

# Geant 4

Example of User Application

The Geant4 Brachytherapy  
Advanced Example

<http://cern.ch/geant4>

# Contents

---

- How to develop a Geant4 application
  - User Requirements
  - Design
  - Implementation
- Overview of the simplified version of Geant4 brachytherapy advanced example

# Before starting

Documentation: <http://geant4.web.cern.ch/geant4>

click on documentation

click on User's Guide: For Application Developers

very useful !

Documentation about the Geant4 brachytherapy application:

[www.ge.infn.it/geant4](http://www.ge.infn.it/geant4)

For example, a process model is the  
Unified Software Development Process (USDP)  
*Iterative-incremental method*

- Collection of the User Requirements

- Study of the experimental set-up:
  - involved particles,  
involved physics,  
detectors
- What is the scope of the simulation

- Design

*Project of the software structure*

- Implementation

- Test

# Capture User Requirements

*Define the scope of the software system to be built  
(“what it should do”)*

# How to collect User Requirements

- Study of the experimental set-up:
  - involved particles,
  - involved physics,
  - detectors
- What is the scope of the simulation

## Example of User Requirement

UR1. The user shall be able to define electromagnetic processes for gamma

Need: Essential

Priority: High

Source: *who asks for the user requirement*

# User Requirements

The application provides the simulation of energy deposit of a I-125 brachytherapeutic source in a phantom

- **Geometry** {
  - 1. The phantom is a box
  - 2. The radioactive source is in the center of the phantom
  - 3. The user shall be able to change the absorber material of the phantom
  - 4. The dose should be collected 1mm wide voxels
- **Physics** {
  - 1. Defined particles: e+, e-, gamma
  - 2. Electromagnetic processes
- **Analysis** {
  - 1. The user shall be able to calculate the total absorbed energy in the phantom:
    - 3D distribution in the volume
    - 2D distribution in the plain containing the source
- **Visualisation** {
  - 1. The user shall be able to visualise the geometry involved and the trajectories of the particles

