Karlsruhe Institute of Technology

# Simulation with GEANT

**Geertje Heuermann**
**Anne Zilles**

Institut für Experimentelle Kernphysik (IEKP)

# GEANT – Geometry And Tracking

- Geant4 is monte-carlo simulation toolkit
  – http://geant4.web.cern.ch/geant4

- general-purpose simulation toolkit for elementary particles interacting with matter

- HEP background
  - high energy cosmics
  - accelerated particle physics
  - nuclear physics
  - the assessment of radiation shielding and satellites
  - medical physics studies.

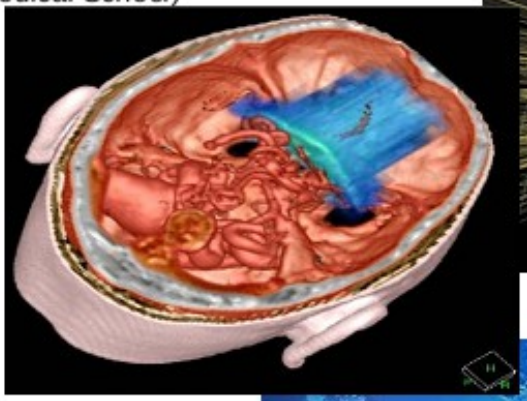
- C++ based / Object orientated

- OPEN and EXTENSIBLE

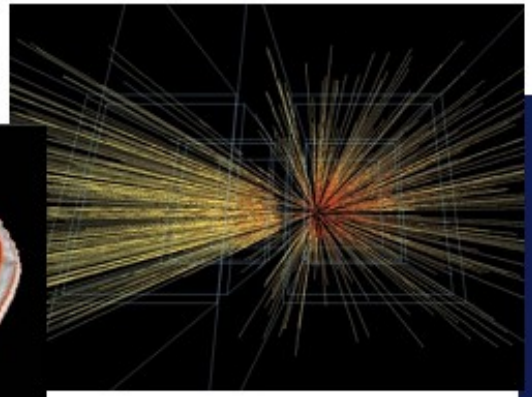
→ as a toolkit with open-source code, Geant4 can be extended in many ways
  - creating a new shape
  - new processes
  - reversing time

# Applications domains

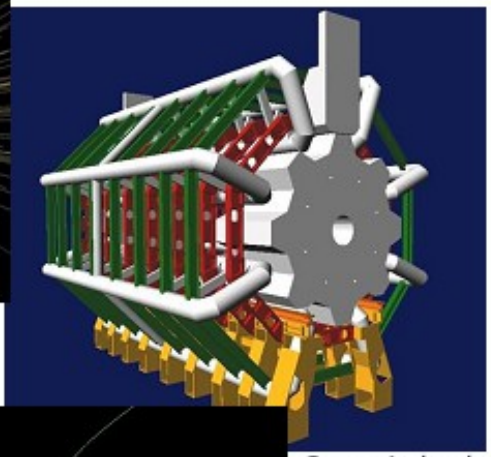
Calculating dose of protons in a clinical environment based on Geant4 (With courtesy of Harald Paganetti, Massachusetts General Hospital, Harvard Medical School) Medical School)



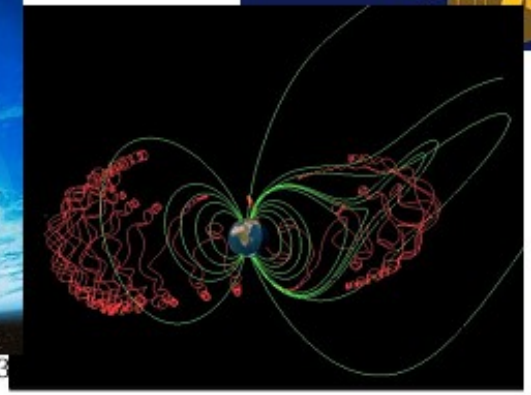
Collisions of heavy ions: an event with> 50,000 particles / event (A. De Roeck, CERN)



