event processing



You should get something like this…

Primary particle Energy Spectrum (1D histogram)



Energy deposit (2D histogram)