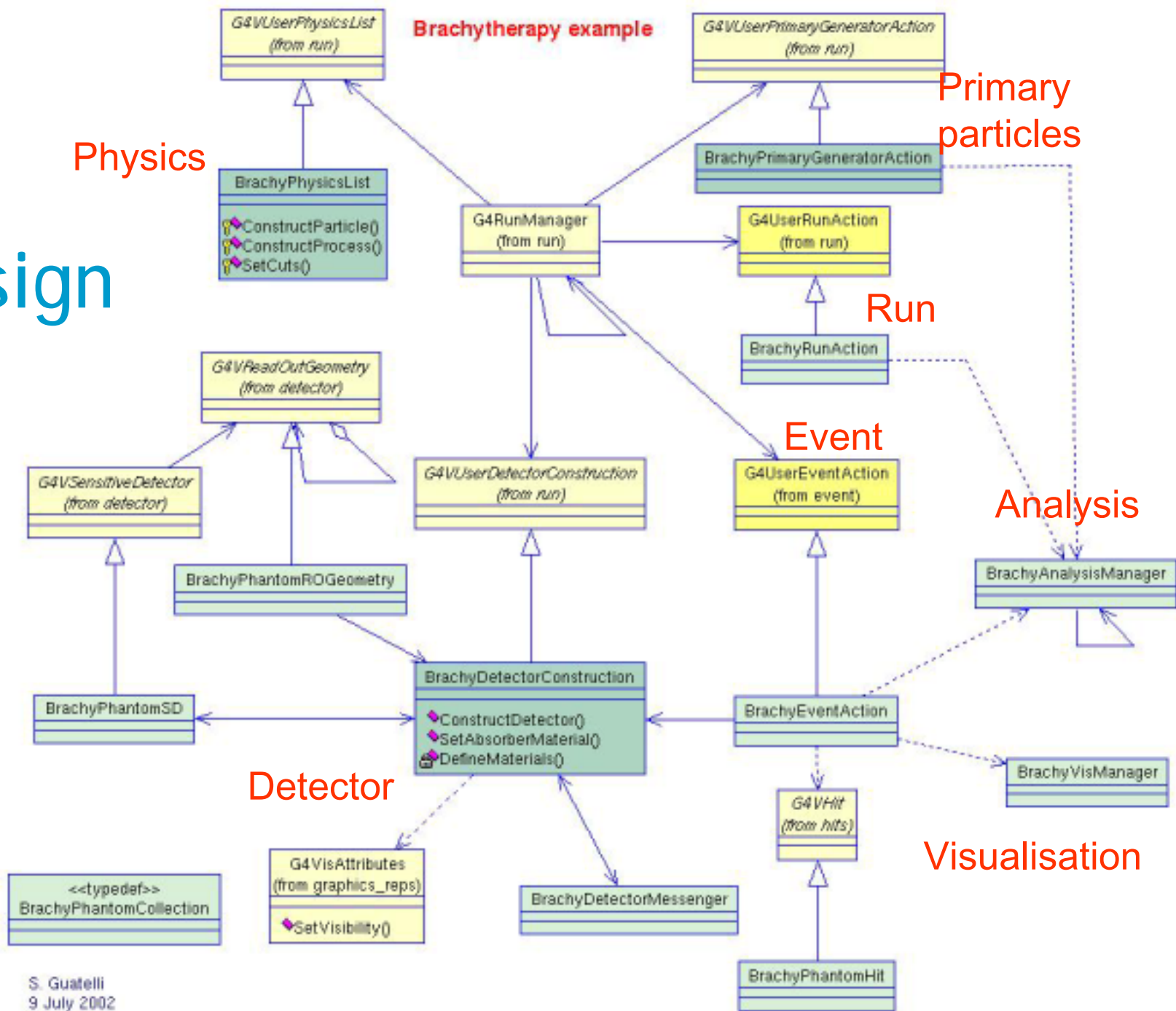
# Design

*Define the structure of the software by a  
software engineering point of view*

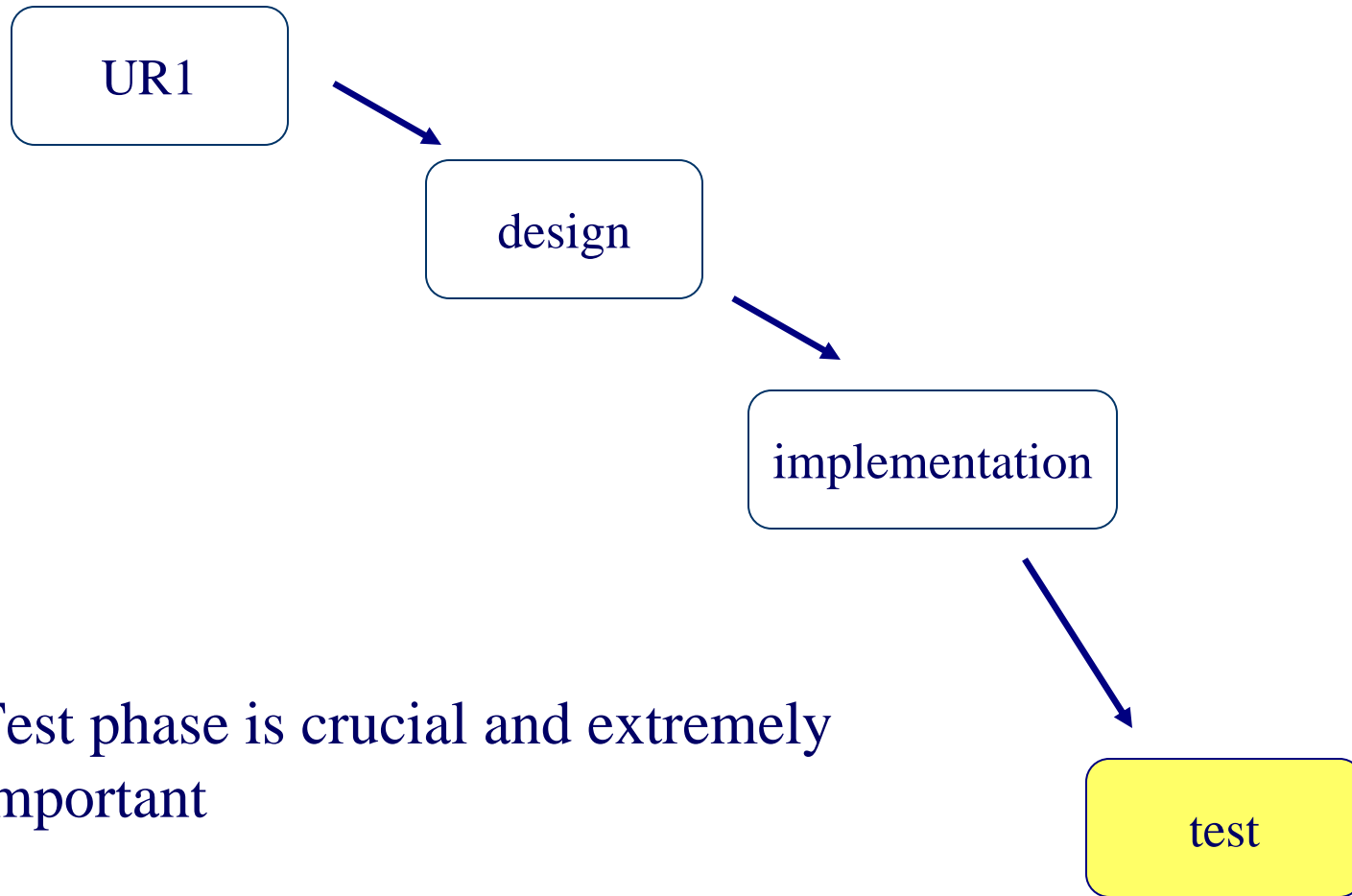


# OOAD

# Design



# Software development



# Test

---

- Important and crucial issue
- Always with respect to experimental data
- Comparisons in respect to other MC toolkit are just curiosity

## Levels of tests:

### ➤ Microscopic test

Test on components of the application (i.e. Physics models adopted)

### ➤ Macroscopic test

Test on the experimental set-up (physics, geometry, etc. together)

# Iterative, incremental method

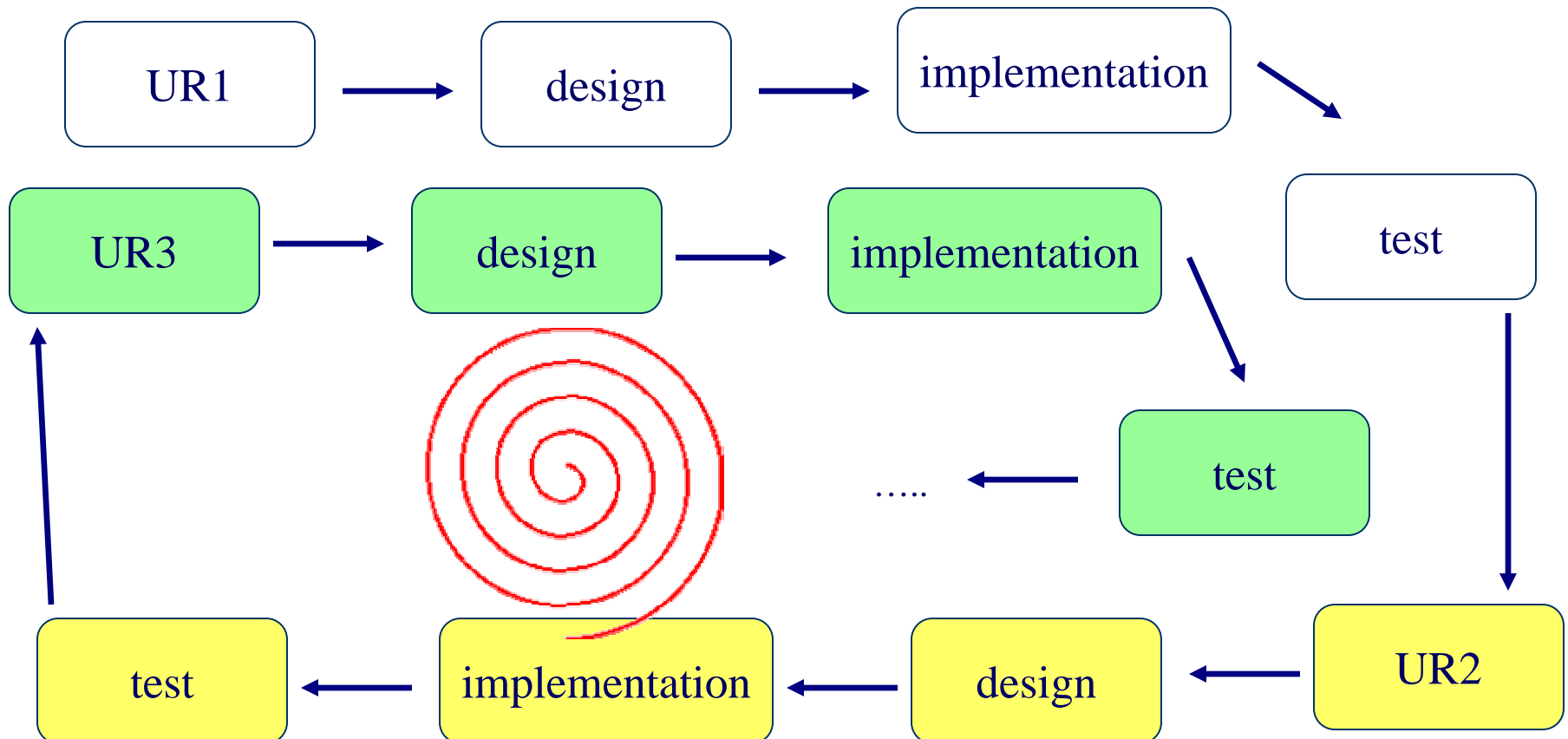
- UR1
- UR2
- UR3

Gain feedback on the developed software:

Examples: - Is the UR satisfied?

- Do we need new UR?

- How can we improve the software ?



# Implementation

# Implementation

---

## Brachytherapy example

header files in include/\*.hh, source code in src/ \*.cc

main in Brachy.cc

macro: VisualisationMacro.mac

## Classes

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

# How to run

---

Define necessary environment variables

`source ...`

How to compile and link

`gmake`

How to run

`$G4WORKDIR/bin/Linux/Brachy`

# Run Brachytherapy

1. It will appear: the visualization of the box
2. It will appear: |idle> (interactive mode)
3. Type */run/beamOn number of events*
4. The simulation is executed
5. Type *exit*

At the |idle> prompt,  
type *help*,  
information  
about interactive commands

About Visualization

OGIX : Immediate visualization  
No images saved!