LHC detectors - ATLAS
(from ATLAS Detector Simulation page)



XMM-Newton X-ray telescope: the effects of the radiation environment on its instruments were modeled with Geant4 before the launch in 1999 (courtesy of ESA))



Geant4 simulation of the propagation of cosmic rays through the Earth's atmosphere (L. Desorgher, EO Fluckuger, M. Moser, R. Bütikofer, Physikalisches Institut, University of Bern)

# Brief History

- Geant4 started as RD44 project (1994-1998)
  - amongst first OO in HEP, 1st for simulation
  - April 1997: 1st alpha release
  - December 1997: 1st supported release Geant4.0.0
- First uses in production in several fields
  - Space: 1999 XMM (X-ray telescope)
  - HEP: 2001 BaBar, 2004 ATLAS/CMS/LHCb
- Regular one public releases every year
  - Beta releases also available
  - announcements on Collaboration Web pages
  - Last Geant4 release: 9.6 – November 2012
- Today: a world-wide collaboration of more than 100 scientists in more than 10 large experiments of different research fields

# The Advantages of Monte-Carlo-Methods

- break down a complex problem into many simple problems that are treated separately for a random initial state
  → break down a long particle track into many small steps
- look at each detector component and each particle seperately
- treat physics processes consecutively instead of having to worry about everything at once
- The output of our simulated detector is qualitatively the same as of the real detector

**But BEWARE**

- A simulation will only produce what you put in it

  → A simulation will help you understand your detector but it's a tool without scientific value by itself

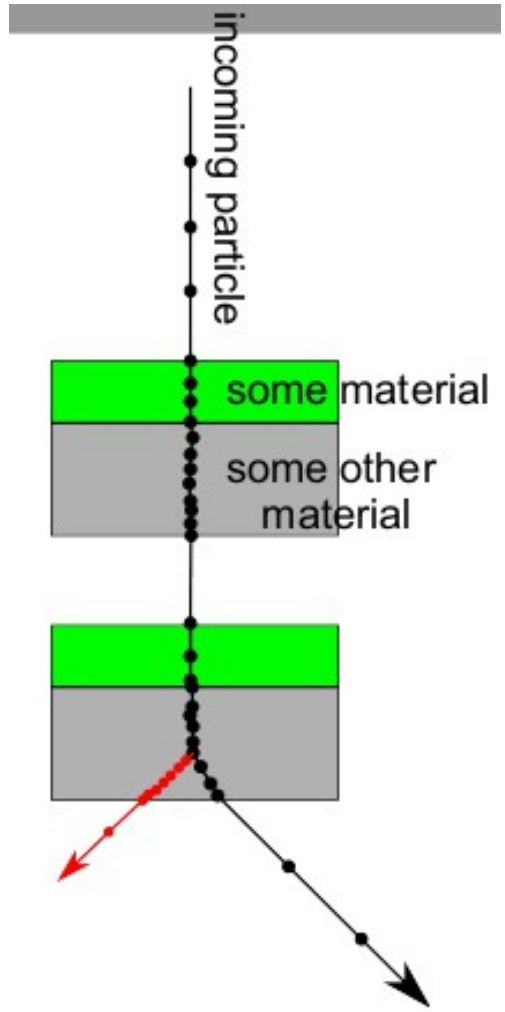
# Setting up GEANT4

- What you need:
  - compatible platform
  - One or more visualization libraries (possibly from systems, e.g. OpenGL)
- Coding is needed
  - Modify existing C++´code´ to describe your setup
  - create own class to describe e.g. a magnetic field
- Download & Install CLHEP (version 2.1.3.1)
  → http://proj-clhep.web.cern.ch/proj-clhep/
- Download & Install ROOT
  → http://root.cern.ch/drupal/content/downloading-root
- Download & Install GEANT4 (+ data files)
  → http://geant4.cern.ch/support/download.shtml
- GEANT4 offers a lot of options when installing
  → read Installation Guide:
  http://geant4.web.cern.ch/geant4/UserDocumentation/UsersGuides/InstallationGuide/html/