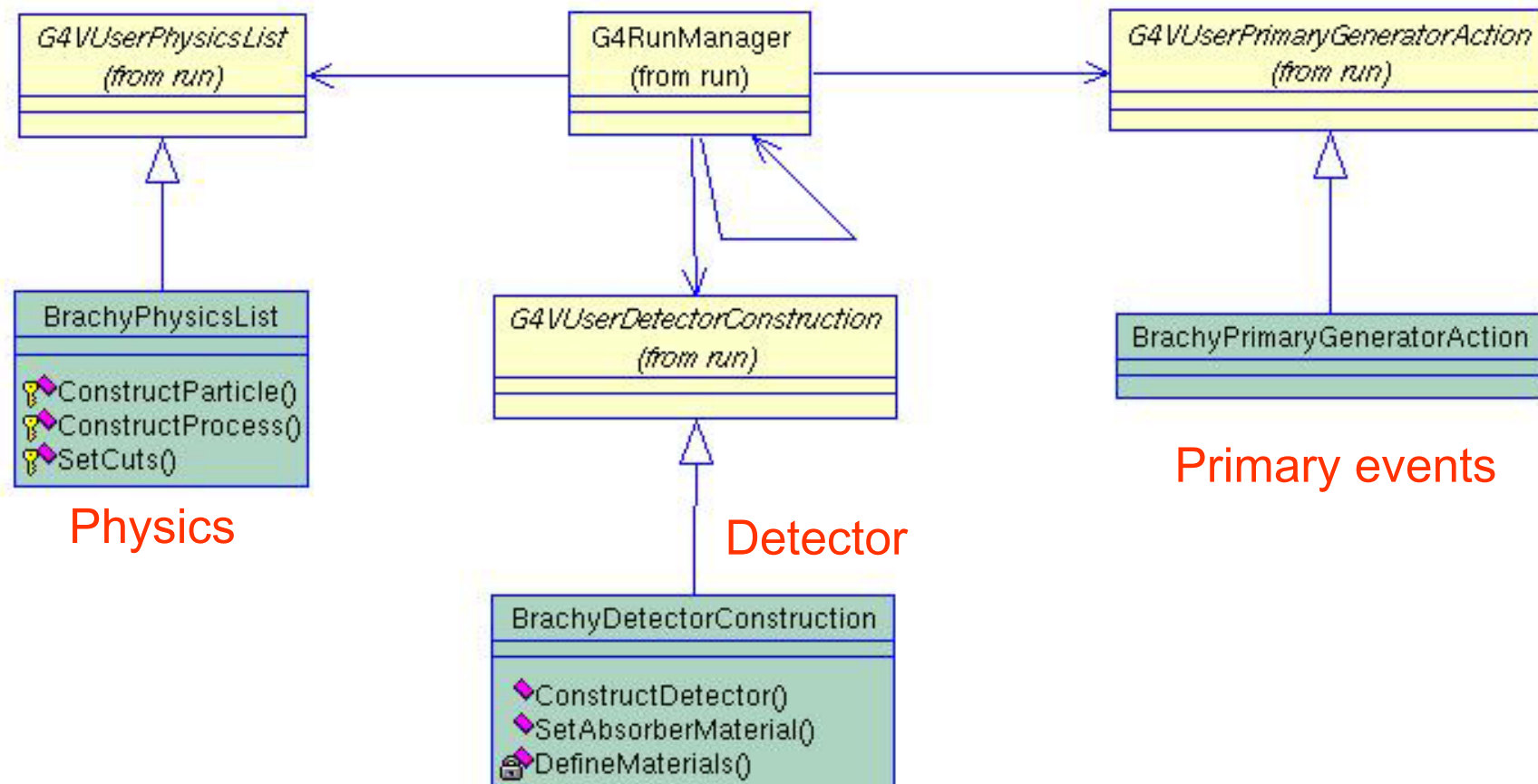
DAWN : Interactive panel  
images saved

Default visualization driver  
OGIX  
defined in VisualisationMacro.mac

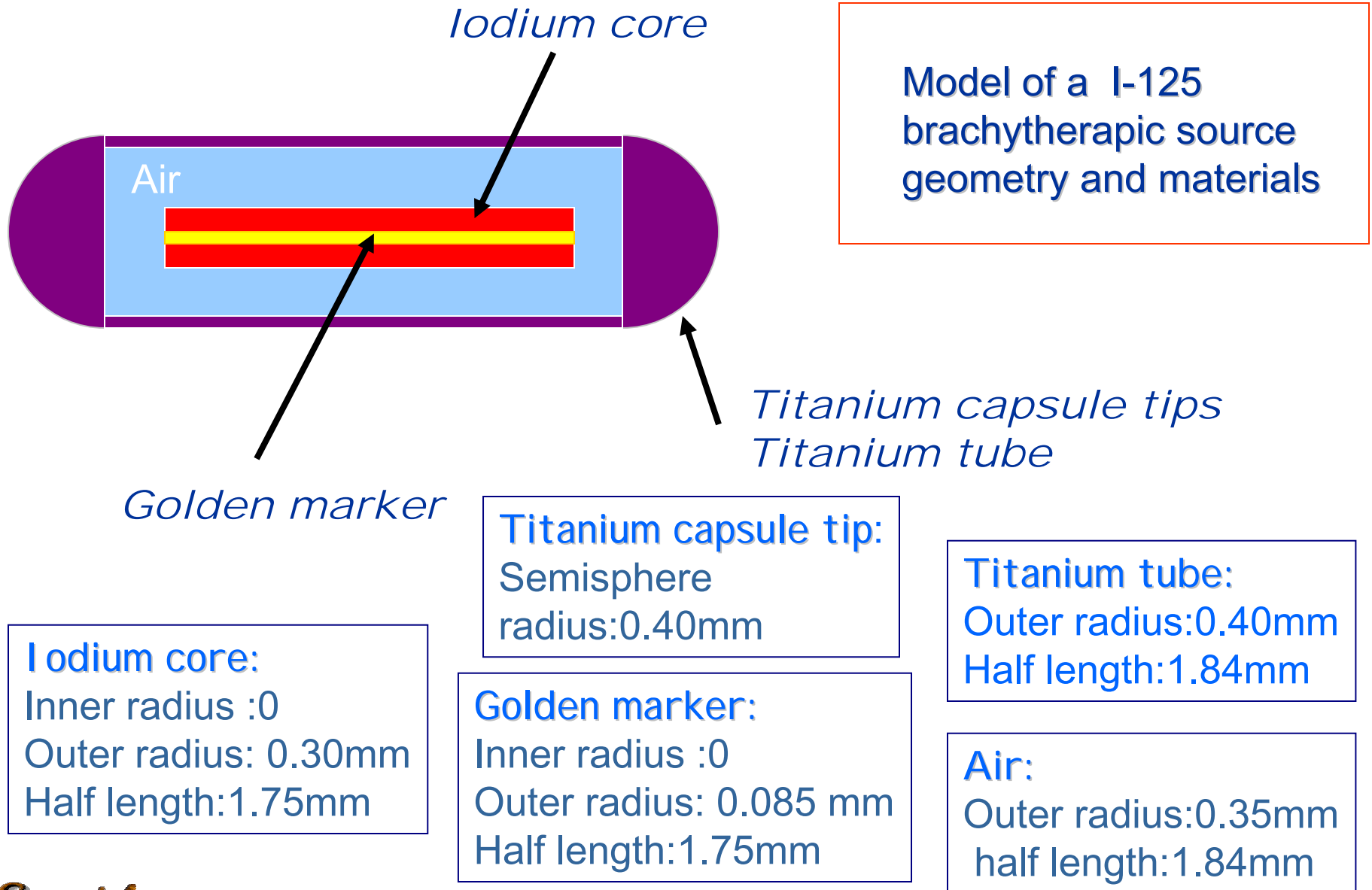


# Mandatory user classes

## Brachytherapy example: mandatory user classes

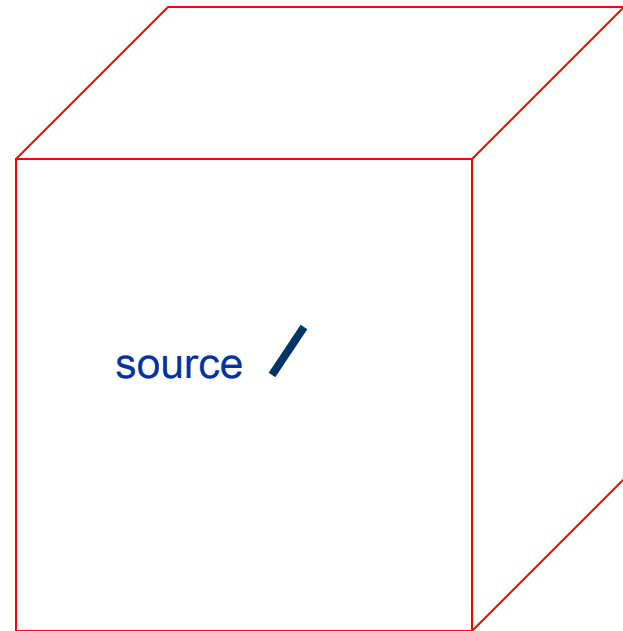


# BrachyDetectorConstruction



# BrachyDetectorConstruction

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



# ConstructSource()

// source Bebig Isoseed I-125 ...



.... construct iodine core and golden marker...  
the mother volume is an air tube

# 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

```
void BrachyPhysicsList::ConstructEM()
```

```
{ theParticleIterator->reset();  
  while( (*theParticleIterator)() ){  
    G4ParticleDefinition* particle = theParticleIterator->value();  
    G4ProcessManager* pmanager = particle->GetProcessManager();  
    G4String particleName = particle->GetParticleName();  
    if (particleName == "gamma") {  
      lowePhot = new G4LowEnergyPhotoElectric("LowEnPhotoElec");  
      pmanager->AddDiscreteProcess(new G4LowEnergyRayleigh);  
      pmanager->AddDiscreteProcess(lowePhot);  
      pmanager->AddDiscreteProcess(new G4LowEnergyCompton);  
      pmanager->AddDiscreteProcess(new G4LowEnergyGammaConversion);  
    } else if (particleName == "e-") {  
      lowelon = new G4LowEnergyIonisation("LowEnergyIoni");  
      loweBrem = new G4LowEnergyBremsstrahlung("LowEnBrem");  
      pmanager->AddProcess(new G4MultipleScattering, -1, 1,1);  
      pmanager->AddProcess(lowelon, -1, 2,2);  
      pmanager->AddProcess(loweBrem, -1,-1,3);  
    } else if (particleName == "e+") {...}  
  }  
}
```

## BrachyPhysicsList

Set EM processes  
for e-, e+, gamma

# BrachyPrimaryGeneratorAction

- I-125 delivers gamma

Energy(keV)

Probability

- Gamma Energy Spectrum

27.4

0.783913

31.4

0.170416

- Random direction

35.5

0.045671

- Random position inside the iodine core

```
void BrachyPrimaryGeneratorAction::GeneratePrimaries(G4Event* anEvent)
```

```
{  
  .....  
  particleGun->SetParticlePosition(position);  
  particleGun -> SetParticleDirection(direction);  
  particleGun -> SetParticleEnergy(energy);  
  particleGun->GeneratePrimaryVertex(anEvent);  
}
```

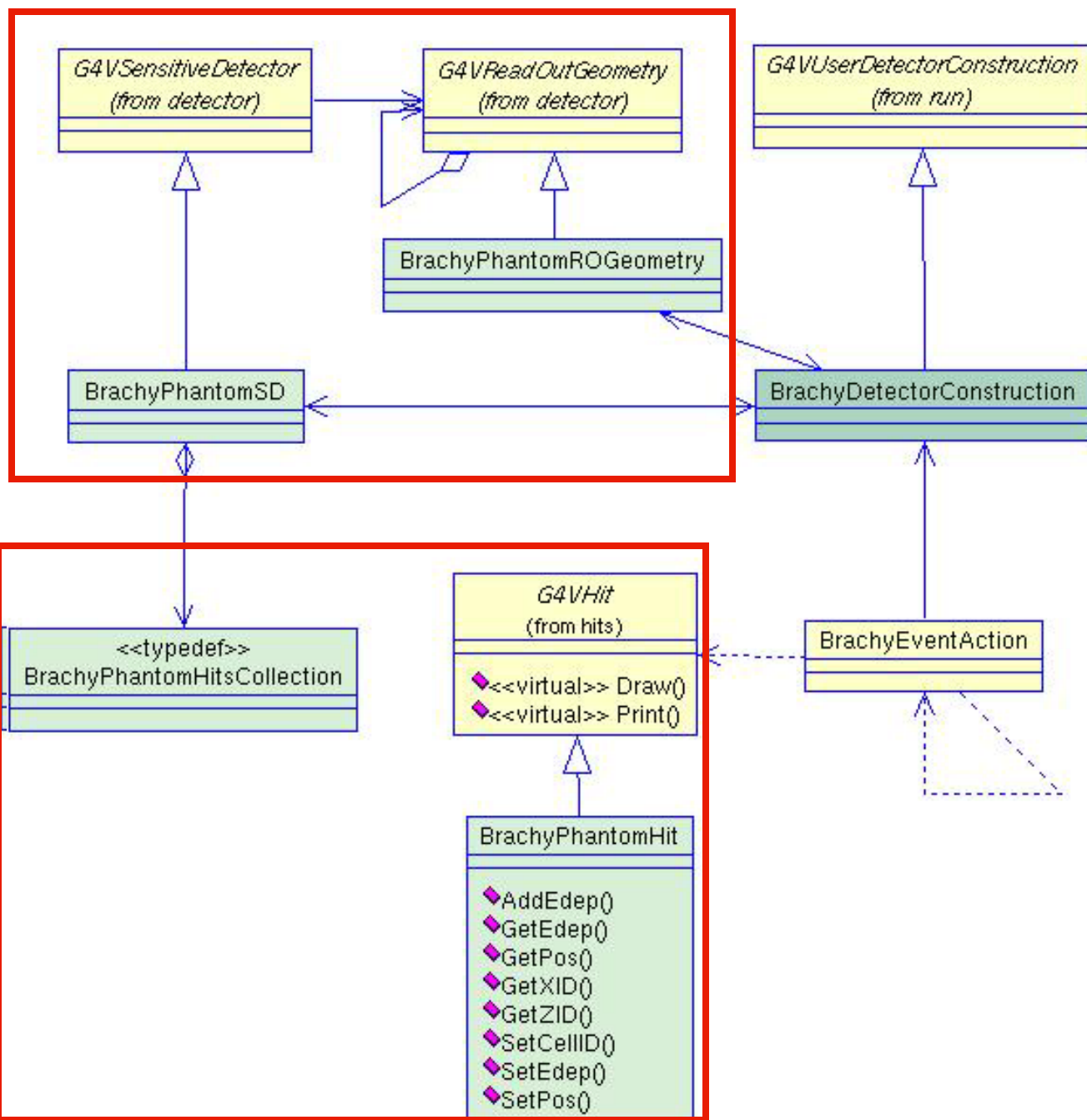
## Brachytherapy example: detector response

### Energy deposit

How to retrieve the energy deposit in the phantom

Concepts:

- Hits
- Sensitive Detector
- Readout Geometry





# Set Sensitive Detector and RO Geometry

```
void BrachyDetectorConstruction::ConstructSensitiveDetector()
{
    G4SDManager* pSDManager = G4SDManager::GetSDMpointer();
    if(!phantomSD){
        phantomSD = new BrachyPhantomSD(sensitiveDetectorName,numberOfVoxelsAlongX,
                                         numberOfVoxelsAlongZ);

        G4String ROGeometryName = "PhantomROGeometry";
        phantomROGeometry = newBrachyPhantomROGeometry(ROGeometryName,
            phantomDimensionX,phantomDimensionZ,numberOfVoxelsAlongX,numberOfVoxelsAlongZ);
        phantomROGeometry->BuildROGeometry();
        phantomSD->SetROgeometry(phantomROGeometry);
        pSDManager->AddNewDetector(phantomSD);
        PhantomLog->SetSensitiveDetector(phantomSD);
    }
}
```

In PhantomDetectorConstruction

**Geant 4**

```
BrachyPhantomROGeometry::BrachyPhantomROGeometry() {}
```

```
BrachyROGeometry::~~BrachyROGeometry() {}
```

```
G4VPhysicalVolume* BrachyPhantomROGeometry :: Build()
```

```
{
```

```
// example : X division
```

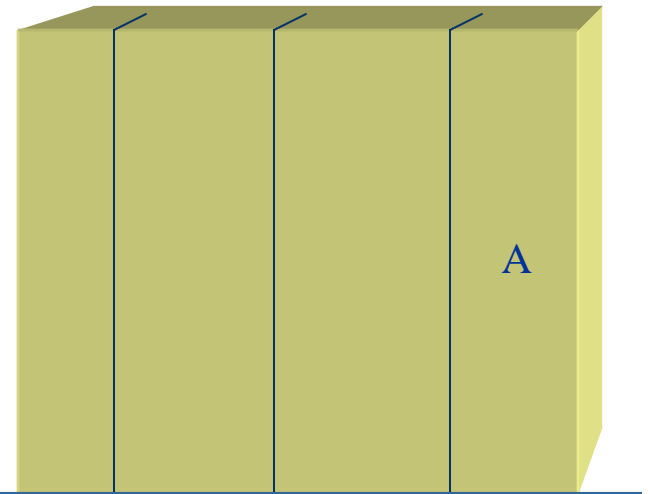
```
ROPhantomXDivision = new G4Box( ....);
```

```
ROPhantomXDivisionLog = newG4LogicalVolume(....);
```

```
ROPhantomXDivisionPhys = new G4PVR replica(....);
```

```
.....
```

```
}
```