# Under the hood of GEANT4

- break down particle track into small steps
- store particle momentum, positions before and after last step, polarization...
- apply processes between steps
  → processes may modify particle momentum, direction, polarization and/or create new particles

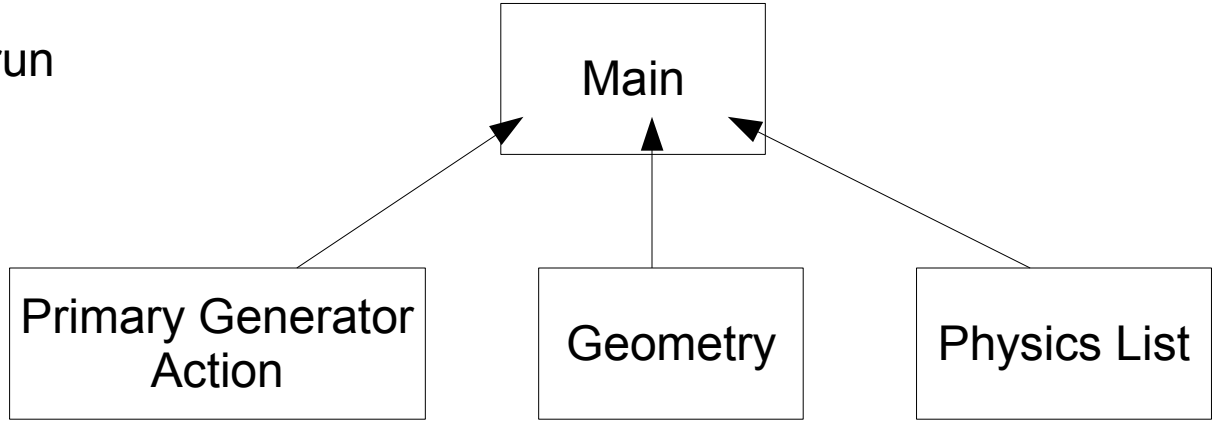
  → processes may also modify current volume (e.g. store an energy deposit)
- newly created particles are also tracked



# Basics

- Simplest Geant simulation needs three classes
  - Geometry
  - Primary Generator Action
  - Physics List