# Sensitive Detector

```
G4bool BrachyPhantomSD::ProcessHits  
(G4Step* aStep, G4TouchableHistory* ROhist)  
{....
```

```
G4double energyDeposit = aStep->GetTotalEnergyDeposit();
```

```
....
```

```
G4VPhysicalVolume* physVol = ROhist->GetVolume();
```

```
// Read Voxel indexes: i is the x index, k is the z index
```

```
G4int k = ROhist->GetReplicaNumber(1);
```

```
G4int i = ROhist->GetReplicaNumber(2);
```

```
G4int j = ROhist->GetReplicaNumber();
```

```
.....
```

```
BrachyPhantomHit* PhantomHit = new BrachyPhantomHit(  
    physVol ->GetLogicalVolume(), i,j,k)
```

```
    PhantomHit->SetEdep(energyDeposit);
```

```
    PhantomHit->SetPos(physVol->GetTranslation());
```

Store the energy  
deposit in one hit

In PhantomSensitiveDetector

# Hits

---

- Hit is a user-defined class derived from G4VHit
- You can store various types information by implementing your own concrete Hit class:
  - position and time of the step
  - momentum and energy of the track
  - energy deposit of the step
  - geometrical information
  - etc.
- Hit objects of a concrete hit class must be stored in a dedicated collection, which is instantiated from G4THitsCollection template class

# BrachyPhantomHit (header file)

---

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

    ....

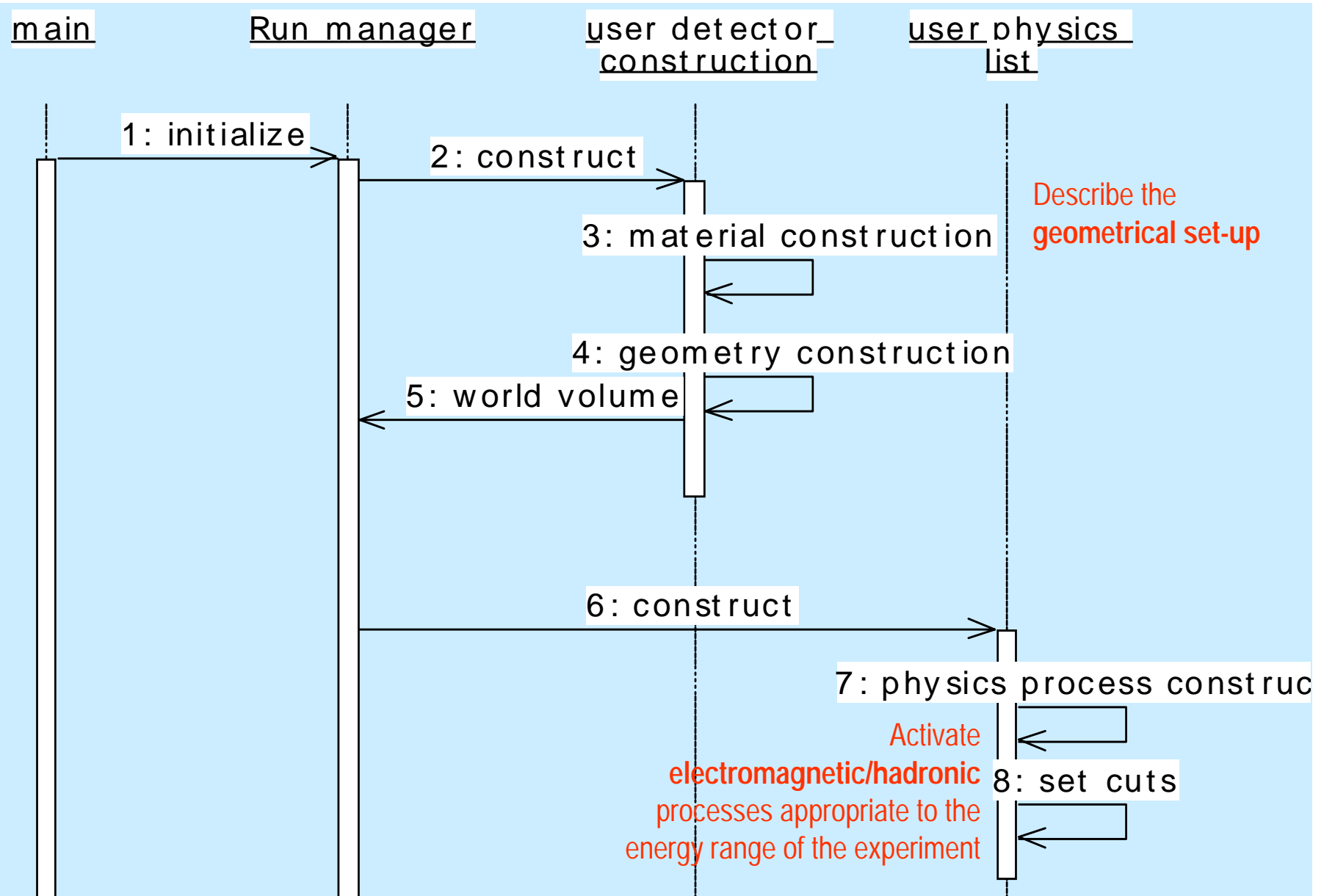
    inline G4int GetXID() {return xHitPosition;} //Get hit x coordinate
    inline G4int GetZID() {return zHitPosition;} // Get hit z coordinate
    inline G4int GetYID() {return yHitPosition;} // Get hit y coordinate
    inline G4double GetEdep() {return energyDeposit;} // Get energy deposit
    ....}
```

## BrachyEventAction

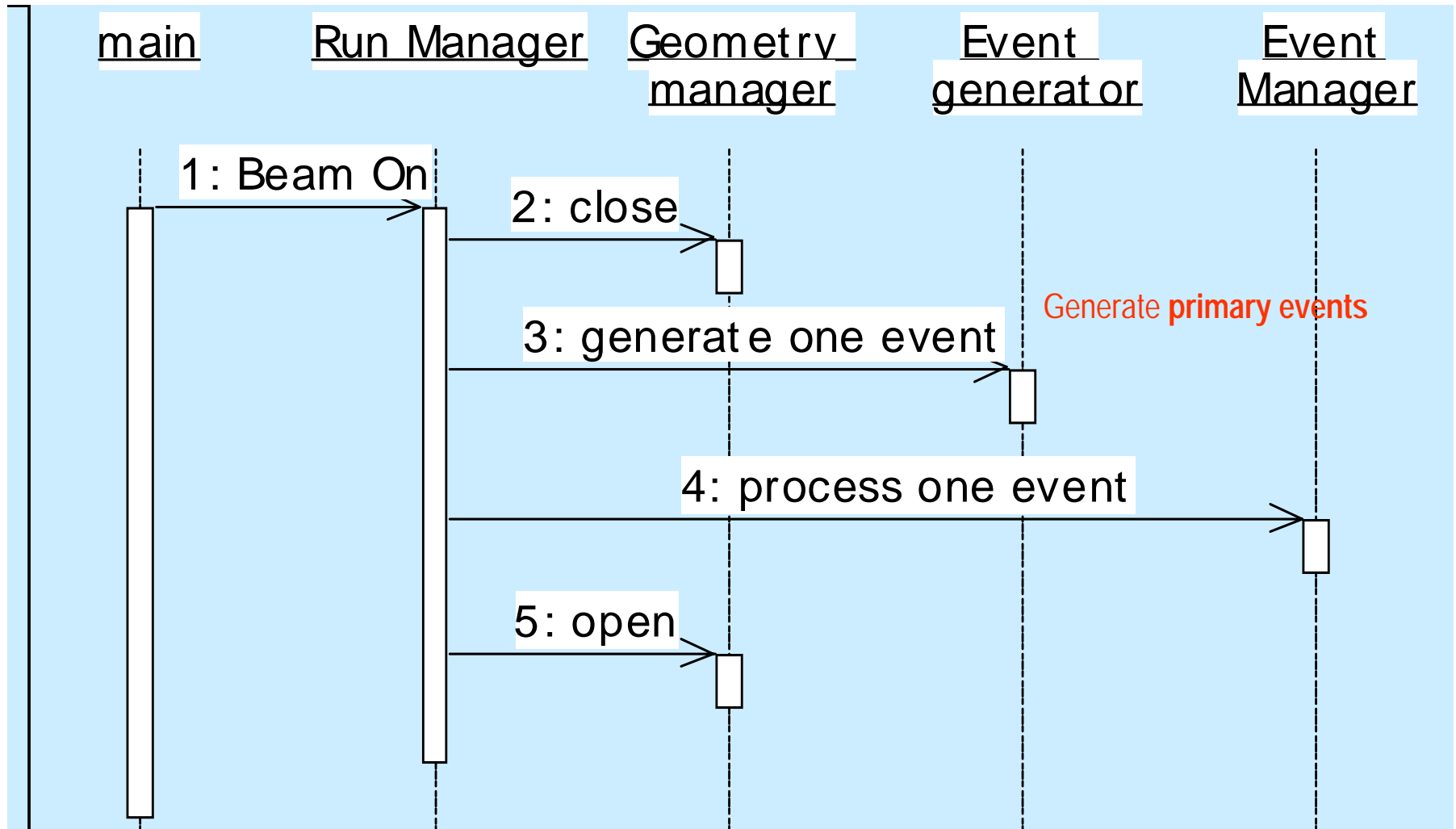
```
void BrachyEventAction::EndOfEventAction(const G4Event* evt)
{....
  G4HCofThisEvent* HCE = evt->GetHCofThisEvent();
  BrachyPhantomHitsCollection* CHC = NULL;
  if(HCE)
    CHC = (BrachyPhantomHitsCollection*)(HCE->GetHC(hitsCollectionID));
  if(CHC)
  {
    G4int hitCount = CHC->entries();
   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