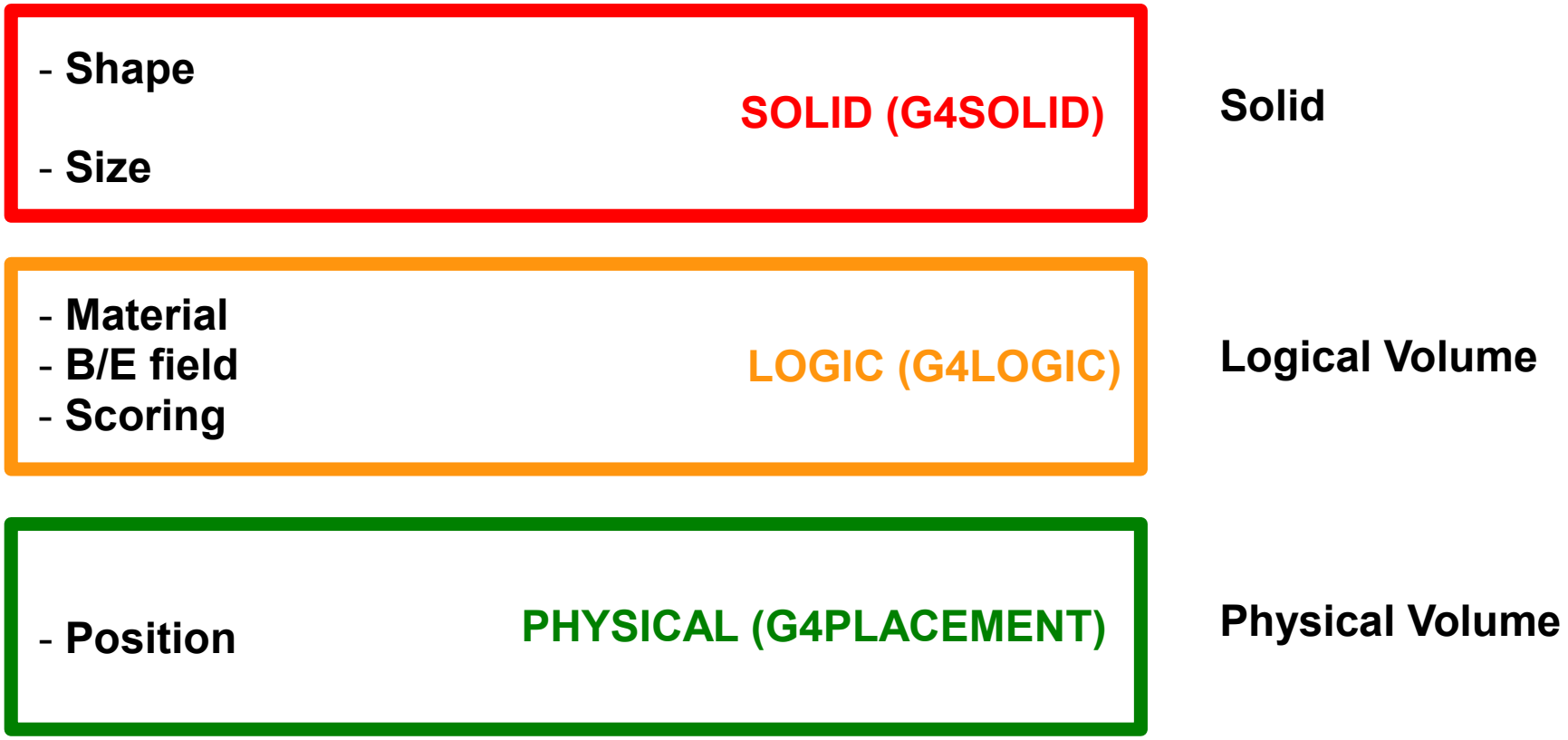
  and a main to run




- Running Geant4:
  - Use the command line
  - Hard code your commands into your main
  - Use a macro → a script containing a list of commands executed sequentially