Retrieve energy  
deposit in  
the phantom

# Initialisation

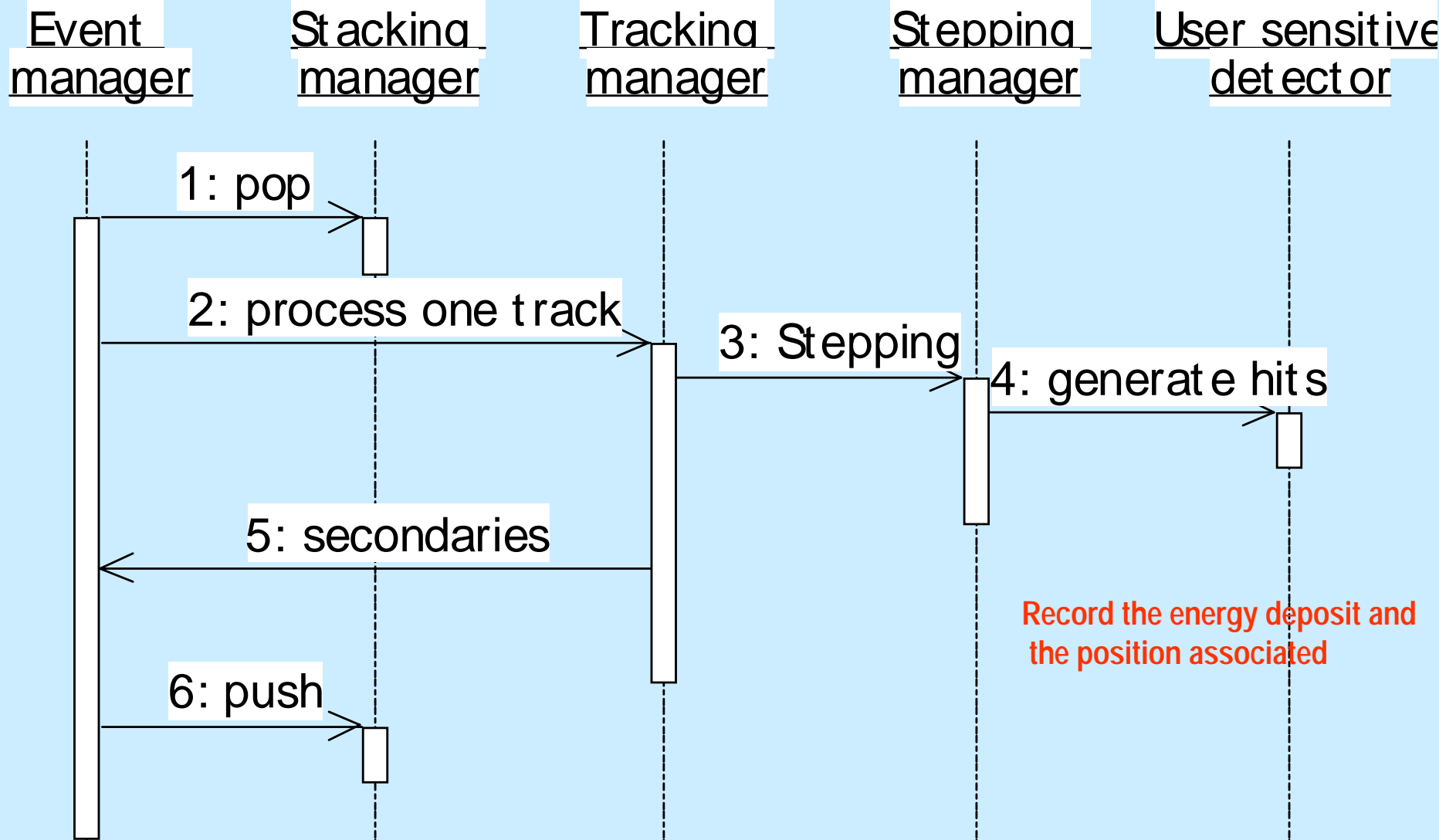


# Beam On



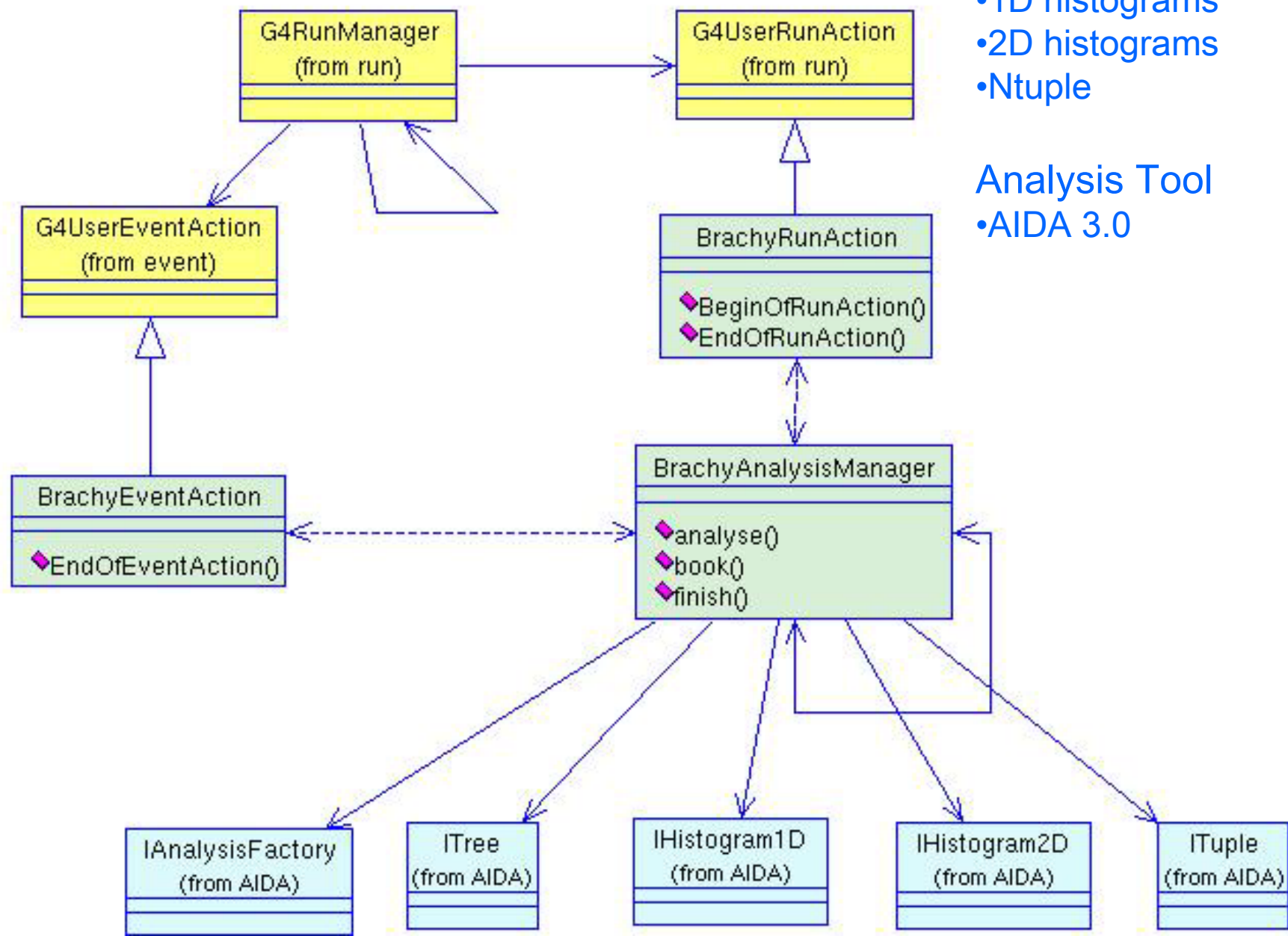


# Event processing



Record the energy deposit and the position associated

## Brachytherapy example: analysis



# BrachyAnalysisManager

BrachyAnalysisManager::BrachyAnalysisManager() :

```
....  
{  
    //build up the factories  
    aFact = AIDA_createAnalysisFactory();  
    AIDA::ITreeFactory *treeFact = aFact->createTreeFactory();  
    theTree = treeFact->create(fileName,"hbook",false, true);  
    ....  
    histFact = aFact->createHistogramFactory( *theTree );  
    tupFact = aFact->createTupleFactory ( *theTree );  
}  
void BrachyAnalysisManager::finish()  
{  
    theTree->commit(); // write all histograms to file ...  
    theTree->close(); // close (will again commit) ...
```

Create the .hbk file...

Close the .hbk file

# BrachyAnalysisManager

---

```
void BrachyAnalysisManager::book()
{
    //creating a 2D histogram ...
    h1 = histFact->createHistogram2D("10","Energy, pos",
                                     300 ,-150.,150., //bins'number,xmin,xmax
                                     300,-150.,150.  );//bins'number,ymin,ymax

    //creating a 1D histogram ...
    h2 = histFact->createHistogram1D("20","Initial Energy", 500,0.,50.);

    //creating a ntuple ...
    if (tupFact) ntuple = tupFact->create("1","1",columnNames, options);
    ....}
```

# BrachyAnalysisManager

## How to fill histograms....

```
void BrachyAnalysisManager::FillHistogramWithEnergy (G4double x, G4double z,  
G4float energyDeposit)  
{  
    //2DHistogram: energy deposit in a voxel which center is fixed in position (x,z)  
    h1->fill(x,z,energyDeposit);  
}  
  
void BrachyAnalysisManager::PrimaryParticleEnergySpectrum  
(G4double primaryParticleEnergy)  
{  
    //1DHisotgram: energy spectrum of primary particles  
    h2->fill(primaryParticleEnergy);  
}
```

# BrachyAnalysisManager

## How to fill Ntuples....

```
void BrachyAnalysisManager::FillNtupleWithEnergy(G4double xx,G4double yy,  
G4double zz, G4float en)  
{.....  
    G4int indexX = ntuple->findColumn( "x" );  
    G4int indexY = ntuple->findColumn( "y" );  
    G4int indexZ = ntuple->findColumn( "z" );  
    G4int indexEnergy = ntuple->findColumn( "energy" );  
    ntuple->fill(indexEnergy, en);  
    ntuple->fill(indexX, xx);  
    ntuple->fill(indexY, yy);  
    ntuple->fill(indexZ, zz);  
    ntuple ->addRow();  
}
```

# Analysis management

```
void BrachyRunAction::BeginOfRunAction(const G4Run*)
```

```
{
```

```
....
```

```
    BrachyAnalysisManager* analysis = BrachyAnalysisManager::getInstance();
```

```
    analysis->book();
```

```
....
```

```
}
```

```
void BrachyRunAction::EndOfRunAction(const G4Run* aRun)
```

```
{
```

```
.....
```

```
    BrachyAnalysisManager* analysis = BrachyAnalysisManager::getInstance()
```

```
.....
```

```
    analysis->finish();
```

```
....
```

```
}
```

Booking  
histograms and  
ntuple ...

...Closing the  
hbook file

In BrachyRunAction

# Energy deposit

```
void BrachyEventAction::EndOfEventAction(const G4Event* evt)
{
    .... // here the energy deposit information is retrieved
    //Store information about energy deposit in a 2DHistogram and in a ntuple ...
    BrachyAnalysisManager* analysis = BrachyAnalysisManager::getInstance
    analysis->FillHistogramWithEnergy(x,z,EnergyDep/MeV);}}
    analysis->FillNtupleWithEnergy(x,y,z,EnergyDep/MeV);
    ...
}
```