# Geometry – Basics

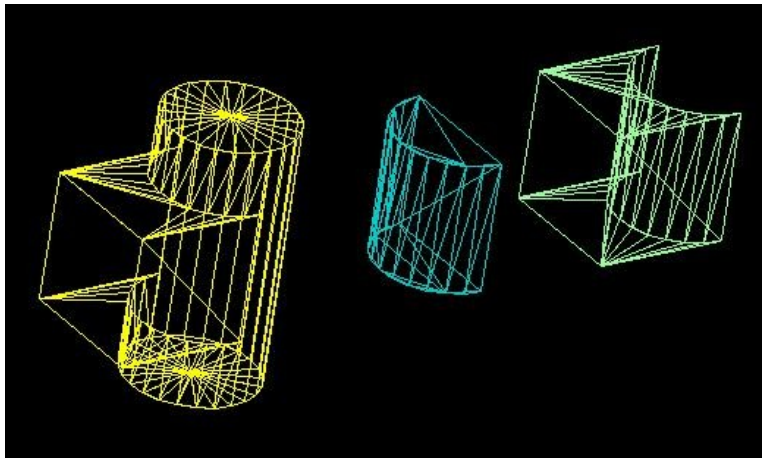
Define all objects within the world by

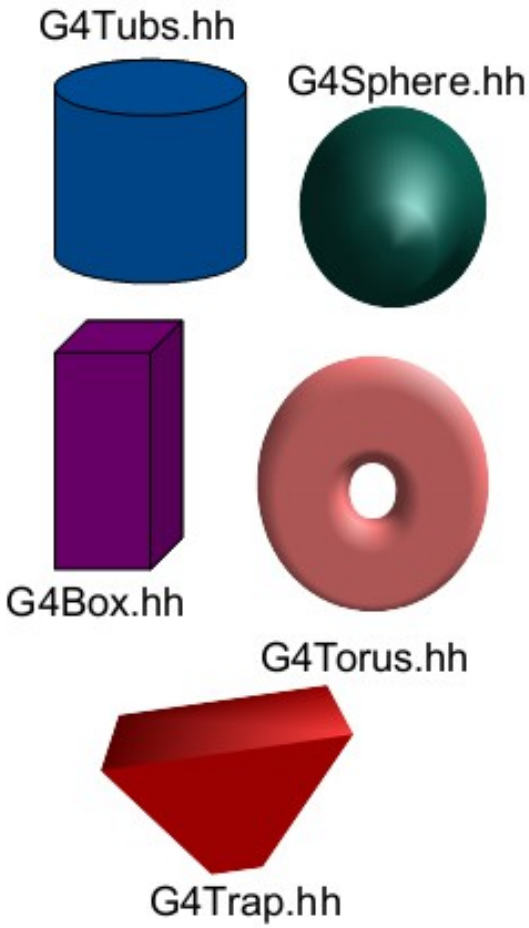


# Geometry - Shape

The simulated detector geometry and materials are defined by a class extending G4VUserDetectorConstruction

- Use predefined volumes from GEANT4
- Each volume can have any number of sub-volumes of different materials
- You can create unions & intersections of volumes





# Materials - Elements

Defining your materials

- **Materials** are defined based upon their chemical structure
- First define your **element(s)**
- Then define your material
- Assigned weighted quantities of element to material:
  Atomic number, molar mass, density

\\ Water H2O:

molMass = 1.01*g/mole;

**G4Element* elH = new G4Element(name= "Hydrogen",symbol="H" , z= 1., molMass);**

molMass = 16.00*g/mole;

**G4Element* elO = new G4Element(name= "Oxygen", symbol="O" , z=8., molMass);**

density = 1.000*g/cm3;

**G4Material* H2O = new G4Material(name="Water",density, ncomponents=2);**

**H2O->AddElement(elH, natoms=2);**

**H2O->AddElement(elO, natoms=1);**

# Defining Geometry

Shape, Size

1. Step: Solid

```
G4Box (const G4String& pName, G4double halfX, G4double halfY, G4double halfZ);
```

Material

2. Step: Logical Volume

```
G4LogicalVolume(G4VSolid pSolid, G4Material pMaterial, const G4String& Name);
```

Rotations, Translation, Position, Number Copies

3.Step: Physical Volume

```
G4PVPlacement(G4Rotationsmatrix* pRot, const G4ThreeVetor& tlate,
              G4LogicalVolume* pCurrentLogical, const G4String& pName,
              G4LogicalVolume* pMotherLogical,
              G4bool pSurfChk=false, G4int pCopyNo);
```

# Defining Geometry - Example

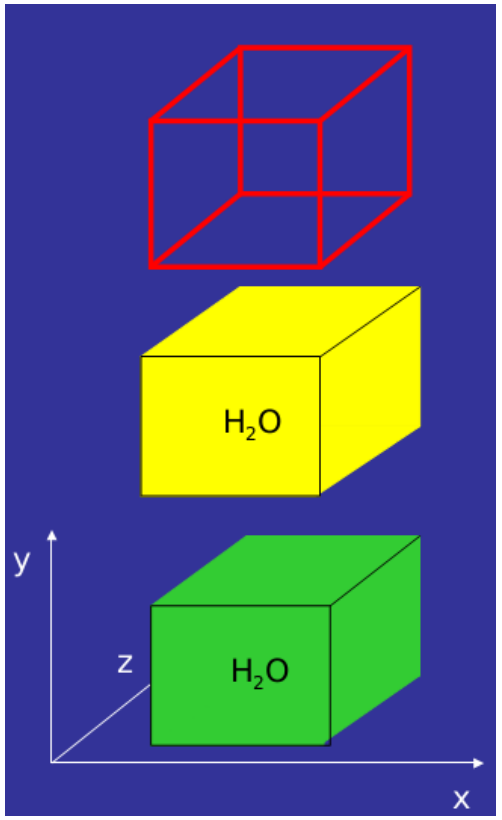
Define a cube of water (3x4x8)cm at p1 = (5,2,3)cm

→ We need a cube or box: Look at the G4Box constructor

```
(1)G4Box *thisBox =
        new G4Box("ThisBoxsName",2.5*cm,1*cm,1.5*cm);
```

```
(2) G4LogicalVolume *thisBox_log =
        new G4LogicalVolume(thisBox,H20,"BoxLogName");
```

```
(3) G4PVPlacement *thisBox_phys =
        new G4PVPlacement(0,
        G4ThreeVector(5*cm,2*cm,3*cm),
        thisBox_log,"BoxPhysName",world, // its mother
        false, 0);
```



# Primary Generator Action

Generate particles using G4GeneralParticleSource() or
G4GeneralParticleGun(G4intNumberofParticles)

- **Source for stationary sources**

```
/gps/source/intensity 1        # intensity, relative to other sources
/gps/particle gamma            # particle type
/gps/pos/type Point            #  source shape
/gps/pos/centre 0. 0. 0. cm    # source central position
/gps/ang/typ iso               #  distribution
/gps/energy 200 keV            # energy
```

- **Gun for a beam experiment**

```
/gun/particle gamma          # particle type
/gun/direction 0 0 1         # momentum direction
/gun/position 0 15 -40 cm    # position the particle originate from in world volume
/gun/energy 1.0 MeV          # kinetic energy
```

# Physics List

- Geant4 doesn't "automatically" include any physics
  (Originates from particles physics background where "new physics" would need implementing)

→ **You <u>need</u> to tell Geant4 what physical interactions you are interested in**

- Include:
  - **Particles** you want:
    - lepton
    - meson
    - baryon
    - boson
    - shortlived
    - ion
  - **Interaction** of the Particle:
    - electromagnetic
    - hadronic
    - transportation
    - decay
    - optical
    - photolepton_hadron
    - parameterisation

- Vertex/tracker/calorimeter for B physics (BaBar)
- Space Sciences and Astrophysics Applications
- High energy tracker/calorimeter for colliders (ILC)
- Space Electronics Applications
- Simple and Fast Physics List - for getting started
- Physics List for Shielding, Undergroundand High Energy Calorimetry

See: http://geant4.web.cern.ch/geant4/UserDocumentation/UsersGuides/ForApplicationDeveloper/html/

# Examples - Physics List

Which Particle?

Which processes are needed ?