# Gamma energy spectrum

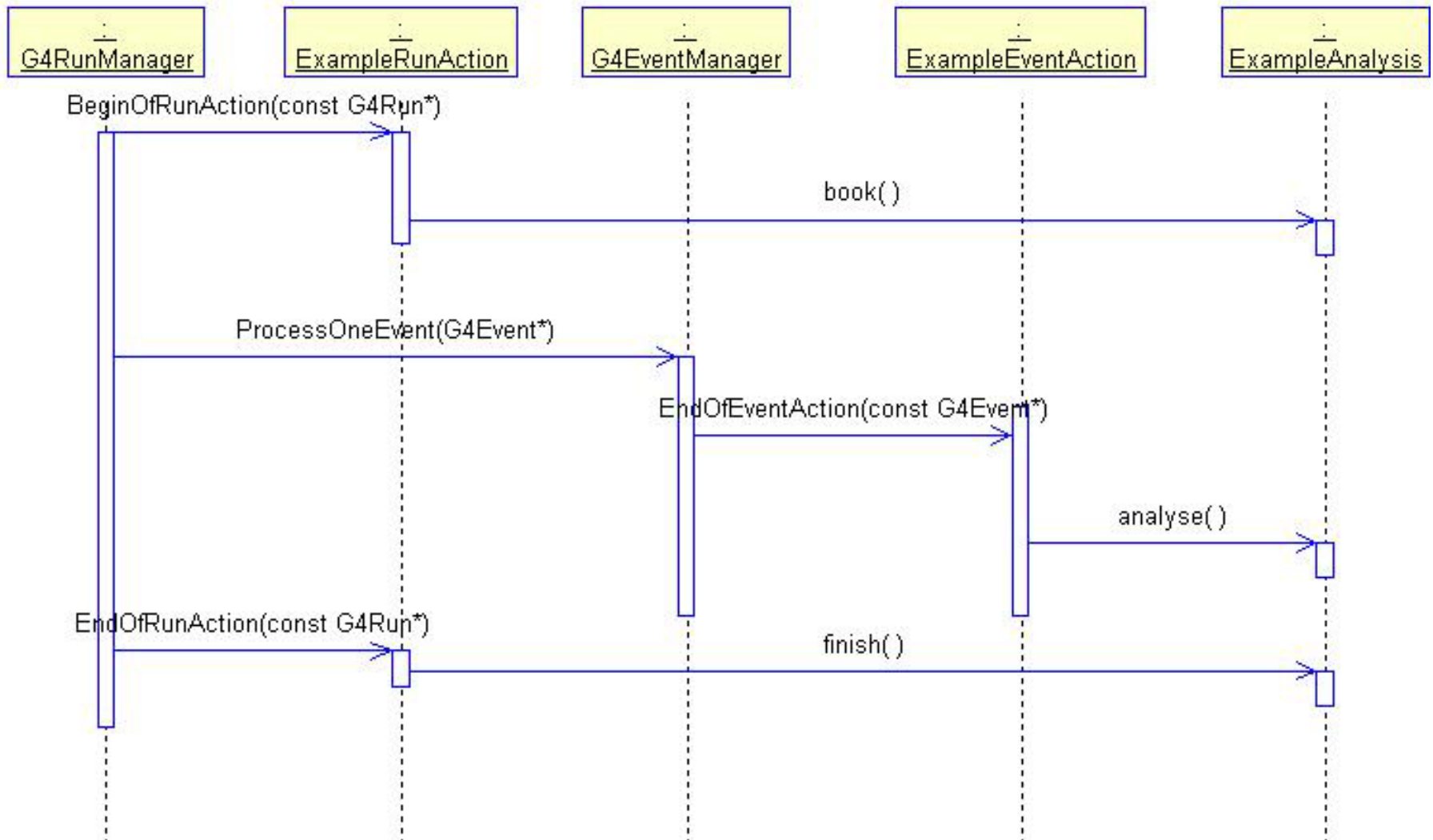
```
BrachyPrimaryGeneratorAction:: GeneratePrimaries(G4Event* anEvent)
{
    //Store the initial energy in a 1D histogram
    analysis-> PrimaryParticleEnergySpectrum(primaryParticleEnergy/keV);
    // generate primary particle
    ...
}
```

In the BrachyPrimaryGeneratorAction

# Analysis dynamic flow

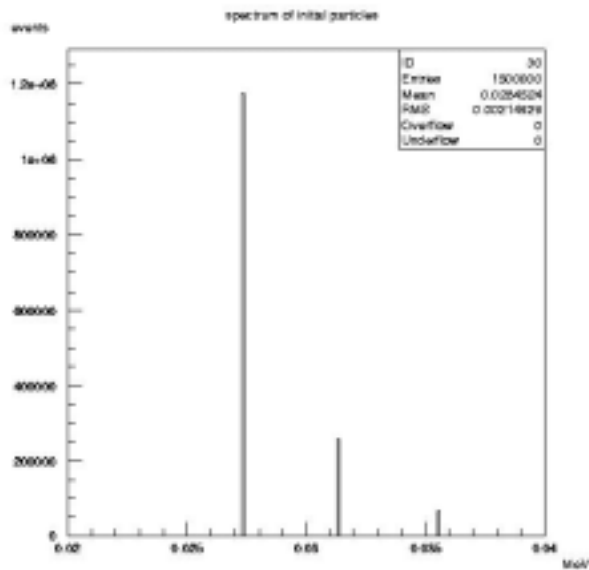
## Analysis in Geant4 advanced examples

A. Pfeiffer, M.G. Pia  
May 2002

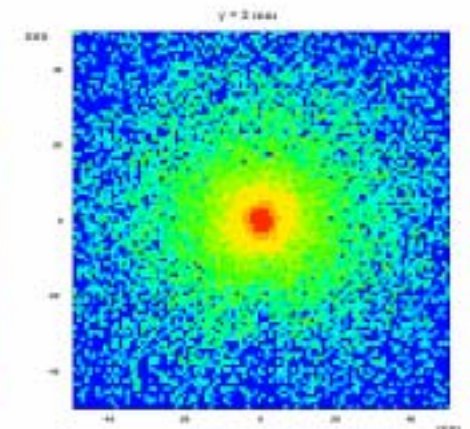
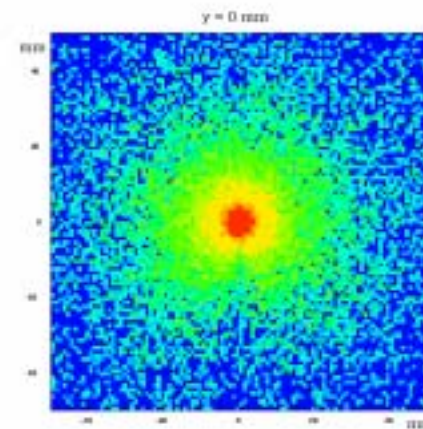
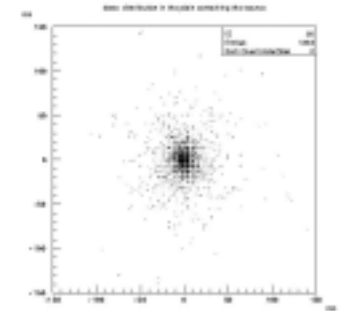


# Some Results

Primary particles  
Energy Spectrum  
(1D histogram)



Energy deposit  
(2D histogram)



# Control, monitor the simulation

# 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

## How to change the phantom absorber material

- Run `$G4WORKDIR/bin/Linux-g++/Brachy`
- (G)UI session : interactive session
- Type `/phantom/selectMaterial Lead`

The phantom absorber material now is lead

# Macro

- A macro is an ASCII file containing UI commands
- All commands must be given with their full-path directories

```
/control/verbose 1  
/run/verbose 1  
/event /verbose 1  
/phantom/selectMaterial Lead  
  
# run 10 events  
/run/beamOn 10
```

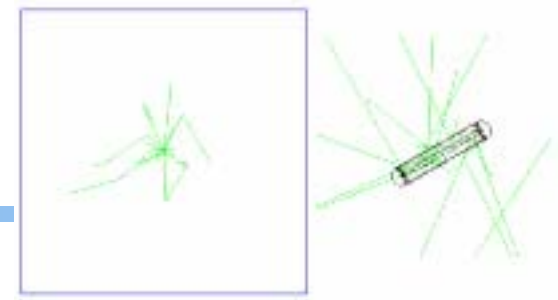
A macro can be executed by

- /control/execute
- /control/loop
- /control/foreach

in UI session

A macro can be executed also typing:  
\$G4WORKDIR/bin/Linux-g++/Brachy macro.mac

# Visualisation



- Control of several kinds of visualisation

- detector geometry
- particle trajectories
- hits in the detectors

- You can interface your Geant4 application with different visualisation packages

```
# Macro file for the visualisation
# create empty scene
#
/vis/scene/create
vis/open OGLIX
/vis/viewer/flush
# for drawing the tracks
/tracking/storeTrajectory 1
/vis/scene/endOfEventAction accumulate
/vis/viewer/update
/run/initialize
/run/beamOn 10
```

VisualisationMacro.mac



# More about visualisation

## How to change driver:

in VisualisationMacro.mac

`/vis/open OGLIX`  
`#/vis/open DAWNFILE`



`#/vis/open OGLIX`  
`/vis/open DAWNFILE`

### How to work with OGLIX:

At the `|idle>` prompt

- Type `help`
- Type the number corresponding to `/vis/`
- Information about visualization commands

Eg. rotation of the geometry  
magnification...

### How to work with DAWN:

The interactive panel appears:

- devices: choose the format of the image
- camera: choose the geometry parameters (rotation, magnification...)

# Conclusions

---

In this tutorial it is shown

- How to develop a Geant4 application
  - UserRequirements
  - Design
  - Implementation
- How to run it
- How to define primary particles
- How to define experimental set-up
- How to build a sensitive detector
- How to produce histograms and ntuples
- How to control the simulation
- How to visualise the experimental set-up