//Electron
G4Electron::ElectronDefinition(); →

Multiple Scattering
Ionisation
Bremsstrahlung

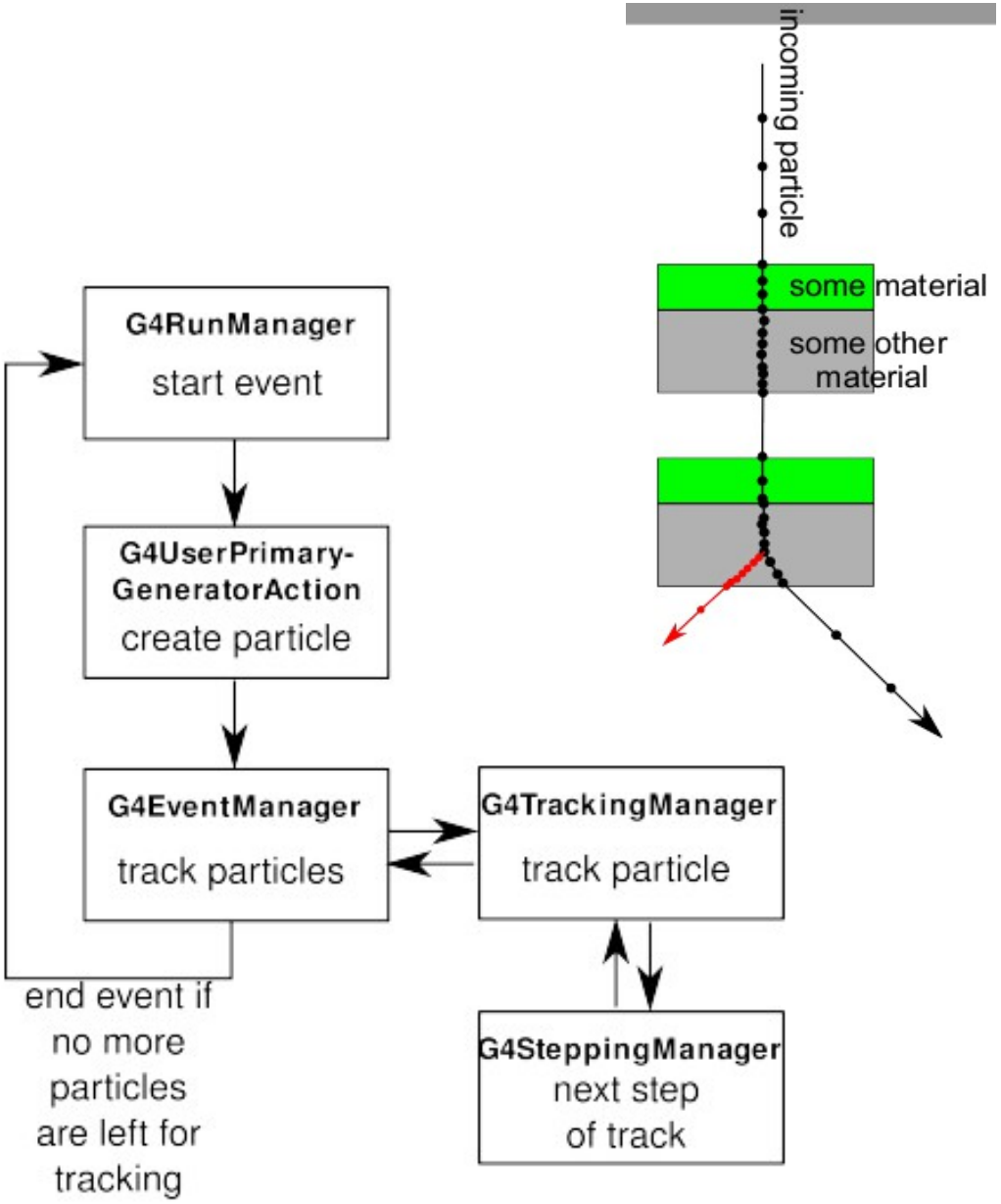
//Positron
G4Positron::PositronDefinition(); →

Multiple Scattering
Ionisation
Bremsstrahlung
Annihilation

```
if (particleName == "e-") {
//electron
pmanager-> AddProcess(new  G4eMultipleScattering , -1, 1, 1 );
pmanager-> AddProcess(new  G4eIonisation       ,        -1, 2, 2);
pmanager-> AddProcess(new  G4eBremsstrahlung ,    -1, 3, 3);
}
else if (particleName == "e+") {
//positron
pmanager-> AddProcess(new  G4eMultipleScattering ,-1, 1, 1 );
pmanager-> AddProcess(new  G4eIonisation     ,        -1, 2, 2);
pmanager-> AddProcess(new  G4eBremsstrahlung ,    -1, 3, 3);
pmanager-> AddProcess(new  G4eplusAnnihilation  ,  0,-1, 4 );
}
```

# Event Flow

- Each event starts with a single created particle
- tracking initial particle and all secondaries
- After each event data for the particle trajectories can be accessed
- GEANT4 allows intervening at any given point of the simulation



# Basic structure